UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Leveraging Security Data for a Quantitative Evaluation of Security Mitigation Strategies

Giorgio Di Tizio

Advisor

Prof. Fabio Massacci

University of Trento, Italy and Vrije Universiteit Amsterdam, The Netherlands

External reviewers

Prof. Michel van Eeten

Prof. Nick Nikiforakis

April 2023

Acknowledgements

Firstly, I must thank my Ph.D. supervisor, Prof. Fabio Massacci, for his incredible knowledge and energy. This thesis would not have been possible without his wise guidance.

I thank my mother, Anna, my father, Stefano, and my sister Federica for their unconditional support throughout all these years. A special thank goes to my best friend Nicola for his constant, never banal, presence.

Finally, I especially thank my Iubi Margarita, who has been with me during this journey. There is a beautiful view from the top of the mountain.

Abstract

Keeping users' and organizations' data secure is a challenging task. The situation is made more complicated due to the ever-increasing complex dependencies among IT systems. In this scenario, current approaches for risk assessment and mitigation rely on industry best practices based on qualitative assessments that do not provide any measure of their effectiveness. In this Thesis, we argue that the rich availability of data about IT infrastructures and adversaries must be employed to quantitatively measure the risk and the effectiveness of security mitigation strategies. Our goal is to show that quantitative measures of effectiveness and cost using security data are not only possible but also beneficial for both individual users and organizations to identify the most appropriate security plan. To this aim, we employed a heterogeneous set of security data spanning from blacklist feeds and software vulnerability repositories to web third-party dynamics, criminal forums, and threat intelligence reports. We use this data to model attackers and security mitigation strategies and evaluate their effectiveness in mitigating attacks. We start with an evaluation of filter lists of privacy extensions to protect individuals' privacy when browsing the Web. We then consider the security of billions of users accessing the Top 5K Alexa domains and evaluated the effectiveness and cost of security mitigations at different levels of the Internet infrastructure. We then evaluate the accuracy of SOC analysts in investigating alerts related to cyber attacks targeting a network. Finally, we develop methodologies for the analysis of the effectiveness of ML models to detect criminal discussions in forums and software updates to protect against targeted attacks performed by nation-state groups.

Keywords
Security mitigation, Security data, Attacker model, Risk assessment

Contents

List of Tables

List of Figures

1

"Why are such [ineffective security] policy choices made? Quite simply, the incentives on the decision makers favour visible controls over effective ones. The result is what Bruce Schneier calls 'security theatre' – measures designed to produce a feeling of security rather than the reality. Most players also have an incentive to exaggerate the threat from terrorism: politicians to 'scare up the vote' (as President Obama put it), journalists to sell more papers, companies to sell more equipment, government officials to build their empires, and security academics to get grants. The upshot is that most of the damage done by terrorists to democratic countries comes from the overreaction... Security engineers have to understand all this; we need to be able to put risks and threats in context, make realistic assessments of what might go wrong, and give our clients good advice. That depends on a wide understanding of what has gone wrong over time with various systems; what sort of attacks have worked, what their consequences were, and how they were stopped (if it was worthwhile to do so)."

Ross Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*

Chapter 1

Introduction

The entire world is relying on IT systems, not only in business but also in other critical areas like the healthcare and energy sectors. Increasingly complex software and infrastructure that heavily rely on third-party dependencies made IT systems fragile and at risk of attacks. The NotPetya ransomware and the Solarwinds hack are just a few recent examples of the impact of cyber attacks caused by, often hidden, dependencies in complex IT systems. Thus, the need for secure systems and software to properly handle sensitive data.

1.1 Problem Statement

A wide range of security mechanisms is advertised and discussed. From endpoint privacy browser extensions and antivirus to protect users on the Web, to security protocols like DNSSEC and IPsec, passing through Security Operation Centers and machine learning algorithms to detect malicious activities. Individuals and organizations are expected to invest more in security and are blamed when they fail to do so [172, 213, 324]. However, individual users and organizations can hardly measure if and how much these security mitigations are effective [360]. The ability to measure the efficiency of security mitigation strategies for risk assessment and response purposes is currently a vital requirement to avoid wasting resources on ineffective and costly solutions that do not provide any reduction of risk. Unfortunately, although the amount of security data produced and available for collection spiked in the recent years [234, 280, 345], the same cannot be said for the methodologies to analyze this data, extract key security principles, produce from them measures of risk, and evaluate the effectiveness of security mitigations. This is also due to the characteristics of this data, which vary in the security context in which a mitigation strategy is evaluated, thus requiring ad hoc collection, data models, and procedures to extract information. Security data are heterogeneous, from abuse and blacklist feeds where researchers have full visibility and data are nicely structured and accessible, to underground forums and threat intelligence reports where researchers have limited visibility and data are highly unstructured. These characteristics also affect the automated collection and application in each scenario, for example, data feeds on software vulnerability can be automatically collected via API in machine-readable formats while criminal discussions in underground forums require extensive preprocessing and analysis to obtain actionable insights.

Table 1.1 summarizes the characteristics of the information security data covered in this thesis to evaluate the effectiveness of security mitigation strategies. These characteristics pose challenges to the collection, correlation, and analysis of data for a quantitative evaluation of security mitigation strategies. The result is that currently, the evaluation of risk and effectiveness of mitigations reduces to a qualitative assessment based on subjective opinions and interpretations and not reproducible methodologies [29, 92].

Current security mechanisms are deployed without knowledge of their effectiveness with the risk of ignoring more effective alternatives. We lack appropriate methodologies to quantitatively evaluate security mitigation strategies by employing the rich set of security data sources available.

1.2 Research Questions

Since data are heterogeneous and security mitigations can be deployed in a variety of different contexts, we cannot have a unique research question. Therefore, we instantiate it for each specific security scenario. Our goal is to show that, by relying on security data models for both attackers and defenders, an evaluation of security mitigations is possible in any situation, from the evaluation of mechanisms to protect the privacy of individual users when browsing their preferred e-commerce website to the analysis of security mitigation strategies for organizations facing nation-state groups. Indeed, our thesis takes different perspectives for the evaluation of security mitigations in different dimensions by considering both security-related and privacy-related scenarios. We furthermore, considered different types of control techniques [169]: corrective countermeasures (Chapters 3 and 4), in which we evaluated the reduction of the impact of a successful attack, detective countermeasures (Chapters 5 and 6), in which we evaluated discovery of attacks and threats, and preventive countermeasures (Chapter 7), in which we evaluated reduction of exposure.

Table 1.1: Security Data Sources Characteristics

● Full ◐ Partial ○ Minimal

Security Data Sources	Description	Structured	Visibility	Automated Collection	Actionable	Chapter
Abuse and Blacklist Feeds	Technical indicators of known threats like IP address, domains, file hashes.	●	●	●	●	Chapter 3
Software and Vulnerability Repositories	Technical information on sw vulnerability, affected software, versions, available updates.	●	●	◐	●	Chapter 7
Web Infrastructure	Infrastructure entities like ASes, NS, domains, CDNs. and their dependencies.	◐	◐	◐	●	Chapter 4
Web Third-party Dynamics	Third-party live interactions like execution of external JS code and access to trackers	◐	◐	◐	●	Chapters 3 and 4
Network Monitoring	Raw logs and security events of network activities	◐	◐	◐	◐	Chapter 5
Underground Forums	Discussions and trading of online illicit activities	○	○	◐	○	Chapter 6
Threat Intelligence Reports	Technical description of adversary capabilities and TTPs	○	○	○	○	Chapter 7

Web Privacy Scenario The prevalence of web tracking [196] is affecting users' privacy when browsing the Web. Large-scale studies analyzed trackers ecosystem and the effectiveness of tracker blockers [124, 218]. However, individual users tend to browse to few websites [53, 280] and thus lack a mechanism to evaluate the privacy risk and effectiveness of tracker blockers in their specific scenario. Our goal is to provide a quantitative evaluation of tracker blockers for an individual. We thus identify the following research question:

***RQ1*:** How can we provide a third-party independent verification of empirical tracking claims and the effectiveness of tracker blockers for individual users?

Internet infrastructure Scenario The Web is currently essential for business and entertainment tasks we perform daily. The modern Web operates thanks to a variety of different protocols and relationships at different levels of the Internet infrastructure. However, the complexity of the Web can be a vector for a number of attacks against Web users [299]. A rich set of mitigations have been proposed to secure the Web, from application-level protections like SRI to resolution and PKI security mechanisms like DNSSEC and Certificate Transparency. However, it is still unclear the effectiveness of each mitigation given the current status of the Web, with the result that the deployment of security mechanisms is faltering [81, 85, 194, 276]. Our goal is to provide a quantitative evaluation of the deployment of security mechanisms at the different level of the Internet infrastructure. We thus identify the following research question:

***RQ2*:** How can we evaluate the cost-effective selection of mitigations for securing the Web given the current Internet infrastructure?

Security Operation Center Scenario The 2022 Verizon Data Breach Investigation Report reported more than 900k intrusions [345]. To detect such attacks, IT infrastructures rely on Security Operation Centers (SOC) to monitor the alerts issued by IDS [84]. Current works focused on finding the true attacks and limiting false alerts [16, 130]. However, correctly identifying and investigating the few alerts referring to an attack is a complex task. Current metrics to evaluate the effectiveness of SOC overlooked this [294, 295] thus measuring only part of the performance of an analyst [180] that can lead to an incorrect allocation of security resources [320]. Our goal is to provide a quantitative evaluation of the accuracy of SOC analysts' incident reports depending on technical skills and technological changes in the SOC. We thus identify the following research questions:

***RQ3a*:** How to model the task complexity of investigating a cyber-attack?

***RQ3b*:** How task complexity helps in comparing changes in accuracy due to technical (skills of analysts) and technological differences (different filtering of alerts)?

Criminal activities in Underground Forums Scenario Forums are an important medium for users to exchange information and offer cybercriminal services. Several works employed machine learning classifiers to identify the type of posts [72, 259], the goods offered [15,342], and the business interactions [47,318]. The performance of these classifiers is based on the random selection of a subset of posts, without considering the forum social network relationships. Our goal is to provide a quantitative evaluation of ML classifiers performance trained to detect cyber criminal activities using different sampling methodologies that exploit social network information. We thus identify the following research questions:

***RQ4a*:** What are the changes in performance for a ML classifier using different centrality metrics to generate stratified training samples?

***RQ4b*:** What are the changes in performance using a different proportion compared to the population for the stratified training samples?

Advanced Persistent Threats Scenario Enterprises are taking up to seven months to update their machines with the most recent software version [184, 286]. This behavior seems irrational when dealing with highly skilled and organized attackers like Advanced Persistent Threats (APT) [83, 275]. However, there is hardly any way to determine if a better approach to software updates is necessary given that most of the studies on APTs are qualitative analyses of a handful of campaigns [83, 275, 338]. Our goal is to provide a quantitative evaluation of the effectiveness of software updates to protect against APT campaigns. We thus identify the following research questions:

***RQ5a*:** What are the APTs characteristics that quantitatively describe the landscape of APT campaigns as observable from public reports?

***RQ5b*:** Given a quantitative description of both APT campaigns and software updates, how effective are different update strategies to protect against APT campaigns?

1.3 Contributions

The main contribution of this Thesis is a quantitative evaluation of security mitigation strategies employing a heterogeneous set of information security data sources that differ in terms of visibility, structure, collection procedure, and analysis. We define data models to describe the underlying processes and relationships between threats and defense mechanisms, instantiate them on heterogeneous security data, and evaluate security mitigations with different threat models.

We show that quantitative measures of effectiveness and cost using security data are not only possible but also beneficial for users and companies to identify better security plans. To demonstrate that, in this Thesis, we present models and methodologies to collect, organize, and analyze security data to quantitatively evaluate the effectiveness of security mitigations.

But what are the benefits of developing data models to evaluate the effectiveness of security mitigations compared to a purely data-driven approach? With our approach, we make explicit causal relations between attacker behaviors, the characteristics of the network in which they act, and the security mitigations available for a defender. The development of security data models allows researchers to investigate what-if scenarios where otherwise part of the security data from attackers or security mitigations would be rare to collect, lack at all, or would require to re-run entirely the empirical analysis. But most importantly, with an explicit model of the underlying processes and relationships we provide, along with a quantitative result of the effectiveness of security mitigations, a transparent explanation of why certain mitigations succeed and others do not, why some are worth it, and if so under which conditions.

To answer to ***RQ1***, we present **a model of Web tracking practices** based on the client-side flow of data sharing exploited by Web trackers. We apply this model to the network configurations of the Internet to independently check Web tracking practices and evaluate tracking blocking tools. We showed the applicability of this approach for individuals based on the Top 100 Alexa domains.

To answer to ***RQ2***, we show **a graph-based analysis of Internet security mitigations** to protect billion of Web users. We considered the infrastructure supporting the Top 5k Alexa domains and determined that currently deployed security mechanisms are ineffective and the centralization of infrastructures to a few entities makes them critical for the security of the entire Internet. We identified mitigations that can increase the security of the entire Web at a relatively modest cost of deployment.

To answer to ***RQ3a*** and ***RQ3b***, we develop **a model to evaluate SOC analysts investigation of cyber-attacks** through network monitoring. We described the process and information required to analyze traces of the kill chain of an intrusion. We quantitatively measure the effect on the analyst performance of technical and technological changes in the investigation of real attacks.

To answer to ***RQ4a*** and ***RQ4b***, we present **a methodology to create stratified samples** using information about the centrality metrics of the network for machine learning classification of online criminal activities and to evaluate their performance. We apply our approach to a forum containing millions of posts on a variety of different topics to measure the effects of the sampling technique. We find that classifiers trained with similar samples can significantly disagree on the classification of criminal activities when deployed on the entire forum.

Finally, to answer to ***RQ5a*** and ***RQ5b***, we develop **a methodology to evaluate software update strategies** to protect against APT campaigns. We constructed a rich database containing more than 350 APT campaigns from 86 APTs over a period of more than 10 years. We observed that updating indiscriminately is not *the* solution and that considering only publicly known vulnerable releases has the same risk profile of always updating the software, but costs significantly less.

1.4 Thesis Structure

This Thesis is structured as follows:

Chapter 2 provides an overview of the sources of security data employed to evaluate the effectiveness of security mitigation strategies. This chapter is partially based on the related works, background, and motivation sections of the research papers.

Chapter 3 presents a model to describe Web tracking techniques and evaluate the effectiveness of tracker blocker tools. I focused on threat modeling, data collection, and evaluation of mitigations. Fabio Massacci focused on formal tractability. The chapter is partially published in:

[331] Giorgio Di Tizio, Fabio Massacci. "A Calculus of Tracking: Theory and Practice" *In Privacy Enhancing Technologies Symposium*, Sciendo, 2021.

Chapter 4 presents a quantitative evaluation of security mitigation strategies at the Internet layers to secure the Web. Patrick Speicher and I were jointly responsible for this work. I focused on the threat modeling and data collection part, while Patrick Speicher focused on the graph-based implementation. This chapter is partially published in:

[100] Giorgio Di Tizio, Patrick Speicher, Milivoj Simeonovski, Michael Backes, Ben Stock, Robert Künnemann. "Pareto-Optimal Defenses for the Web Infrastructure: Theory and Practice". *In ACM Transactions on Privacy and Security,* ACM, 2022.

Chapter 5 presents a model to compute the accuracy of tickets issued by SOC analysts during the investigation of cyber-attacks and implement a controlled experiment to measure the changes in performances due to technical and technological differences in a simulated SOC. I was the main author of this paper. I focused on developing the model, the data collection, and the analysis of the data. Leon Kersten helped in the data collection. Martin Rosso developed the SAIBERSOC tool employed in the experiments [273]. The chapter is submitted to:

[99] Giorgio Di Tizio, Leon Kersten, Martin Rosso, Luca Allodi, Fabio Massacci. "Measuring SOC Analysts Investigation of Cyber attacks using the Entropy of Task Complexity". *To be submitted to ACM CCS.*

Chapter 6 presents a methodology to generate stratified samples to train ML classifiers and evaluates the effectiveness of different stratifications on the performance of classifiers to detect criminal discussions in underground forums. I was the main author of this paper. Gilberto Atondo Siu helped in the manual annotation of posts. The chapter is submitted to:

> [98] Giorgio Di Tizio, Gilberto Atondo Siu, Alice Hutchings, Fabio Massacci. "A Graph-based Stratified Sampling Methodology for the Analysis of (Underground) Forums". *Submitted to IEEE Transactions on Information Forensics and Security.*

Chapter 7 presents an analysis of the Advanced Persistent Threats ecosystem and a methodology to evaluate the effectiveness of software update strategies on historical data about campaigns. I was the main author of this paper. I focused on the data collection and evaluation of software update strategies. Michele Armellini helped in the data collection. The chapter is partially published in:

> [97] Giorgio Di Tizio, Michele Armellini, Fabio Massacci. "Software Updates Strategies: a Quantitative Evaluation against Advanced Persistent Threats" *In IEEE Transaction on Software Engineering*, IEEE, 2022.

Finally, Chapter 8 reflects on the main contributions of this work and provides discussion on future work.

Chapter 2

Information Security Data: Collection and Models

To evaluate security mitigations we rely on information security data from a variety of heterogeneous sources. Information can refer to the characteristics and structure of the network on which we want to evaluate the security mitigations, for example, the Internet dependencies between websites and content delivery networks and the software vulnerabilities present on a machine, or to the characteristics and structure of a threat for a network, for example, the software vulnerabilities exploited by an APT group, the 3rd-party trackers included in a website, and the trades in dark web forums. As discussed in Chapter 1 information security data sources present different characteristics and not all data are created for being used to analyze the effectiveness of security mitigation strategies. Indeed, although some data sources like software vulnerability repositories are intended to be shared, parsed, and analyzed, others like dark web forums discussions, are intentionally resisting data collection and analysis. To extract meaningful measures of risk and effectiveness of security mitigations, we need to identify from these security data the underlying processes and relationships that allow the analysis to join together different sources, describe attackers' and defenders' behaviors, and interpret and measure their interactions. In this chapter, we present a list of data sources of security data employed in this thesis to evaluate the effectiveness and cost of security mitigation strategies. We present them in increasing order of complexity in terms of visibility and structure. We discuss their characteristics and the challenges faced when collecting and analyzing this data.

2.1 Fully Structured and Visible

Some security data sources like vulnerability and abuse feeds are well structured and immediately available in a static and clean way. Furthermore, researchers can access all the information and check that, e.g., a malicious domain is indeed hosting an exploit kit or a vulnerability is present in the latest version of Chrome.

2.1.1 Abuse and Blacklist Feeds

Technical indicators of abuse are used to identify the presence of malicious activities on a network, prevent connections to known threats related to phishing, malware host, and web tracking, or infer the security posture of network owners. Data are generated both by the sharing communities [60, 112] and by specialized companies [61, 136]. Abuse and blacklist feeds contain up-to-date information about known threats in the form of machine-readable lists of indicators IP addresses, domain names, URLs, file hashes, and signatures of exploits. Examples of such sources are AlienVault OTX, Symantec WINE [107], and Virustotal for security-related blacklist feeds and Disconnect, EasyList, and EasyPrivacy for privacy-related filter lists. Trackers and ads filter lists are loaded by browser extensions like Disconnect, AdBlock Plus, and Privacy Badger to prevent connections to trackers' domains while surfing the Web.

General issues for collection and analysis Abuse and blacklist feeds are often available via API or some other machine-readable format. Although useful for an automatic investigation to detect intrusions or to prevent known threats, technical indicators from blacklist feeds quickly become obsolete and suffer from poor accuracy and coverage [197, 284].

Raw data collection In the context of web tracking we collected the filter lists of Ghostery, Disconnect, and EasyList in 2016 from previous works [44, 45] and compared them with the filter lists of Disconnect, EasyList, EasyList&EasyPrivacy, and Privacy Badger in 2019. The Privacy Badger filter list was generated by visiting the Top 200 Alexa domains in December 2019. Due to the risk of fingerprinting, after 2020 the Privacy Badger filter list is not created individually for each user but it is the same for all users [126].

Data models construction From security data related to the known adverts and trackers, we characterize the presence and spread of web trackers on the Web and identify domains that are purely devoted to tracking from domains that behave differently depending if users intentionally interact with them or include them as third parties.

2.1.2 Software and Vulnerability Repositories

Compared to a few years ago, the amount of publicly reported software vulnerabilities skyrocketed. Several vulnerability repositories are available to identify and catalog publicly disclosed vulnerabilities. Famous databases are the MITRE CVE [222] and the National Vulnerability Database (NVD) [233] which cover both commercial and open-source software. Other databases like the Snyk vulnerability database [302] focus on vulnerabilities on open source projects only like Maven, pip, and Debian.

These databases provide a rich set of information: from the timeline of reporting and publishing the vulnerability, to the affected software products and versions, additional resources like vendor advisory and available patches, presence of proof of concept exploits, description of the weakness, and a 'technical assessment' of the vulnerability.

Vulnerabilities are uniquely identified by a CVE ID that is reserved when a vulnerability is reported and then published on the vulnerability database. Software vendors release security bulletins and advisories related to the presence of vulnerabilities in specific products and versions and the associated information to fix or update the software product. Examples are the Adobe Security Bulletin and the Microsoft Security Guide.

To successfully evaluate security mitigations, along with vulnerability databases, information about the release of software versions is critical. Each software vendor provides a list of release history for their products and links the affected software versions to a specific CVE when applicable. The Common Platform Enumeration (CPE) Dictionary is an attempt to provide a standardized and machine-readable format to describe software products, versions, and packages.

General issues for collection and analysis Although some of the vulnerability and software update databases like NVD and Microsoft Security Guide provide machine-readable data that can be easily parsed to extract key information, most of the vendor repositories are not intended for past versions indexing [226]. Another major challenge is that most of the information provided by a CVE entry like the technical assessment or the affected software versions requires expertise and manual effort. Thus, it is not uncommon to observe inconsistencies and errors in the list of product names, the CVSS score [361], the CVE publication date [31], and vulnerable versions [105].

Raw data collection We employed the NVD to automatically extract the release and published date for a set of CVEs collected from security reports (see Section 2.3.2 for more details) as well as the list of products and versions affected. We then integrated this information with the CPE Dictionary. From the CPE, we extracted the list of vulnerable versions based on the CPE Match Strings. For example, CVE-2016-4113 affects all the versions of *Flash Player* up to 21.0.0.213. The associated JSON NVD file does not provide the entire list of affected versions (including the updates) in the CVE description but a CPE URI of the form *cpe:2.3:a:adobe:flash_player:*:*:*:*:*:*:*:*", "versionEndIncluding": "21.0.0.213"*, we thus matched the CPE in the CPE dictionary to get the list of all prior versions affected.

We then manually collected updates and versions for 5 widely employed software products in the period of observation of the attacks (*Office*, *Acrobat Reader*, *Air*, *JRE*, and *Flash Player*) for the Windows O.S. with the associated release date.

Data models construction From security data related to software versions, vulnerabilities, and updates, we characterize the underlying attack surface and security posture of an organization in terms of installed software and strategies for software updates. This conceptualization is put in relation to the responsiveness of software vendors by characterizing the speed with which updates are released for different software and the interplay between releases of updates and publications of software vulnerability information.

2.2 Partially Structured and Visible

Other security data sources can be partially structured where data must be retrieved with interactions with systems following specific protocols and data can be dynamic, for example, Web interactions and network logs. Researchers have partial visibility as it is not always possible to have a complete and global view of the behavior of the infrastructure from which the data is collected. An example is the Web interactions where trackers or banners change depending on the location or exchange server-side information.

2.2.1 Web Infrastructure

The Internet is composed of a complex combination of services and protocols that rely on each other to provide users the ability to browse the Web for fun and profit. Knowledge about the dependencies and interactions required for the Web to properly work is employed to determine points of failure of the infrastructure, the impact of attacks on critical

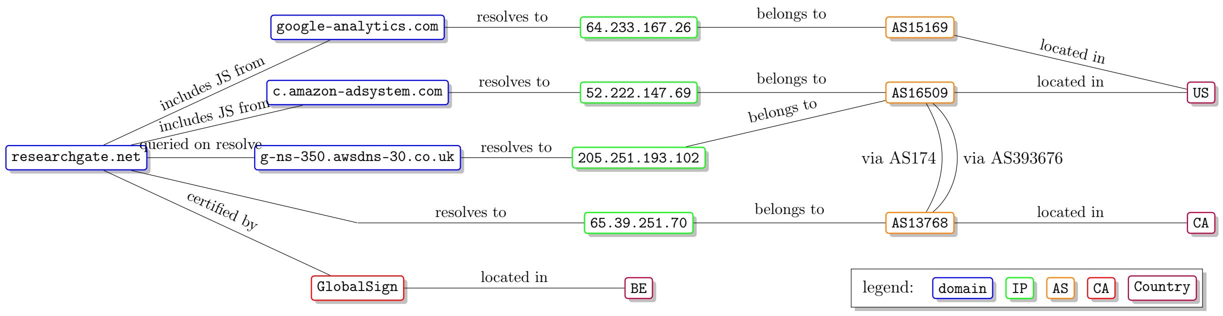


Figure 2.1: A fragment of the infrastructure relations while visiting `researchgate.net`

infrastructure and large organizations [299], and analyze the effect of the application of new security mechanisms [307].

To illustrate the most important infrastructure dependencies in the Web, consider the following example summarized in Figure 2.1. A user browses through a gallery on the popular image-hosting service `researchgate.net`. The browser first has to resolve this domain to an IP. This is done through a series of DNS queries performed by the resolver (usually first the user's local resolver, then its ISP) to contact the authoritative NS. To prevent forged or modified DNS responses, name servers can implement DNSSEC as a security mechanism. Name servers can also provide information to set up a secure connection using TLS through the DANE protocol. The correct resolution of `researchgate.net` depends on each of these NS. After resolution, the user connects to an IP, which belongs to an autonomous system (AS). These ASes are interconnected, and packets need to be routed via multiple ASes. On the network layer, the integrity of the packet transmission depends on each AS that is traversed. The website can now be delivered by the web server. To prevent a variety of attacks like Cross Site Scripting (XSS) or MITM attacks, the web server delivers a sequence of HTTP response headers like CSP and HSTS. In addition to its content, websites often include JS from external content delivery networks, in this case, `google-analytics.com`, which in turn depends on various name servers, on the AS the IP belongs to, etc. We further discuss the third-party dynamics in a website in Section 2.2.1. In case the website and the third-party content are retrieved via HTTPS, the authenticity of the connection depends on the signing CA and *all* root CAs whose certificates come with the user's browser, as any of these may supply the website's certificate.

General issues for collection and analysis A variety of platforms provide measurements and technical information about the Web infrastructure like OpenINTEL, HTTP Archive, and RIPE Atlas [103]. However, achieving a global and complete routing coverage, where all possible routes between ASes are covered is an infeasible task, if not impossible [147, 242]. BGP data is available to a limited extent, leaving a meaningful part of the AS-level topology hidden. Along with available datasets, ad hoc crawlers can be developed to probe name servers and websites. For most of this information, static crawlers are sufficient to collect information about the IPs, NSs, CAs, ASes, and HTTP headers. It has been shown that a limited number of large organizations have a key role in making the Web available and secure [174, 299]. One main challenge is to associate different infrastructures like domains, NS, and AS to a single entity due to the presence of aliases that require heuristics and manual investigation [174].

Raw data collection We considered the Top 5k Alexa domains obtained from Tranco [258] on 1 Oct 2020. The data collected represents a snapshot of the status of the Internet at a specific moment. Out of the 5k domains, 4608 were accessible (92%) at the time of the crawl. The remaining domains either provide services not related to web browsing or were down. We crawled each accessible domain to collect the web server configuration, its CDNs, DNS data, routing data, geolocation, and (if applicable) CA information. The data collection was performed from a single location at a European university.

We collected the `strict-transport-security` and `Content-Security-Policy` security headers to determine the presence of the HSTS and CSP mitigation respectively. In the case of CSP, we parse for the presence of the *upgrade-insecure-requests* directive. We ignored the remaining policies. It is important to underline that we assumed a correct implementation of the security headers and we did not parse them to determine misconfigured policies. We discuss in Section 2.4.1 security data related to misconfigured security mechanisms. We then probe the server with HTTP requests on the standard port 80 to collect the sequence and type of redirections to an HTTPS connection.

For each website and CDN, we collect the list of authoritative NSs that are contacted during iterative DNS resolution. We then queried for DNSSEC and DANE records. For DANE, we requested the TLSA record for the website on port 443. For DNSSEC we required the `DNSKEY` records for the zone, and we then trace the presence of the `DS` and its `RRSIG` records in the parent zones up to the root zone. Although DNSSEC is prone to misconfiguration [85] (e.g. expired signatures), we did not investigate these issues and we discuss them in Section 2.4.1.

To model the internet connectivity, *i.e.,* connectivity between ASes, we collect traceroutes provided by RIPE Atlas [268] for the set of autonomous systems considered in our dataset. We observed traceroutes that have the domains from our dataset as their destination. For each ASN, we then retrieved the holder's name. We assume that BGP routes are reasonably stable [265].

For each NS, website, and CDN, we include their geolocation in our dataset. We link IPs to ASes using the RIPEstat database service [269] and we map ASes to countries using the MaxMind database [215]. In addition, each CA is mapped to a specific country using the information stored in the issuer section of the digital certificate.

For websites that support HTTPS, we use the X.509 certificate to identify the issuing certification authority. To that end, we combine the information stored in *Certificate Fields* that are reserved for the issuer of the certificate: Common Name (CN), Organization (O) and Organizational Unit (OU). Finally, to identify the country, *i.e.,* the administrative entity of CA, we collect the Country (C) field.

We then collected statistics about the number of visits to each domain using Alexa's `UrlInfo` API.

Data models construction From security data related to the Internet infrastructure, we characterized the underlying processes and relationships between entities on the Web that interact when users request content on the Internet. We describe routing interactions among ASes, dependencies between domains and NS, domains and CA, as well as between domain and content delivery networks. Based on these dependencies we characterized attack vectors carried on different levels of the Internet infrastructure, and where security mitigations like IPsec, DANE, DNSSEC, CT, HTTPS, HSTS, CSP, and SRI apply to prevent attacks.

2.2.2 Web Third-party Dynamics

Online services heavily rely on third parties to provide ads, load external JavaScript code, embed social media, and collect analytics. Third parties dependencies can create problems in terms of security and privacy. For example, third-parties can be exploited to perform malvertising, drive-by download, cryptojacking, and web tracking [231,335]. Third parties are embedded using JavaScript or iframe in a web page. Although some of these inclusions are observable in the source code of the web page, others are created dynamically and can independently load further resources at run time [165,335]. Web tracking heavily relies on the third-party dynamics between websites to track users online via stateful, e.g. cookies, and stateless techniques, e.g. browser fingerprinting, or a combination of the two [125]. Requests to third parties can hide the exchange of identifiers to perform cookie sharing among different trackers [284]. Although user tracking is regulated in the EU by the General Data Protection Regulation (GDPR) and ePrivacy Directive, it is not uncommon to observe violations [58, 214]. Collection of third-party dynamics on the Web is obtained by directly interacting with the websites and the resources loaded during the visits with crawlers that register a variety of information like URLs and domains contacted, sequence of HTTP redirections, JavaScript loaded, data stored on the browser, etc.

General issues for collection and analysis Measurement of third-party dynamics starts from a list of top websites like Tranco [258] that are accessed using crawlers. Different implementations of crawlers are available depending on the goals of the analysis. Stateful crawler better emulates human behaviors by storing cookies and other persistent data and the results observed are dependent on the previous browsing history. While stateless crawlers provide results as a new user for each interacted website. A widely used tool to dynamically crawl websites is OpenWPM [116].

Third parties and dependencies on a website can vary from the landing page and its internal pages. Thus, crawlers must dynamically interact with different pages of the website [34, 335] and implement bot mitigation techniques like random scrolling. Furthermore, the identification of the third parties included in a website is challenging. The dependencies are highly dynamic and present complex chains of intermediaries [284, 335] that change the included resources over time [312]. Location and configurations of crawlers can influence the collection of website behaviors [277] in particular for what concerns web tracking practices [91]. Identification of owner organization behind included third-party resources is a challenging task due to the complex relationships between companies [284] and often requires a manual effort to avoid false positive [312].

Raw data collection To explore the security risk related to the inclusion of third-party JavaScript code we obtained as a starting point the Top 5k Alexa domains from Tranco [258] on the 1st October 2020. For each domain, we extract the external JS resources that are loaded either statically or dynamically, analyze the protocol and check for the presence of the subresource integrity using a crawler developed with the *Selenium Web* driver, an object-oriented API for web-app testing that automates browsing of websites from the command line. As the content of the landing page and its internal pages likely differ [34], to avoid the risk of capturing a limited subset of third-parties [336], we further visited up to 25 random internal pages obtained from the links on the landing page of the visited domain. We employed the *tldextract* package [261] to extract the TLD+1 of each resource. We consider a resource an internal page that belongs to the domain visited if it shares the TLD+1 with the landing page of the domain but has a distinct URL by excluding the fragment component. For example, from the landing page of the domain *foo.com*, the URL *foo.com/#home* is not considered an internal page while the URLs *foo.com/content/index.html* and *bar.foo.com/index.html* are visited as internal pages.

To collect web tracking behaviors and dependencies we employed the 10k Site ID Detection 2016 and 2019 datasets[1] collected using a stateful instance of OpenWPM. The dataset contains the interactions during the subsequent visits of the 10k domains. In particular, it contains the list of URLs loaded, the HTTP responses, and the HTTP status code. To compute the sequence of redirections exploited to track users among websites and share cookies, we first extract the sequence of HTTP responses for each visited domain, then order the responses using the timestamp to avoid considering intermediate redirections as the beginning of a new connection and extract the redirections from the set of HTTP responses. Cookie syncing can be detected by analyzing URLs [124] and payloads in POST requests. For illustrative purposes, we use here 200 empirically validated domain pairs performing cookies syncing from [44].

Data models construction From security data related to the dynamic third-party interactions with websites, we characterize the underlying processes of data sharing performed by trackers employing HTTP redirects and exchange of cookies, like cookie syncing, as well as the relationships between websites and CDNs on which domains depends on via JS inclusions. Based on these interactions, we characterize attack vectors carried by trackers and compromised CDNs to target the privacy and security of users on the Web.

2.2.3 Network Monitoring

Network monitoring allows IT services and SOC operators to identify phases of an attack, from reconnaissance to exfiltration. These data sources can be either raw logs or security events generated by security devices like Intrusion Detection Systems (IDS). This data can be used to generate new detection signatures, collect artifacts like malware, define the TTPs of the adversary, attribute the intrusion to a specific group, and remediate effectively the attack. Network monitoring can be performed with different levels of details from aggregate statistics related to the number of connections and packet sizes to full packet collection [206, 340].

Network activities are generated by a variety of sources like routers, firewalls, DNS servers, and IDS sensors like Suricata and Snort. The logs are often matched against signatures of anomalous and malicious behaviors. It is not uncommon for signatures to produce a high amount of false positives, i.e. alarms generated without an actual security-

[1] `https://webtransparency.cs.princeton.edu/webcensus/`

related event. This is because signatures are written in a way to cover a large spectrum of malicious behaviors [16]. Thus, benign applications can also trigger the event. For example, an unusual amount of requests towards a web server can be produced by a DDoS attack or by a high amount of traffic due to a sale offer. A match will trigger an event that will require an intervention to determine its validity.

Modern IT infrastructures are monitored by Security Operations Centers where these sources are analyzed by a mix of automated rules and human analysts to filter true alerts from false positives. The security operators triage alerts, logs, and events from the network and its endpoints to determine if a malicious activity is occurring. Given the high amount of data collected by the different sensors, SOCs may use a hierarchy where higher tiers analysts treat a subset of escalated incidents but analyze each incident more in-depth. Tier 1 analysts compose the largest group with generally less experienced analysts compared to other tiers. This group focuses on real-time monitoring of security incidents, triaging through a large number of alerts and logs per day, and does not spend much time investigating each alert. Tier 1 analysts issue tickets related to incidents containing information like IP/machine affected, malicious IP contacted, and attack vectors employed to Tier 2. The Tier 2 analysts are often more experienced analysts who analyze each incident in greater depth and make decisions on appropriate next steps to take, if any (e.g. ignore, report to customers, engage attack containment strategy, etc.) [146, 180].

General issues for collection and analysis Information security data related to network activities are automatically collected in log files by the O.S., the applications, and IDS. The logs and events have several different formats that vary based on the sensor [106]. Security Information and Event Management (SIEM) tools can help to automatically parse, normalize, and combine different logs sources for inspection. However, humans are still critical to perform in-depth analysis [16].

Raw data collection We employed the SAIBERSOC tool [273] to reproduce the network traffic of attacks in a network and forward it to the Network Intrusion Detection sensors to generate alerts. We collected the IDS alerts and logs for two attack scenarios: the first reproduces the infection by the Mirai botnet, and the second is the exploitation of a Remote Command Execution on an EXIM SMTP server. The IDS alerts contain information on the IP address of the machines, the source ports contacted, the protocol used, the alert body information like `"ET TROJAN Mirai Botnet Domain Observed"` or `"ET EXPLOIT bin bash base64 encoded Remote Code Execution"`, the Suricata rule triggered, the number of time the rule is triggered, and the full packet captured by the

IDS. The logs contain information on the connections to a machine and the DNS queries.

We further collected the tickets issued by the Tier 1 analysts during the investigation. The tickets contain the following information for an attack: victim IP, attacker IP, reconnaissance technique, the type of vulnerability exploited, server contacted to download malware, C&C machine contacted for data exfiltration.

Data models construction From security data related to network activities, we characterize the underlying procedure of cyber-attacks through the entire kill chain: from the reconnaissance to the initial compromise and exfiltration. We identified the relations between each phase and the data available from IDS in a SOC platform and how this information is processed to identify the attack.

2.3 Unstructured and Not Visible

Finally, security data from underground forums and threat intelligence reports are based on natural language and thus mostly unstructured. Furthermore, they provide limited visibility for researchers. Underground forums are intentionally resisting collection and analysis, while for threat intelligence reports researchers must rely on the (already limited) visibility of security companies that claims a certain criminal group exploited a specific CVE or used a certain attack vector as there is hardly any way to have access to the telemetry.

2.3.1 Underground Forums

Analysis of underground forums provides timely information about adversaries' capabilities like exploits and malware, preferences in targeted software, and their criminal infrastructure and business models. The major challenge is that only a minor part of the discussions in these media refers to illicit activities [250].

Web forums are organized in boards that identify general categories of discussion like "Beginner Hacking", "Ebook Bazaar", "Remote Administration Tools", etc. In each board, members can start a new discussion, called thread, by writing an initial post. Some boards are dedicated to networking and information sharing while others are dedicated to the commerce of illicit products called marketplaces. Some web forums e.g. `Hack Forums` are covering a broad set of illicit activities, while other forums are specialized to certain topics like exploit kits and malware (e.g. `Darkode`), or blackhat search engine optimization techniques (e.g. `Blackhat World`) [259]. Web forums also differ by how their content is accessible, where exist both open and closed forums. The latter requires to be admitted by the existing members of the community or pay a fee [205].

Web forums discussions contain information about the type of cybercrime and goods traded in marketplaces, their evolution, prices, and characteristics that make them successful in the market [305, 341, 351]. Forums are analyzed to characterize the traded exploits and malware to predict the likelihood of exploitation [17, 20]. The exploits and associated CVEs discussed in posts can be used to identify preferences in targeted software or prioritize updates, malware artifacts can be obtained to generate new signatures, and the rates in which new exploits are introduced in the market can be analyzed to determine changes in strategies by the adversaries.

The underground economy quickly evolved into a commoditization of cybercrime, where criminals can buy specialized technical services for their illicit activities. These models lower the entry barriers for criminals that do not require any more in-depth technical knowledge to perform cyber crime. For example, to infect web users, a criminal can buy a RAT (Malware as a service), set up an Exploit Kit with a set of web browsers or plugins exploits (Exploits as a service), and obtain victim traffic to reach the Exploit Kit (Traffic Redirection as a Service) without knowledge of how to develop exploits or write malware. Forums are observed to identify the emergence of such cybercrime-as-a-service (CaaS) models and the criminals' demands [15, 76, 318, 342].

The social media structure of the web forums itself is useful to determine key forum members and the information flow in the network to detect new trends and define interventions by law enforcement agencies to disrupt the markets [250, 256].

General issues for collection and analysis Underground forums require significant effort for the initial collection. Infiltration into closed web forums can require paying a fee or obtaining an invitation from other members. The latter often requires establishing credibility in the web forum by actively participating in the criminal trades and discussions [17]. Once gained access, automated crawlers can scrape the forums to collect posts and members' information [252]. This requires ad-hoc crawlers to bypass anti-crawling techniques [76]. Given the unstructured nature of web forums and social media and the large volume of posts, ML and Natural Language Programming (NLP) techniques are employed to categorize and identify relevant data. This approach can employ unsupervised methods to explore the forum and cluster discussions, or supervised methods to classify posts. The latter requires manual annotation of posts to generate training samples for the classifiers. This procedure is laborious and time-consuming and requires researchers to identify and annotate a sample that allows the classifier to learn the key characteristics of the criminal topics. Another major challenge to both automated and manual inspections of web forums is the use of dark jargon to hide criminal discussions into innocent-looking posts [365].

Raw data collection We relied on the CrimeBB dataset [252], a database of underground forums available under request for research. We focused on `Hack Forums`, which is one of the largest and long-lived underground forums in the English language, famous among the others for the release of the Mirai botnet source code. We considered all posts and members that are classified as 'Offer','Request','Exchange', and 'Tutorial' by [72] and posted before June 2018 to compare with related works. The dataset comprises ≈11.3M posts from ≈447k members in ≈3M threads.

Data models construction From security data related to underground forum discussions, we characterized the underlying relationships and interactions between users and online criminal activities like DDoS, exploitation of software vulnerabilities, malware development, trading credentials, and spam using social network analysis and network metrics like degree and eigenvector centrality.

2.3.2 Threat Intelligence Reports

Threat intelligence reports aim to identify and inform about all relevant threats and provide context at each level of the organization [61]. A recent SANS report showed that more than 82% of the CTI teams leverage this type of resource [287].

Threat intelligence reports can be employed for network detection, to enrich local threat intelligence resources from the enterprise network (Section 2.2.3) to improve the situation awareness on the threat landscape modus operandi, to inform business decisions, to develop end-user awareness programs, and to perform threat hunting [61, 287].

Threat intelligence reports can take the form of a blog post or a bulletin on a security vendor website, an extensive technical report, or a conference presentation. The report can be public or private as part of threat intelligence feeds subscriptions. The difference between open source reports and paid ones stands in the number of details and indicators provided by the vendors [61]. Reports can differ in their focus. Reports can discuss a specific campaign targeting a software vulnerability, the results of reverse-engineering malware, or longitudinal analysis of the modus operandi of threat actors [61].

The information available ranges from a list of IOCs like IP addresses and domains, file hashes, modified registry keys, Powershell commands, etc. observed on a victim machine to a more high-level description of the TTPs exploited by a certain criminal group to perform the initial compromise, the elevation of privileges, and the exfiltration, the software vulnerabilities exploited, the attribution to a specific group or government organization, and the targeted sectors. In contrast to blacklist feeds that also cover IOCs in a structured way, technical reports provide additional context information and comprehensive description of the attacks with causal and temporal relations among artifacts [161,199,289]

To improve threat intelligence sharing of the rich set of information available in threat intelligence reports, several languages and formats have been proposed. Of particular interest is the MITRE Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) [5]. MITRE ATT&CK is a knowledge base of adversary tactics and techniques observed to be employed by adversaries. It provides a standard high-level description of techniques employed to achieve the different phases of an attack. For example, to achieve persistence on a machine, adversaries can employ different techniques like `boot or logon initialization scripts (T1037)`, `create or modify system process (T1543)` or exploit `Browser Extensions (T1176)`.

General issues for collection and analysis Threat intelligence information is obtained from the telemetry and incident response investigation of individual security companies, thus reports often lack validation, can be subject to marketing and geopolitical agendas [195], and are based on non-overlapping information about APT campaigns [61, 197]. Furthermore, different naming schemes are employed by different companies thus making it complicated to merge information from distinct sources. For example, Mandiant, CrowdStrike, F-Secure, and Microsoft refer to the same group with four different names: APT29, Cozy Bear, The Dukes, and YTTRIUM respectively.

Information about attacks is described informally in natural language thus making challenging to recover key information like IOC, TTPs, CVEs exploited automatically. Automatic extraction of information from technical reports using keywords [189] or Natural Language Programming (NLP) techniques lack accuracy and coverage [199]. The challenge of analyzing these reports is to discern if the meaning is referring to the present (the actual campaign the report is discussing) or the past (additional information that creates the context). Thus, a manual inspection of reports, although time- and labor-consuming, is necessary to extract accurate information like CVEs, attack vectors employed, and the date of the campaigns.

Raw data collection We considered APT groups that launched at least one campaign from 2008 to 2020 (excluded) and for which a precise date for the campaign is present in at least one report.

We manually collected data about APT campaigns from more than 500 technical reports and blogs. The final database contains information about 86 APT groups. For the excluded APTs, we either did not find information for their campaigns, or the date of their campaign was not known. For example, the Kaspersky article on Equation Group [175] provides a list of CVEs but does not provide information about the campaign when they were exploited. We started from the MITRE ATT&CK APT groups list. One researcher first collected the reports associated with each APT group from the Threat Actor Encyclopedia [328], which relies on sources like Malpedia, MISP, AlienVault, and MITRE. Then the researcher extended this set of resources by searching on the Internet for reports using as keywords the APT name as stated in MITRE (e.g. Stealth Falcon) and the term "CVE" until data saturation was reached, i.e. new reports do not add new information to the APT campaigns. The reports are obtained from cyber-security companies like Kaspersky, FireEye, Palo Alto Networks, and Google Project Zero as well as from technical forums and blogs.

Two researchers independently analyzed the content of each report manually to identify the following information for a campaign: the *date* when the campaign is first observed, the *CVE(s)* exploited, and the *attack vector(s)* employed.

We uniquely identify a campaign using the *date* in which it is first observed. If a campaign employs different attack vectors and/or different CVEs, we create multiple entries in the form $<APT\_name, attack\_vector, date>$ or $<APT\_name, CVE, date>$. Each entry is linked to one or more reports containing this information.

Data models construction From security data related to APT campaigns, we characterize the underlying behavior of APT when targeting their victim in terms of preference of attack vectors, type of software vulnerabilities (e.g. 0-day vs public vulnerabilities) and products, as well as their timeline and speed in vulnerability exploitation.

2.4 Other possible sources of security data

Other sources of security data that we did not employed in this thesis are available to evaluate effectivess of security mitigation strategy. For completeness we describe the most common ones.

2.4.1 Internet Misconfigurations

Information security data about an infrastructure includes mismanagement and misconfigurations of networks that describe the security posture of an organization. Misconfiguration can not only be exploited to compromise the owner of the misconfigured machine but also to perform attacks on other users and organizations on the Internet. For example, open DNS resolvers of an enterprise can be exploited to perform DDoS amplification attacks against target hosts. Misconfigurations can not only affect network infrastructures like DNS servers, routers, and web servers [366], but also the security mechanisms like CSP and HTTPS making them ineffective [355].

Misconfigured Infrastructures At the routing level, BGP configuration errors can produce short-lived route announcements that incorrectly modify the routing of packets among Autonomous Systems (AS). For example, a misconfiguration forced Meta DNS servers to disable BGP advertisements and took its network down [87].

Open SMTP servers that lack filtering of sources can be exploited to perform spam campaigns on the Internet. Several UDP network protocol allows amplification if machines are not properly configured among which NTP and DNS [274]. For example, open DNS resolvers can be exploited to perform a DoS amplification attack by requesting ANY queries with a spoofed IP [204, 339].

Certification Authorities (CAs) are the fundamental block of the Web PKI on which security mechanisms like HTTPS relies on. However, over the years CAs, intentionally or unintentionally, miss-issued certificates [191]. The majority of the errors are produced by mid and small-sized authorities that fail to populate fields, encode incorrectly the data, or insert invalid DNS names [185].

Misconfigured Security Mechanisms To protect against DNS poisoning attacks, DNS servers implement source port and query ID randomization. However, misconfigurations in the NS configuration can disable source port randomization or implement a predictable ID generation algorithm [204]. DNSSEC is a hierarchical public key infrastructure (PKI) that prevents forged or modified DNS responses. However, it is not uncommon to configure incorrectly this security mechanism making it ineffective or weak. Common errors include missing DNSSEC records like DS or RRSIG that make it impossible to validate the signature, presence of expired signatures, and use of weak keys [85]. Misconfiguration in DNSSEC can also propagate to other security mechanisms that rely on it, for example, the DNS-based Authentication of Named Entities (DANE). DANE is a mechanism to replace Certification Authorities (CAs) for the establishment of TLS connections and it is currently deployed for SMTP services encryption. Unsuccessful deployment of DNSSEC and failure in changing TLSA records due to certificate changes make DANE validations fail [193].

To protect against email spoofing attacks, email services can implement a set of security mechanisms among which DomainKeys Identified Mail (DKIM). DKIM relies on digital signatures to verify the integrity of the email content. Similarly to the previous security mechanisms incorrect or missing DKMIN records shared and weak keys allow adversaries to bypass the protection and send spoofed emails [354].

Client-side security mechanisms to prevent a variety of threats like Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), click-jacking, and MITM attacks are enabled via HTTP response security headers by the web server. For example, accessibility of cookies is limited by the presence of attributes like *HttpOnly*, *Secure*, and *SameSite*, the X-Frame-Option header defend against click-jacking, the HTTP Strict Transport Security (HSTS) header forces to upgrade to HTTPS connection, and the Content Security Policy (CSP) mitigate a variety of threats among which XSS. However, the adoption and implementation of these mechanisms are poor and can lead to the bypass of the protection. It is not uncommon that cookie protection [75, 277], click-jacking protection [74], and CSP are inconsistent and misconfigured [276, 355].

General issues for collection and analysis Several tools like ZMap, Masscan, Shodan exist to perform Internet-wide scanning to probe machines for misconfiguration. In other cases, interactive crawlers that mimic human behaviors are developed for an analysis of a web site security posture. Collection can be influenced by the location [353] and the configurations [277] of the crawler.

2.4.2 Host Monitoring

Data related to cyber attacks can be obtained from the enterprise machines or from (a network of) isolated machines, called honeypots, specifically designed to collect intelligence from intrusions and divert the attention of the attackers from the real assets of a network. Depending on the level of details to collect honeypots can just simulate the presence of a service and reply to probes (low-interaction) or implement a fully functional machine with (vulnerable) services (high-interaction) [77].

In contrast to network monitoring, host monitoring focuses on logs generated by the operating system (O.S.) and its application (e.g. the web browser). Host-based intrusion detection (HIDS) and Endpoint Detection and Response (EDR) systems analyze system calls traces to determine malicious activities on a machine [202] and keep track of file accesses, created processes, registry changes, and other system logs to identify patterns of elevation of privileges, lateral movements, or malware execution [173]. To detect slow and persistent attacks performed by Advanced Persistent Threats, data provenance graphs are generated to include causal relationships between system calls [25, 150, 153].

General issues for collection and analysis Host monitoring automatically generates high volumes of logs that are hard to retain for more than a few days and it is still subject to a high number of false positives [153]. Furthermore, each O.S. (Windows, Mac O.S., Linux) and application (Web server, database, etc.) produce its own logs with their format and specific locations.

2.4.3 Social Media

Social media like Facebook, LinkedIn, and Twitter are critical platforms for communication. However, they are also a vehicle of a variety for malicious activities such as phishing, malware delivery, spam, harassment, misinformation, and disinformation. Social media were employed to perform both mass [148, 282] and targeted attacks [18] against their users. Social media honey accounts can be set up to investigate the attackers' behavior in stolen social accounts [244]

However, social media provide also a platform for the timely sharing of information about criminal activities by cyber-security experts from newly detected malware, to ongoing DDoS and data breaches. Twitter discussions can be employed to determine the exploitability of software vulnerabilities [317], the presence of real-world exploits [281], or to predict when a CVE will be exploited [82].

General issues for collection and analysis Social media platforms provide API for data extraction. Although the access is much more standardized compared to underground forums, it is sometimes limited by the platform itself. For example, Twitter API allows one to retrieve only 1% sample of all the data matching certain parameters. The collection and processing of social media content present similar challenges of the underground forums (Section 2.3.1). The large volume of unstructured posts must be analyzed with ML and NLP techniques by means of supervised or unsupervised methods.

2.5 Integration of security data sources

Depending on the context, the integration of different security data sources can be used to analyze and evaluate security mitigation strategies.

In Chapter 3 we integrated information from blacklists of known trackers (Section 2.1.1) with third-party dynamics on the Web (Section 2.2.2) to evaluate the effectiveness of tracker blocking tools. In Chapter 4 we integrated information about the Internet infrastructure (Section 2.2.1) with information about the third-party JS dynamics (Section 2.2.2) to evaluate the effectiveness of mitigations for web users. While in Chapter 7 we integrated information about software vulnerabilities from NVD and software releases from vendors (Section 2.1.2) with threat intelligence reports (Section 2.3.2) to evaluate effectiveness and cost of software updates.

Other examples of integration are already widely employed in real-world settings where network monitoring (Section 2.2.3) is combined with system-level monitoring (Section 2.4.2) and enriched with external data like threat intelligence reports (Section 2.3.2), social media (Section 2.4.3), abuse and blacklist feeds (Section 2.1.1), and software and vulnerabilities repositories (Section 2.1.2) and correlated by SIEM tools to provide an in-depth monitoring of the network [16].

However, several challenges exist when integrating different security data sources. One challenge is to join different security data representations and formats in a common structure to perform the evaluation. This requires a data modeling process where the causal relations between security data are clearly specified. Another challenge is the temporal frame of the security sources, joining together sources from different time frames can produce incorrect and biased results. Consider for example the integration of network monitoring of DHCP logs that assign internal IP in a network with system-level logs. If data refers to different points in time there is the risk to associate a suspicious IP with the incorrect machine in the network. Similar considerations when employing IP blacklists from abuse feeds and network monitoring due to IP address reuse [262]. Combination of data from different sources among which forums, malware blogs, and abuse feeds to train classifiers to detect malware [38] can be subject to sampling and temporal biases where classifiers learn to distinguish malicious applications based on spurious features [37, 255].

Chapter 3

Web Tracking: Representation and Reasoning about Mitigations

Online tracking techniques and the interactions among trackers have received increasing attention in the last few years. In this chapter, we employ security data describing the Web third-party dynamics and information from filter lists of known trackers to evaluate the effectiveness of different privacy mitigations and determine if websites should comply with privacy regulations. We propose a novel formal model that describes the foundations on which the client-side process of data sharing behaves in terms of the network dynamics on the Web (included CDNs, shared cookies, etc.). Any formal derivation in the model corresponds to an actual tracking practice that can be implemented given the security data available. With this approach, we explicit causal relations between network activities on websites and tracking practices and explain why a certain tracking tool is better than another for individual cases. We apply our model to a dataset obtained from OpenWPM to evaluate the effectiveness of tracking mitigations up to Alexa Top 100.

3.1 Introduction

Online tracking of users for targeted advertising is the reality of today's Internet and the extent of such tracking has been the subject of an intense research activity [196]. Research studies span from facts-finding studies (e.g. [45, 137]) to technical analysis both for classical techniques (e.g. based on cookies syncing [248]) and novel techniques (e.g. based on browser fingerprinting [114]). Economic analyses are also not uncommon (e.g. [212]). Several mitigation tools also emerged to (partly) block trackers (e.g. Ghostery, Disconnect, Adblock Plus, etc.) and researchers have also investigated whether they are effective (e.g. [218, 272]) and their side-effects (e.g. [232, 343]).

These robust research activities, which we sample in Table 3.1, generated a number of open databases that provide internet snapshots[1] and that can be used to experiment with tracking behavior, the effectiveness of tracking mitigations as well as derive metrics for a tracker market share or trackers concentrations [52], at least for what is measurable from the Internet.[2]

The major privacy concern of the users is with whom and for which purposes their personal information is shared and not by which technology [151, 243]. In other words, users are not worried that an entity collects data when interacting with it but that these data are shared with other entities. How can we provide a third-party independent verification of empirical tracking claims? A study might claim that tool A is more effective than tool B at mitigating trackers but there is hardly any way for a third party to check why and how unless one re-runs the entire study and traces all results. Even the claim that a tracker *burstnet.com* can potentially know whoever visited website *amazon.com* is hard to check unless one re-runs the entire study. We thus identify the following research question:

***RQ1*:** How can we provide a third-party independent verification of empirical tracking claims and the effectiveness of tracker blockers for individual users?

We answer such question by a formally grounded mechanism: *a calculus of tracking for the internet*. We associate a formal relationship between various ways to exchange data (access and inclusion of web pages, cookie syncing, redirects, etc.) as they are measurable from the internet (and available in open datasets such as Web Census) and identify formal rules that capture how internet visits can be tracked.

We can formally prove that a tracker can potentially know that a user visited a website. The formal model allows one to determine if a website should comply with privacy laws (e.g. COPPA) or to compare different mitigations and conclude whether a mitigation is strictly better than another or at least quantitatively better along a Pareto frontier. In contrast to a pure data-driven approach, a formal model produces *a result that can be independently checked* [228] with a clear and transparent description on the causal relations that produce that outcome.

For this approach to be a useful link between theory and practice, such calculus must

[1]E.g. Web Census: `http://webtransparency.cs.princeton.edu/webcensus`

[2]Obviously, data exchange agreements between owners of seemingly different trackers do affect conclusions based on the internet. Detecting such agreements is hard given the present information asymmetry [11].

be tractable and the analysis can be performed on the fraction of the Internet visited by a user as available from open datasets (e.g. OpenWPM) and we do so up to Alexa Top 100.

The overview of the key works in this area (Section 3.8) summarized in Table 3.1 shows that the majority of the research focused on large-scale analysis and technological analysis, while no attempt has been done to formally describe the sharing procedures.

In this chapter we make the following contributions:

- We present a formal model that describes the passage of tracking information across websites that can be externally measurable (Sections 3.2 and 3.3)
- We formalize some of the interesting tracking relations that can be captured by our system (Section 3.4) and we discuss scalability issues and challenges (Section 3.5)
- We instantiate our model and compare the effectiveness of different mitigations (`Ghostery`, `Disconnect`, `Adblock Plus`, and `Privacy Badger`) on the Top 5, Top 10, Top 50, and Top 100 *visited* domains (Section 3.6).

Goals: we provide a framework that generates a third-party independent verification of tracking practices for *individual* cases, i.e. for single users that browse a limited number of websites. Large-scale studies over millions of domains are unrealistic and inappropriate to help users in determining the best countermeasure to enhance their privacy or determine which websites should comply with COPPA. Furthermore, these studies lack transparency and do not provide concrete evidence of the effectiveness of the mitigations analyzed (as well as provide in some cases contrasting results). Our framework fills the gap by providing *an explanation*, in the form of a formal proof, that can help users to evaluate the effectiveness of different adds-on or provide a proof that shows if a website should comply with COPPA.

Non-goals: we do not consider methods that rely on back-office data exchange for user identification. For example, collecting browser fingerprinting and using it in back-office data sharing. As such we could assert that certain websites perform fingerprinting but, in absence of a publicly known relation between these websites on how fingerprints (or other personal information) are shared, this would be a meager knowledge. Furthermore, the application of the framework is not intended for large-scale analysis of the whole Internet, where formal reasoning hardly scales.

3.2 A Formal Model for Tracking

We define a privacy threat when a website can know that a user visited other websites as a result of a process of data sharing. The major concern for a user is not that a website

Table 3.1: Web Privacy Topics addressed by the State of the Art

Research	Research Topic	[134]	[137]	[212]	[142]	[44]	[45]	[171]	[272]	[116]	[248]	[124]	[240]	[218]	[8]	[49]	[42]	[216]	[96]	Our work
Fact finding	Analysis of the tracking ecosystem				✓	✓	✓	✓							✓					✓
	Tracking coverage		✓		✓		✓		✓	✓	✓		✓	✓						✓
	Search context exposure to tracking				✓				✓											
	Detection of hidden flow among trackers					✓				✓	✓		✓							
	Detection of privacy regulation violation															✓				
	Detect duty compliance to privacy regul.																			✓
Economic analysis	Cookie syncing incentives	✓											✓							✓
	Revenue with and without cookies	✓	✓	✓									✓							
Technical analysis	Effectiveness of blocking techniques						✓		✓	✓		✓		✓						✓
	Development of a detection mechanism							✓			✓				✓	✓				
	Classification of Trackers							✓	✓			✓								
	Analysis of Tracking techniques									✓		✓			✓					
Logic	Formal expression of privacy policy																✓	✓	✓	
	Formal analysis of tracking																			✓

knows this is a recurring user, but that unrelated websites get knowledge of this activity thanks to the exchange of data. We did not model tracking techniques that are not been observed in the wild (e.g. [323]).

We represent entities like websites and trackers and their interactions with a set of predicates and rules. Each rule is composed of one or more preconditions required to activate a postcondition. Both preconditions and the postcondition are composed of predicates. The predicates composing the precondition are in conjunction. With this structure, it corresponds a formal representation using the Gentzen rules for the quantifier-free fragment of intuitionistic first-order logic (IFOL) we reported in Appendix A. We use IFOL since its proofs are constructive and thus there is a pairing between proofs and tracking.

3.2.1 Predicates

Table 3.2 contains predicates that capture network interactions among websites and users (in our analysis we reduced the websites to their PS[3] and PS+1 of the URL[4]) as well as the type of mitigations considered. $\mathit{IncludeContent}(w, w')$ indicates an inclusion of some content of web site w' in website w. The predicate $\mathit{IncludeContent}_{\mathit{cookie}}(w, w')$ describes, in addition, a transmission of cookies collected by w to w'. $\mathit{Redirect}_{\mathit{cookie}}(w, w')$ ($\mathit{Redirect}(w, w')$) indicates a HTTP redirection from the website w to the website w' with

[3]`https://publicsuffix.org/`

[4]For example, *https://s.ytimg.com* is reduced to *ytimg.com.*

Table 3.2: Ground Truth Network Interactions and Mitigations

The predicates are obtained from ground truth data (G) and are *not* derived from rules of the model.

$IncludeContent(w, w')$	G	website w includes 3rd-party content from the website w' (e.g. within an `i-frame` tag).
$IncludeContent_{cookie}(w, w')$	G	website w includes 3rd-party content from the website w' sending its cookies.
$Redirect(w, w')$	G	website w redirects visitors to the website w'. w *does not* append cookies in the redirection.
$Redirect_{cookie}(w, w')$	G	website w redirects visitors to the website w'. w appends its cookies in the URL (or payload).
$Visit(w)$	G	intentional access to website w by a user.
$Block\_request(w)$	G	extension blocks connections directed to the website w based on filter lists (e.g. `Disconnect`).
$Block\_tp\_cookie(w)$	G	3rd-party cookie blocking for website w.
$Kids(w)$	G	website w is directed to children under 13 y/o

(without) a transfer of cookies collected by w.

$Block\_request(w)$ indicates that the connection to the website w is blocked, for example by an add-on. $Block\_tp\_cookie(w)$ indicates that the website w is not allowed to set cookies. These two mitigations protect users using different techniques. If $Block\_request(w)$ is evaluated as true (e.g. `Disconnect` blocks the website w), then all requests to w are blocked. If $Block\_tp\_cookie(w)$ is evaluated as true (e.g. 3rd-party cookie blocking protection is active), it means that the browser does not allow w to set HTTP cookies, however, it does not block HTTP requests directed to w.

Table 3.3 summarizes the predicates that describe a possible exchange of users' information between websites. $Link_{cookie}(w, w')$ and $Link(w, w')$ identify a possible path, as a result of an inclusion or a redirection, between w and w' that can be exploited for tracking. $Access(w, w')$ and $Access_{cookie}(w, w')$ capture a successful redirection or inclusion that forces a user to contact website w' from the website w. $Cookie\_sync(w, w')$ indicates cookie syncing between w and w' to share IDs.

The predicates in Table 3.2 are obtained from the collected data as we will see in Section 3.5. The predicates in Table 3.3 are inferred from the rules of the model.

3.2.2 General Derivation Rules

We denote the pure "status" of the internet with $\mathcal{N}$ and the set of predicates capturing a specific mitigation $\mathcal{X}$ on the internet snapshot $\mathcal{N}$ with $\mathcal{N}^*_{\mathcal{X}}$.

Table 3.3: Web Tracking Predicates

The predicates are derived (D) from one or more rules of the model.

$Link(w, w')$	D	websites w and w' have a *possible* path to share information w/o exchange of cookies.
$Link_{cookie}(w, w')$	D	websites w and w' have a *possible* path to share information via cookies from w to w'.
$Access(w, w')$	D	website w forces to access a resource in w' via a redirection or an inclusion.
$Access_{cookie}(w, w')$	D	website w forces to access a resource in w' via a redirection (inclusion) attaching w's cookies.
$Cookie\_sync(w, w')$	D	website w synchronizes its cookies with the website w'. The operation is unidirectional.
$Knows(w, w')$	D	*potential* ability of website w to track users on a (possibly different) website w'.
$req\_COPPA(w)$	D	website w should comply to COPPA.

Definition 1 (Internet Snapshot). *The symbol $\mathcal{N}$, possibly with subscripts, denotes a finite (possibly empty) set of instantiated predicates from Table 3.2 and Table 3.3 that captures the interactions (inclusions, redirections, etc.) among websites observed on the Internet.* ■

For example, we can have $\mathcal{N} = \{IncludeContent(w, w'), Redirect(w', w''), \ldots\}$ and $\mathcal{N}^*_{\mathcal{X}} = \{Block\_request(w'), Block\_request(w''), \ldots\}$.

The turnstyle $\vdash$ separates the assumptions on the left from the conclusions on the right. The sequence of formulas on the left of $\vdash$ are in conjunction.

The capital letters A, B, and C, possibly with subscripts, denote formulae of the quantifier-free fragment of IFOL, whose predicates are drawn from Table 3.2 and Table 3.3. The variable $w \in \mathcal{W}$, possibly with apices, is a variable over websites. Constants (e.g *facebook.com*) denote websites.

We can have domain-specific axioms of the form $A \to B$ that can be added to a derivation with the rule:

$$\frac{\mathcal{N}, A \to B \vdash C \quad A \to B \text{ is Domain Axiom}}{\mathcal{N} \vdash C}\ DomAx$$

We represent a domain-specific axiom $A_1 \wedge \ldots \wedge A_n \to B$ as a rule $\dfrac{A_1, \ldots, A_n}{B}$ and vice-versa as in IFOL, $\vdash$ and $\to$ are interchangeable. Tracking specific rules in Section 3.3 are indeed domain-specific axiom.

3.3 Tracking Specific Rules

Information Flow In Figure 3.1a the rules `IncludeW` and `RedirectW` show how a link between two websites w' and w can be created. Rule `Redirect` (`Include`) in Figure 3.1a illustrates how the redirection to (inclusion of) another website can be employed by trackers to pass information, for example, cookies. The rule `ImpRed` (`ImpInc`) in Figure 3.1b shows that the predicates $\mathit{Redirect}_{\mathit{cookie}}$ ($\mathit{IncludeContent}_{\mathit{cookie}}$) implies the predicates *Redirect* (*IncludeContent*).

Network Interactions Rules `AccessToW` and `AccessTo` in Figure 3.1a describe access to resources with a possible propagation of information between two websites w and w' (w/ or w/o an exchange of cookies). The rule `PropagateAccess` shows how the access can be propagated through websites.

Third-party Tracking The rule `3rdpartyTracking` in Figure 3.1a shows that 3rd-parties present on a website w can track users. This rule can be applied recursively to describe complex interactions among websites as shown in Appendix A. This rule does not consider the possibility of browser fingerprinting (not blocked by the *Block_tp_cookie* mitigation). Since we are interested in the data sharing process that brings to track users and no information on how fingerprints are shared is available. Furthermore, as pointed out by several works [143, 344], browser fingerprinting is not as accurate as cookies to identify users. Thus, ad transactions carried out without the presence of cookies are not enough to produce targeted advertisements [212]. We further discuss this extension in Section 3.9.

Cookie Syncing The rule `Sync` in Figure 3.1c shows the preconditions required to implement cookie syncing between websites w and w'. Cookie syncing requires the exchange of cookies to link the IDs used by the two trackers. This technique is also called *First to Third-party Cookie Syncing* [124]. Rule `PropagateSync` shows how to propagate cookie syncing through a sequence of websites.

Tracking via Cookie Syncing In Figure 3.1c, the rule `SyncTracking` describes how cookie syncing between websites w' and w'' allows one to track users even on websites where a tracker is not explicitly present. This is known as *Third to Third-party Cookie Syncing* [124]. We did not define a rule to describe *cookie forwarding* because it is a special case of `3rdpartyTracking` where the tracker passively receives the user's history

$\dfrac{\mathit{IncludeContent}(w, w')}{\mathit{Link}(w, w')}\ \texttt{IncludeW} \quad \dfrac{\mathit{Redirect}(w, w')}{\mathit{Link}(w, w')}\ \texttt{RedirectW}$	If a website w includes content from (redirects to) site w', then there is a link between w and w' that allows an exchange of information.
$\dfrac{\mathit{Redirect}_{\mathit{cookie}}(w, w')}{\mathit{Link}_{\mathit{cookie}}(w, w')}\ \texttt{Redirect} \quad \dfrac{\mathit{IncludeContent}_{\mathit{cookie}}(w, w')}{\mathit{Link}_{\mathit{cookie}}(w, w')}\ \texttt{Include}$	During a redirection (inclusion) it is possible to append a cookie of w for w'.
$\dfrac{\mathit{Link}(w, w') \quad \neg \mathit{Block\_request}(w')}{\mathit{Access}(w, w')}\ \texttt{AccessToW}$ $\dfrac{\mathit{Link}_{\mathit{cookie}}(w, w') \quad \neg \mathit{Block\_request}(w')}{\mathit{Access}_{\mathit{cookie}}(w, w')}\ \texttt{AccessTo}$	If a website w includes content from (redirects to) a website w' (this case includes connections exploiting social buttons) that is not blocked by any extension, then the user is forced to access the resources of w' from w.
$\dfrac{\mathit{Access}(w, w') \quad \mathit{Access}(w', w'')}{\mathit{Access}(w, w'')}\ \texttt{PropagateAccess}$	If a website w forces to access the resources of w' and w' forces to access the resources of w'', then the user that visits w is forced to access website w''.
$\dfrac{\begin{array}{l}\mathit{Visits}(w) \\ \mathit{Access}(w, w') \quad \neg \mathit{Block\_tp\_cookie}(w')\end{array}}{\mathit{Knows}(w', w)}\ \texttt{3rdpartyTracking}$	If a user visits a website w that forces to access a website w' not blocked by any mitigation, then w' knows that the user visited w.

(a) Tracking flow

$\dfrac{\mathit{IncludeContent}_{\mathit{cookie}}(w, w')}{\mathit{IncludeContent}(w, w')}\ \texttt{ImpInc} \quad \dfrac{\mathit{Redirect}_{\mathit{cookie}}(w, w')}{\mathit{Redirect}(w, w')}\ \texttt{ImpRed}$	$\mathit{Redirect}_{\mathit{cookie}}$ and $\mathit{IncludeContent}_{\mathit{cookie}}$ are a particular case of *Redirect* and *Include* respectively.

(b) Tracking Implications

$\dfrac{\mathit{Access}_{\mathit{cookie}}(w, w') \quad \neg \mathit{Block\_tp\_cookie}(w')}{\mathit{Cookie\_sync}(w, w')}\ \texttt{Sync}$ $\dfrac{\mathit{Cookie\_sync}(w, w') \quad \mathit{Cookie\_sync}(w', w'')}{\mathit{Cookie\_sync}(w, w'')}\ \texttt{PropagateSync}$	A website w redirects the user to a website w' inserting cookies of w in the request. If the connection to w' is not blocked by any mitigation and the browser allows w' to set its cookies then w' can receive w's cookies and synchronize them with its cookies. Cookie syncing can be propagated.
$\dfrac{\mathit{Knows}(w', w) \quad \mathit{Cookie\_sync}(w', w'')}{\mathit{Knows}(w'', w)}\ \texttt{SyncTracking}$	The presence of cookie syncing with w' allows a website w'' to track users on the website w even if it is not explicitly present.

(c) Tracking with Cookie Sharing

Figure 3.1: Tracking Derivation Rules

$\dfrac{Knows(w,w') \quad Kids(w') \quad w \neq w'}{req\_COPPA(w')}$	`COPPAcomplRelease`	If a website w tracks users on a children related website w', then w' should comply with COPPA. This rule covers case (2).
$\dfrac{Knows(w,w') \quad Kids(w') \quad w \neq w'}{req\_COPPA(w)}$	`COPPAcomplCollect`	If a website w tracks users on a children related website w', then w should comply with COPPA. This rule covers case (4).
$\dfrac{Kids(w) \quad Knows(w,w') \quad BehavioralAds(w)}{req\_COPPA(w)}$	`COPPAcomplBehav`	If w is a children related website that collects PII on an external website w' then it can perform behavioral advertising. This rule covers the cases (1) and (3).
$\dfrac{Kids(w) \quad Cookie\_sync(w',w)}{req\_COPPA(w)}$	`COPPAcomplCS`	It is a special case of `COPPAcomplBehav`. If w is a children related website and performs cookie syncing with w' (i.e. it receives cookies from w') then it can create profiles for its users for behavioral advertising.

(a) COPPA compliant

Figure 3.2: COPPA Derivation Rules

collected by a 3rd-party. The cookies forwarded could be used for back-office exchange that is outside of the scope of our model. All the rules assume the intention of the websites to track users.

COPPA Compliance A website w must comply with COPPA if at least one of these conditions hold [2]:

1. w is directed to children under 13 y/o and it collects PII.
2. w is directed to children under 13 y/o and it allows another website w' to collect PII.
3. w has a general audience, but it has actual knowledge that it collects PII from children under 13 y/o.
4. w collects PII from users of a website w' directed to children under 13 y/o.

However, websites that fall in conditions (1) and (3) and collect *only* persistent identifiers (e.g. cookies) are not obliged to comply with COPPA if the persistent identifier is used for internal operations *only*. It is important to underline that this exception does not allow behavioral advertising. Figure 3.2a shows the rules that describe when a website should comply with COPPA. The predicate $\mathit{Kids}(w)$ describes a website directed to children under 13 y/o, $\mathit{req\_COPPA}(w)$ identifies a website that should comply with COPPA.

`COPPAcomplRelease` and `COPPAcomplCollect` describe conditions (2) and (4): if a children-related website w' allows a website w to track its users then both websites must comply with COPPA. We impose $w \neq w'$ to not fall in the conditions (1) and (3) where COPPA is not mandatory if used for internal activities. It is important to underline that in our model the *Knows* predicate implies the employment of persistent identifiers (e.g. cookies). The scenario described in `COPPAcomplCollect` is not always straightforward to be observed due to the exchange of cookies (e.g. cookie syncing) among websites.

`COPPAcomplBehav` captures cases (1) and (3). Our model describes only personal identifiers, thus we need to determine if a certain website uses this information for external operations (e.g. behavioral advertising). $\mathit{BehavioralAds}(w)$ could be instantiated using the approach presented by Liu et al. [201] and we leave for future work the integration with our model. Rule `COPPAcomplCS` shows a special case of `COPPAcomplBehav` where a children-related website w receives cookies from another website. Cookie syncing is a known technique utilized for behavioral advertising [248]. It is important to underline that the opposite case ($\mathit{Cookie\_sync}(w, w')$), in which the children-related website w sends cookies to an external website, is already covered by the rule `COPPAcomplCollect` since $\mathit{Cookie\_sync}(w, w')$ generates a $\mathit{Knows}(w', w)$ that triggers the mentioned rule.

Do we really need a formal approach? Consider Youtube and assume $Kids(youtube.com)$ always holds (as sometimes it might be necessary to treat information according to COPPA). We might be tempted to conclude that any website that includes cookies from *youtube.com* should be COPPA compliant. This informal reasoning seems to imply that any website importing a social button or a video from Youtube should be COPPA compliant. However, by applying the set of rules we previously presented we can instead prove that this is actually incorrect. Suppose $\mathit{Kids}(\mathit{youtube.com})$, we have an $IncludeContent(\mathit{abc.com}, \mathit{youtube.com})$ due to the social gadget, and thus by applying rules `IncludeW`, `AccessToW`, and `3rdpartyTracking` we have $Knows(youtube.com, \mathit{abc.com})$. At this point, none of the COPPA rules can produce $\mathit{req\_COPPA}(\mathit{abc.com})$. Instead, it is possible to trigger rule `COPPAcomplBehav` by showing that *youtube.com* is performing behavioral advertising and thus should comply with COPPA.

As previously stated, the predicate *Knows* describes the *potential* ability of a website to track users. Thus, the obtainment of the predicate *req_COPPA* is not by itself a *definitive* proof of the need for compliance. However, it provides an explanation that can trigger further investigations by the FTC on which data are actually sent (for example, due to a complaint from a parent). Websites must then prove that the exchange either did not occur or did not contain children's information.

3.4 Using the Calculus for Tracking Relations

We formally define tracking relations that are of practical interest through our formal model. We illustrate some of these relations in the practical case of Alexa Top 5, 10, 50, and 100 websites later in Section 3.6.

Flow Propagation Given the sharing of information through redirections, content inclusions, and cookie syncing and given a sequence of visited web sites, we can study how the knowledge about this sequence is distributed on the Internet. This is possible through a graph where we underline edges with predicate $Knows(w', w)$ to identify the websites that know if a user visited another website. We can map this representation to a Venn diagram where we identify which trackers are directly and indirectly included in the websites visited. We define a mapping between the predicate *Knows* and the set theory:

$$KnowsUser(\mathcal{N}, w) = \{w^* \mid \mathcal{N} \vdash Knows(w^*, w)\} \tag{3.1}$$

where $KnowsUser(\mathcal{N}, w)$ represent the set of websites w^* that are able to track a user on the website w in an Internet snapshot $\mathcal{N}$.

Lowest Tracking Coverage Our formal model generates relations between websites through *Knows* predicates for a given $\mathcal{N}$. We compare different mitigations to determine which produces the lowest tracking coverage. A mitigation $\mathcal{X}$ in an Internet snapshot $\mathcal{N}$ ($\mathcal{N}^*_{\mathcal{X}}$), disables some *Knows* predicates.

Definition 2 (**Mitigation subsumption**). *Let $\mathcal{N}$ be an Internet snapshot and $\mathcal{N}^*_{\mathcal{X}}$,$\mathcal{N}^*_{\mathcal{Y}}$ two mitigations. We say that the mitigation $\mathcal{N}^*_{\mathcal{X}}$ is* more effective *than $\mathcal{N}^*_{\mathcal{Y}}$ iff $\forall$ pairs (w, w'): $\mathcal{N}, \mathcal{N}^*_{\mathcal{X}} \vdash Knows(w', w)$ implies $\mathcal{N}, \mathcal{N}^*_{\mathcal{Y}} \vdash Knows(w', w)$.* ■

Intuitively, any *Knows* predicate obtained from $\mathcal{N}$ applying the mitigation $\mathcal{N}^*_{\mathcal{X}}$ is also obtained from $\mathcal{N}$ applying $\mathcal{N}^*_{\mathcal{Y}}$. Unfortunately, this definition can be rarely applied. Indeed, if the mitigations modify different parts of the graph of *Knows* predicates, the results cannot be compared. Thus, we propose a quantitative analysis.

Given an Internet snapshot $\mathcal{N}$, a mitigation $\mathcal{N}^*_{\mathcal{X}}$ is better than a mitigation $\mathcal{N}^*_{\mathcal{Y}}$ if *both* conditions hold:

- *C1:* The mitigation $\mathcal{N}^*_{\mathcal{X}}$ blocks access to a smaller number of websites compared to the mitigation $\mathcal{N}^*_{\mathcal{Y}}$
- *C2:* The trackers obtained with $\mathcal{N}^*_{\mathcal{X}}$ are smaller in number compared to the trackers obtained with $\mathcal{N}^*_{\mathcal{Y}}$

Given an Internet snapshot $\mathcal{N}$, we define its *access size* and *knowledge size* respectively as follows

$$||\mathcal{N}||_A = \sum_{w \in \mathcal{W}} \left|\{(w, w^*) \in \mathcal{W}^2 \mid \mathcal{N} \vdash \mathit{Access}(w, w^*)\}\right|$$
$$||\mathcal{N}||_K = \sum_{w \in \mathcal{W}} \left|\{(w, w^*) \in \mathcal{W}^2 \mid \mathcal{N} \vdash \mathit{Knows}(w, w^*)\}\right|$$

Site breakage, used to compare mitigations' performance [166], is directly influenced by the *access size*. We can now compare two mitigations $\mathcal{N}^*_{\mathcal{X}}$ and $\mathcal{N}^*_{\mathcal{Y}}$

Definition 3 (**Quantitative mitigation subsumption**). *Mitigation $\mathcal{N}^*_{\mathcal{X}}$ is* quantitatively more effective *than mitigation $\mathcal{N}^*_{\mathcal{Y}}$ iff the access size of $\mathcal{X}$ is larger (or equal) than the access size of $\mathcal{Y}$, $||\mathcal{N}^*_{\mathcal{X}}||_A \geq \left|\left|\mathcal{N}^*_{\mathcal{Y}}\right|\right|_A$, and the knowledge size of $\mathcal{X}$ is smaller than the knowledge size of $\mathcal{Y}$, $||\mathcal{N}^*_{\mathcal{X}}||_K < \left|\left|\mathcal{N}^*_{\mathcal{Y}}\right|\right|_K$.* ■

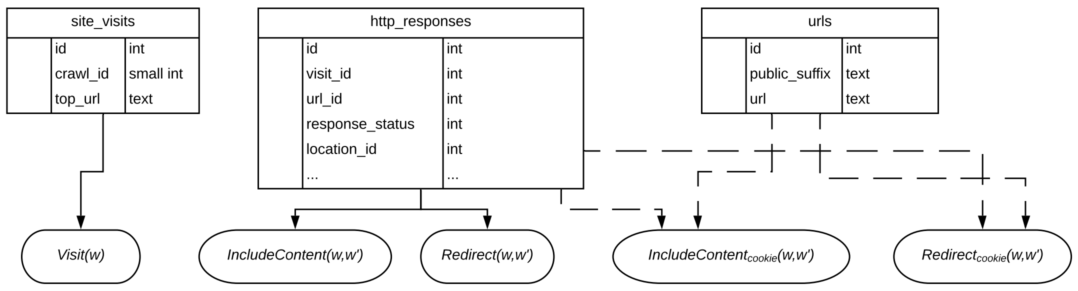
Intuitively the mitigation $\mathcal{X}$ reduces the number of trackers that know the user's visited websites more than $\mathcal{Y}$ does, while still keeping a larger (or equal) number of accessible sites than $\mathcal{Y}$. The ideal performance would be to drop one accessed site per blocked accessed tracker (i.e we lost only the tracker itself).

However, one of the major concerns in terms of privacy is not that a high number of trackers knows about fragments of a user's activity but that few trackers can reconstruct (almost entirely) the activity of a user. For example, *Google* is present in roughly 80% of the Top 1 million domains [116] and thus, has a high tracking coverage. We thus propose an additional definition:

Definition 4 (**Mitigation subsump. per tracker**). *Mitigation $\mathcal{N}^*_{\mathcal{X}}$ is* quantitatively more effective *than mitigation $\mathcal{N}^*_{\mathcal{Y}}$ against the tracker w iff the access size of $\mathcal{X}$ is larger (or equal) than the access size of $\mathcal{Y}$, $||\mathcal{N}^*_{\mathcal{X}}||_A \geq \left|\left|\mathcal{N}^*_{\mathcal{Y}}\right|\right|_A$, and the knowledge size of $\mathcal{X}$ projected to w is smaller than the knowledge size of $\mathcal{Y}$ projected to w.* ■

In other words, the mitigation $\mathcal{N}^*_{\mathcal{X}}$ produces a higher reduction of websites where the tracker w can control a user compared to the mitigation $\mathcal{N}^*_{\mathcal{Y}}$ while still keeping a larger (or equal) number of accessible sites than $\mathcal{Y}$.

Unfortunately, it is hard to fulfill both conditions (as we will see in Section 3.6, Figure 3.7): being less tracked means losing more access.



Tables from OpenWPM used to instantiate the predicates *Visits*, *IncludeContent*, and *Redirect* of our model (continuous lines). It is also possible to instantiate the predicates $IncludeContent_{cookie}$ and $Redirect_{cookie}$ from the tables `http_responses` and `urls` (dashed lines) but in Section 3.6 we employed empirically validated pairs from [44].

Figure 3.3: Mapping of the predicates to the dataset.

3.5 Instantiation and Scalability

We summarize the tables and columns employed to instantiate the predicates of our model in Figure 3.3. The table `site_visits` contains the list of the Top 10k Alexa visited domains. The table `urls` contains the set of URLs loaded during the crawling. The table `http_responses` contains the HTTP responses.

From the dataset, we extracted the sequence of redirections and inclusions necessary to instantiate the predicates. From table `http_responses` we employed *visit_ id* (ids for the top 10k websites), *url_ id* (ids for URLs of the HTTP responses), *response_ status* (HTTP Status Codes), *location_ id* (in case of redirection, ids for a new URL to visit. NULL otherwise), and *time_ stamp* (timestamps of HTTP responses).

Table 3.4 shows the number of HTTP responses received for the Top Alexa. The responses are used to find the sequence of redirections. In addition, Table 3.4 shows the number of predicates obtained by applying our model on the Top 10, 20, 30, 40, and 50 Alexa domains of the dataset without any mitigation. After visiting the Top 50 domains the users contacted 190 different websites with more than 6k connections. The number of redirections remains relatively small compared to the number of inclusions observed.

The number of HTTP responses in Table 3.4 for the Top 30, 40 and 50 domains are slightly different from the values of $Link(w, w')$ because the crawler failed to collect some HTTP responses during a sequence of redirections probably due to network problems. However, we can correctly close the sequence even if a response is missing.

Table 3.4: # of predicates and HTTP responses for the Top Alexa

Variables vs Top Domains	10	20	30	40	50
HTTP responses	925	1957	2864	3618	4530
$IncludeContent(w, w')$	824	1803	2681	3391	4272
$Redirect(w, w')$	101	154	184	229	261
$Link(w, w')$	925	1957	2865	3620	4533
$Link_{cookie}(w, w')$	3	3	3	5	6
$Access(w, w')$	925	2272	3636	5024	6382
$Access_{cookie}(w, w')$	3	3	3	5	6
$Cookie\_sync(w, w')$	3	3	3	7	8

Scalability

To show the decidability of our construction we rely on the relation between logic programs and a fragment of intuitionistic logic (in particular Harrop formulae [221]).

Theorem 1 (PTIME Knows decidability). *It is possible to decide whether the internet snapshot $\mathcal{N}$ allows a website w^* to know about the user's visit to another website w ($\mathcal{N} \vdash Knows(w^*, w)$) in polynomial time in the size of the snapshot $\mathcal{N}$.* ■

Proof. We rely on embedding both snapshot and rules as a Harrop formulae.

$$G ::= A \mid G_1 \wedge G_2 \mid H \to G \mid \tag{3.2}$$

$$\mid G_1 \vee G_2 \mid \forall w G \mid \exists w G \qquad \text{\%Not used here}$$

$$H ::= A \mid G \to A \mid \forall w H \mid H_1 \wedge H_2 \tag{3.3}$$

where A is a predicate, G is a *goal formula* and H is an *Harrop formula.* An internet snapshot $\mathcal{N}$ is encoded as a (large) conjunction which is a Harrop formula. Each rule from Section 3.3 is encoded as a goal formula. For example, `3rdpartyTracking` can be coded as a Harrop formula:

$$\overbrace{\forall w w' \; \overbrace{\overbrace{\overbrace{Visits(w)}^{G::=A} \wedge \overbrace{Access(w, w')}^{G::=A} \wedge \overbrace{Block\_tp\_cookie(w') \to \bot}^{G::=H \to A}}^{G::=G_1 \wedge G_2 \wedge G_3} \to \overbrace{Knows(w', w)}^{A}}^{H::=G \to A}}^{H::=\forall w \forall w' H}$$

From Theorem A in [221] the pair of the query and the rules LJ from Appendix A are a logic programming language. As we have no disjunction on the right of $\vdash$ for the query of interest, the rules ($\vee R_i$) responsible for the PSPACE complexity of intuitionistic logic do not apply.

Since $\mathcal{N}$ is finite, there are at most $O(|\mathcal{N}|^2)$ different constants as we have at most two arguments for each predicate. Hence, the instantiation of all quantified formulae embedding the rules generates *at most* $O(|\mathcal{N}|^6)$ ground propositional rules (we have at most three variables per rule), even if no optimization can be done (e.g. distinguishing between content delivery networks and actual websites). Thus, the ground instantiation of the rules is poly in the size of the snapshot and also the query of interest can be decided in polynomial time. ■

We do *not* claim that the calculus of tracking for *arbitrary formulae* including knowledge predicates is tractable. The presence of disjunction on the right would make decidability jump to PSPACE [311]. From Theorem 1 follows that COPPA compliance rules can be also encoded as Hereditary Harrop formulas using the knowledge relations as basic atoms:

Corollary 1 (**PTIME COPPA compliance**). *It is possible to decide whether the internet snapshot $\mathcal{N}$ requires a website w to be COPPA Compliant ($\mathcal{N} \vdash req\_COPPA(w)$) in polynomial time in the size of $\mathcal{N}$.*

The complexity of $O(|\mathcal{N}|^6)$ is inadequate for the application of the approach beyond very compact domains. Indeed our goal is not to provide Internet-scale analysis but third-party verifiable evidence for *individual cases* where numbers are manageable. For example, users rarely visit more than 100/120 websites [53, 239][5], the cookie duration is typically short [145] and cliques, important for COPPA, are relatively small [116]. Thus, scalability in this application is not a problem.

There may be more than one proof because a prover can choose to apply one rule before another one according to a suitable heuristic that may lead to a faster proof search (see GAPT [113] for additional information). Different proofs may also come from the existence of different tracking possibilities on the Internet. The important thing is that one can be found in polynomial time (see Theorem 1).

Theorem Proving Implementation We leverage the GAPT tool [113] to generate proofs for the *Knows* and the *req_COPPA* predicates. We use the intuitionistic prover `Slakje` [129] to produce formal proofs based on the rules in our model. We encode the model and the data using the TPTP syntax. Figure A.5 in Appendix A shows an example of proof for *req_COPPA*. The data used for the axioms is *generated from actual data* obtained using OpenWPM. We instantiated the *Kids*(w) predicate using the Top

[5]Skewed towards tech-savvy users, thus these values are likely upper bounds.

Table 3.5: Timing for successful and failed proof attempts

Run `Slakje` to prove *Knows*(*fbcdn.net*, *facebook.com*) and viceversa (not provable) with different visited domains

# visited domains	TPTP input [# axioms]	Successful proof Time [sec]	Failed proof Time [sec]
5	75	1.4	1.1
10	209	1.8	1.6
50	867	10.5	19.3
100	2,343	1,469.8	>3,600

50 Alexa In the *Kids and Teens* category. This approach is fully automated by a script that generates a sequence of axioms from the database, the model, and the conjecture to prove.

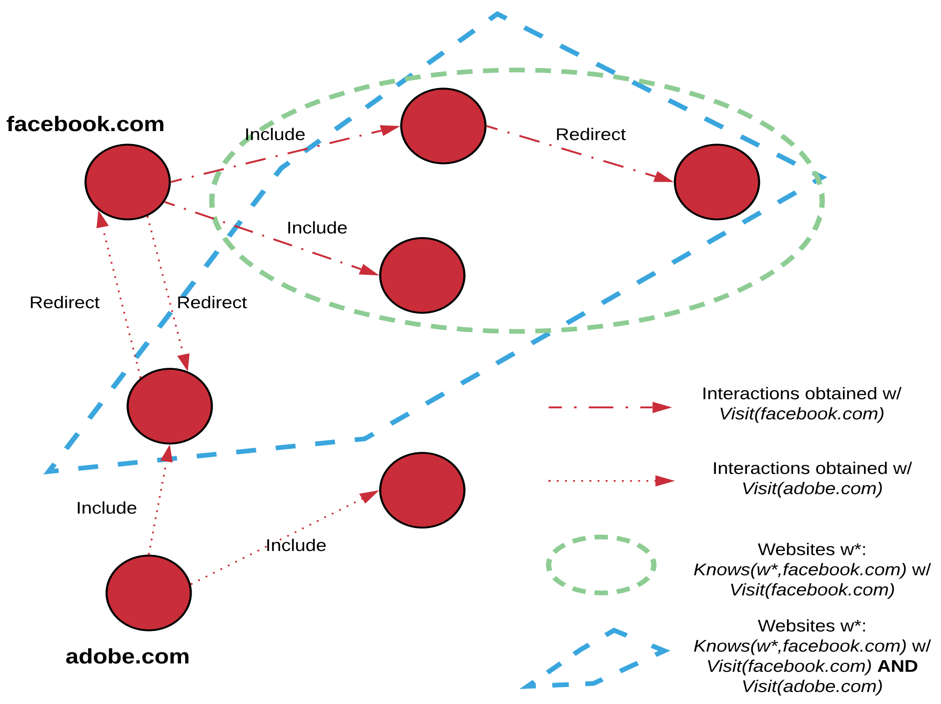
We evaluated the performance of `Slakje` by assuming the Top 5, 10, 50, and 100 as *visited* domains to generate a proof for $Knows(facebook.com, fbcdn.net)$ and the vice versa. Table 3.5 shows the performance with an Intel i7-8750H @ 2.20GHz and 2 GB RAM for the Java VM.

As shown in Table 3.5, the time required to generate a proof increases with the number of axioms. This number is dependent on the interactions observed by the user on the *visited* websites. To improve performance we perform DBMS slicing by extracting only the interactions that are obtained from the user's visited websites (e.g. Top5, Top10, etc.) and not the entire Internet interactions and then perform proof reconstruction. Unsound search followed by proof reconstruction is a new trend in Automated Reasoning [54]. This is the minimum set of interactions (and thus axioms) that must be considered to avoid missing possible tracking practices. For example, if we extract only the interactions generated by visiting a website w and not all the other visited websites we can miss interactions generated from other visits that reach w as shown in Figure 3.4.

3.6 Evaluation of Filter Lists Mitigations

We can now answer to ***RQ1*** and evaluated our approach on the dataset previously presented with the filter list of three widely deployed extensions (`Ghostery`, `Disconnect`, `Adblock Plus`). We neither consider the Firefox third-party cookie blocking feature for unvisited websites[6] nor other Firefox configurations that were either too restrictive

[6]This feature is unable to block Google in certain situations. Firefox employs by default the Google search engine and, thus, establishes connections with Google domains if the website is not accessed


facebook.com
Include
Redirect
Include
Redirect
Redirect
Include
Include
adobe.com
Interactions obtained w/ Visit(facebook.com)
Interactions obtained w/ Visit(adobe.com)
Websites w*: Knows(w*,facebook.com) w/ Visit(facebook.com)
Websites w*: Knows(w*,facebook.com) w/ Visit(facebook.com) AND Visit(adobe.com)

Determine which websites w' knows about the visit of *facebook.com* ($Knows(w', facebook.com)$) by analyzing only on the interactions generated by the *facebook.com* visit misses interactions generated by *adobe.com* (visited by the user) with *facebook.com* and thus potential trackers.

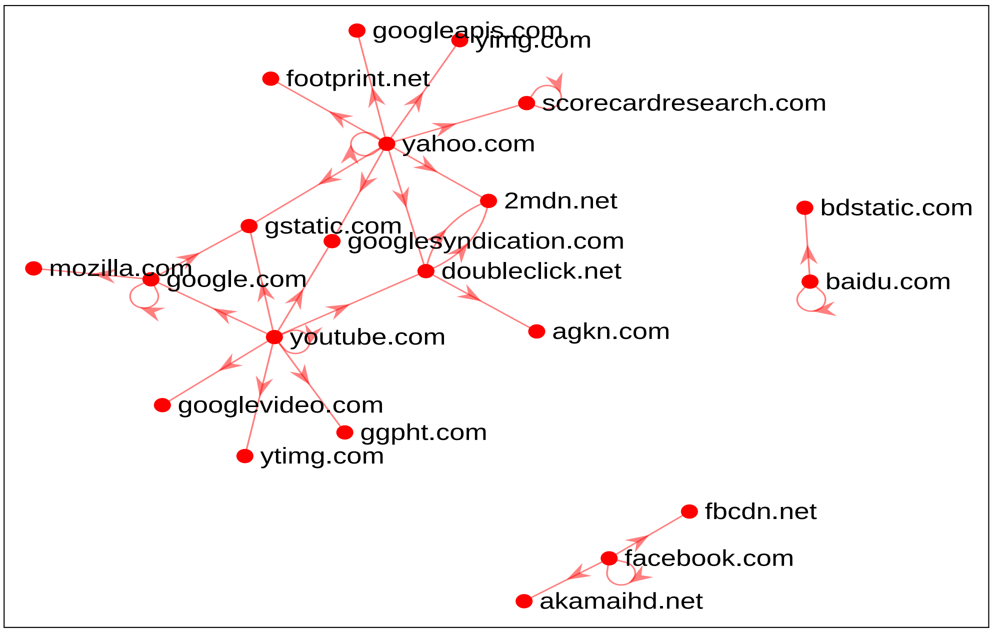
Figure 3.4: The problem of determine $Knows(w', facebook.com)$

(e.g. block all cookies) or overlap (e.g. Firefox uses `Disconnect` blacklist). We used the blacklist of `Ghostery`, `Disconnect`, and `Adblock Plus` from Bashir et al. [44, 45] (the data was *collected* in 2016 too). We then compared the effectiveness of some of the mitigations (`Disconnect` and `Adblock Plus`) in 2016 with their 2019 version. We also extended the comparison with `Privacy Badger` and `Adblock Plus` (enforced with *EasyList&EasyPrivacy*) in the 2019 scenario.

3.6.1 Results

Flow Propagation Figure 3.5 shows the graph of *Access* obtained by applying our rules on the Top 5 Alexa domains without any mitigation. While Figure 3.6 shows the Venn diagrams obtained computing the *Knows* predicates of the model without any mitigation and with the `Disconnect` mitigation.

directly through its domain name (we assume non-tech-savvy users behave in this way). As a result, Google domains (and all its subdomains) are whitelisted by Firefox and can bypass the third-party cookies block.



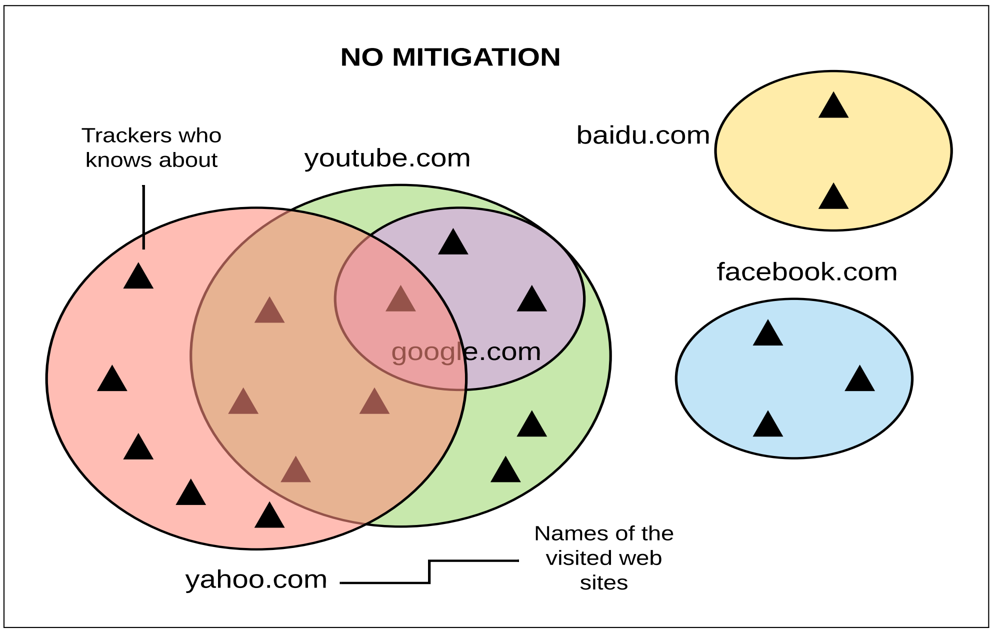
Access predicates obtained without any mitigation in the Top 5 Alexa domains. Several connections are made to different third-party domains. Understanding how many trackers can potentially know about your *youtube.com* visits is far from trivial (even ignoring any back-office sharing agreement).

Figure 3.5: *Access* graph Top 5 Alexa domains

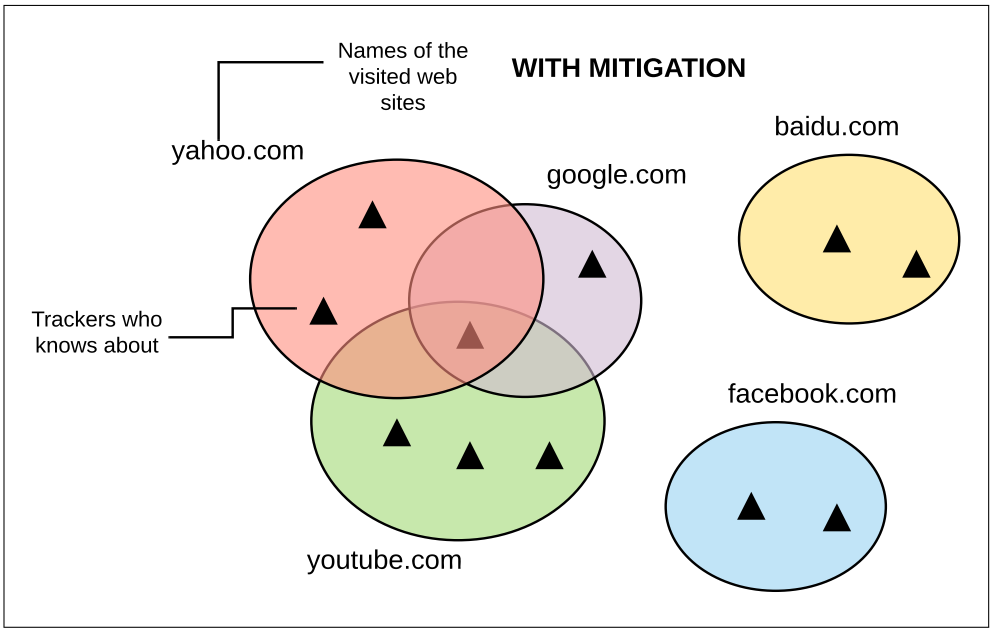
Lowest Tracking Coverage Figure 3.6a shows the Venn diagrams for the Top 5 domains without any mitigation ($\mathcal{N}^*_{\mathcal{B}} = \emptyset$) while, Figure 3.6b shows the Venn diagram with `Disconnect` mitigation ($\mathcal{N}^*_{\mathcal{A}} =$ `Disconnect`). From Def. 2 we have that $\mathcal{N}^*_{\mathcal{A}}$ is more effective than $\mathcal{N}^*_{\mathcal{B}}$.

Comparing different Mitigations We compared the effectiveness of the filter list of `Ghostery`, `Disconnect`, `Adblock Plus` (based on *EasyList*) in 2016 and `Disconnect`, `Adblock Plus` (based on *EasyList*), `Adblock Plus` (enforced with *EasyList&EasyPrivacy*), and `Privacy Badger` ("trained" on the Top 200 Alexa domains in December 2019[7]) in 2019 based on the Def. 3 presented in Section 3.4. We extracted the requests from the dataset and we recursively apply the filter lists of the different mitigations to the connections established for the Top 5, 10, 50, and 100 Alexa domains. Except for `Privacy Badger`, which provides the list of domains it "learned" either to block completely ($Block\_request(w)$) or to stop setting the cookies ($Block\_tp\_cookie(w)$), for all the other mitigations we rely on their blacklist of domains, i.e. only $Block\_request(w)$.

[7]Now the construction of the blocklist is not performed anymore on the individual machine due to the risk of fingerprinting [126]



(a) *KnowsUser*$(\mathcal{N}, w)$ without mitigations. Each circle is a visited Top 5 Alexa site and includes trackers which can potentially know about this visit



(b) *KnowsUser*$(\mathcal{N}, w)$ with `Disconnect` mitigation. `Disconnect` significantly limits potential trackers when visiting *youtube.com* (from 9 to 4) and *yahoo.com* (from 9 to 3)

Figure 3.6: Comparing Tracking Knowledge for Alexa Top 5.

The results are normalized to $\mathcal{N}$ without any mitigation.

We employed the filter lists from [44,45] and the database previously presented to analyze the effectiveness of the mitigations in 2016. We then computed the current effectiveness of the filter list of `Adblock Plus` (with and without the addition of the *EasyPrivacy* list), `Disconnect`, and `Privacy Badger` in 2019 with an up-to-date database[8] from June 2019. Figure 3.7 shows the comparison of the mitigations. The dashed line and the dash-dotted line correspond to two different efficiency levels. The first is a 1-for-1 drop: for each connection that the mitigation blocks, it blocks one tracker, while the second represents a 1-for-2 drop: for each connection that the mitigation blocks, it blocks two trackers. Figure 3.7a shows that, among the filter lists in 2016, `Disconnect` is the most aggressive mitigation up to the Top 100 domains, where `Ghostery` behaves similarly. `Adblock Plus` is the most permissible mitigation in 2016. However, `Adblock Plus` shows a big increment of efficacy in its 2019 version. For example, in the Top 100, a 26% reduction of the accessed content generates a 66% decrement in trackers. It is worth mention that the filter list of `Adblock Plus` from [45] is also roughly 46 times smaller than the list in 2019 and that currently there is overlap between `EasyList` and `EasyPrivacy` [166]. In contrast, `Disconnect` does not significantly improve in 2019 with a more restrictive behavior. We found that the combination of EasyList and EasyPrivacy (`EasyList&EasyPrivacy`) achieves the highest protection at the cost of the most restrictive approach. Finally, `Privacy Badger` showed a similar level of protection compared to `Disconnect` and `EasyList&EasyPrivacy` but with a significantly higher number of connections allowed due to the balancing of blocking connections and cookies.

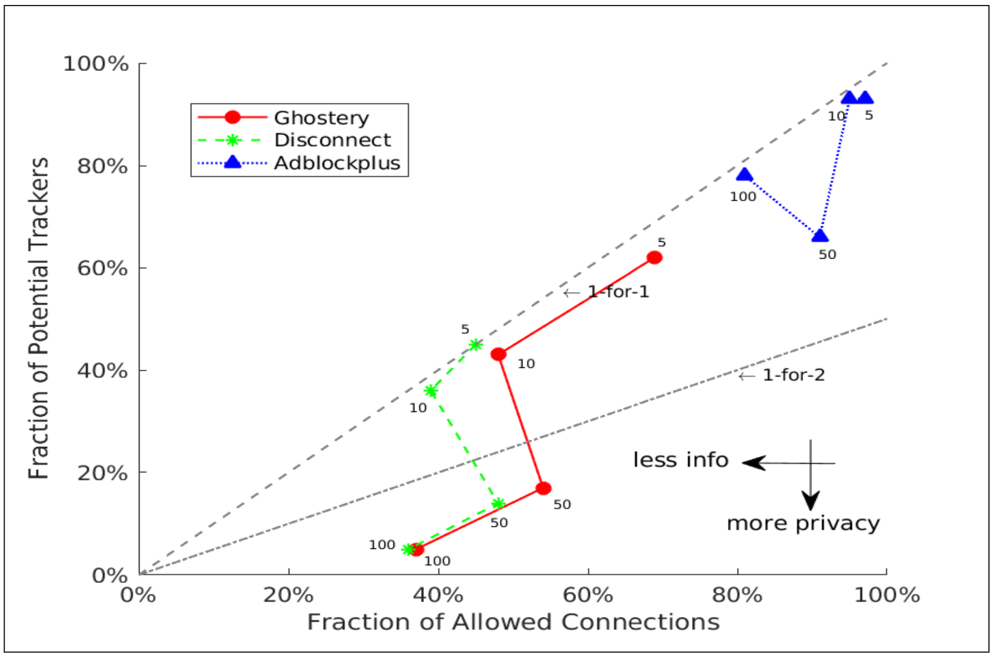
3.6.2 Comparison of Effectiveness with Related Work

We compared our findings[9] with the results of [124] and [218]. Albeit our analysis is currently limited to the Top 100 domains, while [124] and [218] analyzed the top 10k and 200k domains respectively. It is still of interest to see if similar results can be obtained from a formal model and thus justifiable to third parties instead of a pure experiment[10]. Table 3.6 shows the comparison of the effectiveness of the mitigations in terms of percentage of allowed connections for different works. Fouad et al. [124] found that `Disconnect` and `EasyList&EasyPrivacy` are roughly comparable in terms of % of allowed requests,

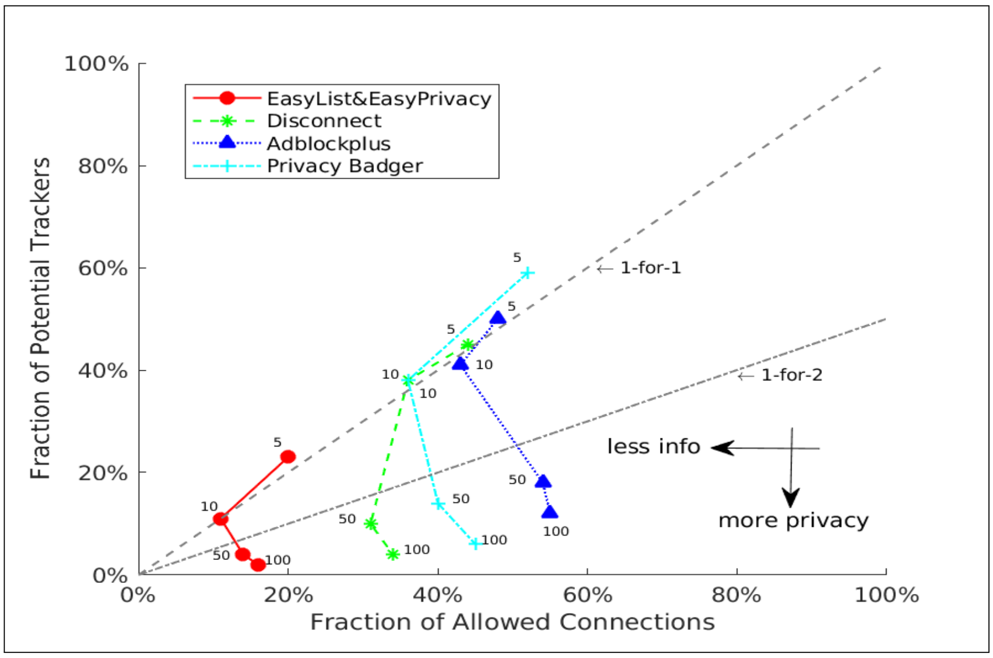
[8]The database contains the same information of the 2016 database with small differences in the structure, for example, the 2019 version presents a table for the redirections.

[9]Here we limit our analysis for the 2019 database.

[10]The results of the previous works are obtained crawling the web with OpenWPM and the chosen mitigation.



(a) Effectiveness of the mitigations in 2016



(b) Effectiveness of the mitigations in 2019

Comparison between the fraction of potential trackers (counted as # of unique *Knows*) and the allowed connections (counted as # of unique *Access*) on the Top 5, 10, 50, and 100 domains with different mitigations. The most aggressive mitigations are `Disconnect` in 2016 and `EasyList&EasyPrivacy` in 2019. `Adblock Plus` significantly increased its efficacy in 2019, while `Disconnect` did not improve in terms of protection. `Privacy Badger` shows performance comparable to `Disconnect` in terms of trackers blocked with a less conservative approach.

Figure 3.7: From 2016 to 2019 mitigations reduced the amount of access to gain privacy

Table 3.6: Comparison of % of allowed connections by `Adblock Plus` (AdB), `Disconnect` (D), `Privacy Badger` (PB), and `EasyPrivacy&EasyList` (EL&EP) with previous works

Paper	`AdB`	`D`	`PB`	`EL&EP`
Our work	54.9%	33.5%	44.8%	16.4%
[124]	$\approx$ 60%	34.6%	30-35%	39.3%
[218]	65-70%	25-30%	$\approx$ 40%	N.D.

while we found that `EasyList&EasyPrivacy` is significantly more aggressive. However, we obtained similar values for `Disconnect` and `Adblock Plus`. `Privacy Badger` differs from [124] but it is similar to [218] probably due to the different size of the "training domains" used by the studies. Finally, as in [124] and [218], we confirmed that adblockers are less effective than trackers blockers.

3.7 Limitations

The presented framework fills the gap between graph approaches, which scale but lack transparency of the results, and manual inspection, which is explainable but cannot scale even for a few domains (see Table 3.4 where considering only the top 10 domains requires to analyze more than 900 interactions). For a user that wants to determine what is the best adds-on she should install on her browser, large-scale studies ([124, 218]) provide contrasting and unverifiable claims (see Table 3.6). Our framework provides a more focused analysis compared to Internet-scale crawling and an *explanation* of why a given mitigation is better than another via a formal proof. The proof could then be traduced to natural language format [271] to hide the complexity to the user.

Our model relies on observable sharing behaviors from the client-side and it misses back-office information flows. While these mechanisms constitute a sizable part of the data-sharing economy they cannot be externally measured until legislators would oblige to divulge to which sites such information is shared. When this would be available the model could be extended by including knows axioms for back-office data sharing. The same applies to cookie syncing, as well as the usage of cookies obtained from cookie forwarding, thus the results of cookie syncing represent a lower bound.

For the first approximation, we only focus on tracking based on cookies and we ignored other techniques (e.g. fingerprinting) for which we do not know how the process of data sharing is done and that it cannot be observed from the client-side.

The database is obtained from previous projects and it could be not representative of the current status of the Internet. However, our model can be applied to any new

OpenWPM database. Still, the results apply to what it is possible to collect and observe using a crawler.

The filter lists of `Ghostery`, `Disconnect`, and `Adblock Plus`, obtained from [45], are domain blacklists. To maintain consistency in the comparison[11] with the 2019 version, we compared only the domain blacklist (for EasyList and EasyPrivacy we used the rules that begin with || and |).

The blacklist for `Disconnect` from [45] does not include any information about the *entity relationships*[12] that allows one to determine if a domain is loaded as first or third-party. We thus decided to not include this feature also in the 2019 evaluation. Future works can extend the analysis to consider the entity relationship. We did not produce a blacklist for `Ghostery` in 2019 because it is proprietary software and does not provide access to the blacklist and the mechanisms implemented. We plan to compare more mitigations in the future.

The results obtained from the computation of the *Knows* predicate represent an upper bound of the *observable* data sharing scenario, it shows the number of websites that *potentially* can collect information about users' habits due to the presence of interactions among them. We highlight that this situation describes the *worst* possible scenario for the privacy of a user, in which websites intentionally propagate the information collected to other partners. This choice is more conservative than some current approaches that only consider the elements of a blacklist as the all and only trackers.

3.8 Related Work

Over the last few years, researchers have identified different tracking techniques on the Internet. To make the chapter self-contained we present an overview of the techniques. Most papers examined the effectiveness of tracker blocking tools, while others focused on trackers' pervasiveness and the techniques used to track Internet users. We considered works about the Online Advertising Ecosystem, privacy policies, and formal modeling of the Internet infrastructure.

A Summary of User Identification *HTTP Cookies* are IDs associated with a user and are set by websites through JavaScript codes or HTTP responses. Cookies are automatically attached by the browser to all subsequent requests to the websites. The major

[11]EasyList and EasyPrivacy present a richer syntax, that allows one to block specific requests of scripts, etc.

[12]`https://feeding.cloud.geek.nz/posts/how-tracking-protection-works-in-firefox/`

difference compared to browser fingerprinting is that the ID is stored locally on the user's machine [71, 272].

Browser Fingerprinting is used by websites to collect information from the browser to build a unique fingerprint [188]. For example, to personalize the content, a website can request device-specific information like `user-agent`, `HTTP headers`, `plugins` and `browser extensions`, `fonts`, `screen resolution`, `OS`, `canvas` and `AudioContext` [7, 114, 225, 303] via HTTP headers or JavaScript codes [8]. These attributes can be used to generate a unique fingerprint for tracking purposes. Other approaches exploit O.S. and hardware properties to generate *device fingerprints* that allow cross-browser tracking [78, 285].

Other Browser Storage, for example `HTML5 localStorage`, `Flash LSOs`, and `HTTP headers` (e.g. `ETag`) [40, 304], are used by websites to store IDs and track users even if HTTP cookies are deleted.

Other tracking techniques exploit browsing history [239] and caching process of DNS records [179].

Data Sharing across Websites *Cookie Syncing* is an increasingly popular technique [248] employed by trackers to share the IDs associated to a user [7, 44, 116]. A common cookie syncing technique is to pass the IDs as parameters in an HTTP request. This procedure allows the websites to map different IDs to a single user and link information from different trackers.

Analysis of the Online Advertising Ecosystem Ghosh et al. [134] analyzed the leakage of information in the Real-Time Bidding (RTB) protocol and modeled the revenue of advertisers w/ and w/o syncing. Gill et al. [137] employed HTTP traces to model the revenue earned by different trackers, whereas Marotta et al. [212] empirically estimated the value of targeted advertisement depending on the presence or absence of cookies. We aim to address an orthogonal problem, i.e. how can we formally prove that cookies are employed for tracking users, independently from how trackers utilize them.

Iqbal et al. [166] developed a graph-based ML classifier of ads and trackers. The tool builds a graph representation of the HTML structure, the network requests, and the JavaScript in the web page to determine tracking practices from specific features. Gomer et al. [142] analyzed how the search context exposes users to tracking practices using directed network graphs based on referral header information. Bashir et al. [44, 45] detected flows of information between advertisers based on retargeted ads. They constructed an inclusion graph to model the advertising ecosystem, analyzed the graph properties, and simulated the impacts of tracker blocking tools like `Disconnect` and `Adblock Plus`. Kalavri et al. [171] represented traffic logs through 1-mode and 2-mode graphs to highlight the connected communities of the trackers and proposed an automated tracker detection mechanism based on graph properties.

Several papers employed relationships between websites to investigate Web tracking activities. However, we lack a formal description of the trackers' interactions to prove tracking practices and privacy compliance.

Formal Models for IT-Security Speicher et al. [307] used a model based on AI planning with grounded predicates in the context of the email infrastructure. Simeonovski et al. [299] proposed a model based on property graphs in the context of Internet core services.

We lack formal models to evaluate the effectiveness of mitigations against tracking practices.

Analysis of Children's Online Privacy Protection Act Compliance Data collection and web tracking are regulated by data protection laws. For example, the General Data Protection Regulation (GDPR) [3] is currently in force in the EU. Still, it is not uncommon to observe violations [214].

When it comes to collect personal information from children additional laws are applied. In the U.S. the Children's Online Privacy Protection Rule (COPPA) [2] imposes requirements for websites that collect personally identifiable information (PII)[13] from children under 13 y/o. COPPA requires posting a privacy policy containing the personal information collected, with who and for which goal this data is shared, to get verifiable parental consent (for example calling a tool-free number), and to allow parents to review the PII collected and revoke the consent. Determining if a website should comply with COPPA is not easy. There have been several violations in the past, for example by Playdom [1] and Youtube [4], with fines of several millions of dollars. Apart from U.S. FTC reports, some studies developed frameworks to analyze Android apps to determine violations [49, 266, 267]. However, there is not yet a tool for parents to determine such violations due to the complex interactions among websites.

Several papers tried to formally express privacy policy (e.g. [96, 216]). In the context of COPPA, Barth et al. [42] proposed a framework to formally describe this policy based on first-order temporal logic.

[13]E.g. First and last name, home address, SSN, persistent identifiers (e.g. cookies, fingerprinting), etc.

The state of the art focuses on the formal description of privacy policies. We lack a formal description of tracking behavior to generate proofs that websites should comply with privacy policies like COPPA based on security data.

3.9 Conclusions

We presented the first formal model to characterize tracking procedures based on data sharing. From the model, we extracted sharing relations to determine the tracking ecosystem, the effectiveness of different mitigations, and websites that should be COPPA compliant. We evaluated these properties on a real dataset (Top 100 Alexa sites) extracted from OpenWPM.

A tough question is whether the formal model bought us anything that we could not derive from running an alternative graph algorithm like the one discussed in Chapter 4. From a pure computational complexity perspective, the answer is no: being both in PTIME a data-crunching procedure must exist that maps the result of one into result of the other.

We argue that the difference is in the representation. A formal model produces *a result that can be independently checked* [228]. For example, one can produce a minimal derivation that shows that a website should be COPPA compliant and such derivation could be transformed (automatically) into a legal argument or a legal document. See [271] for a practical example where a formal logic model based on Datalog (also a PTIME inference framework) is used to reason about privacy practices and the results are presented to final customers (the local health authority) in a table or natural language format that is far easier to consume for them. It is important to underline that such proof is not by itself a *definitive* proof of COPPA violation but can be used by e.g. parents to trigger a first and more accurate investigation by appropriate agencies (e.g. FTC).

Future works can expand in several directions. Updated experimental results can be obtained by crawling again the internet with OpenWPM or improved algorithms to capture additional features. For example, the algorithm proposed by Fouad et al. [124], can be implemented to extract new data about cookie syncing. Another interesting extension would be to consider disjunctions in the mitigations for the rules. For example, in *3rdpartyTracking* one could include a disjunction on the (un)blocking of cookies or (dis)abling of scripts that are used for fingerprinting. We would also need to define fingerprint data-sharing agreements among parties. This might happen at the price of tractability. The model can be extended with mitigations that offer different fingerprints to different websites (assuming that websites know about the fingerprint via back-office agreements).

Chapter 4

Pareto-Optimal Defenses for the Web Infrastructure

The integrity of the content a user is exposed to when browsing the web relies on a plethora of non-web technologies and an infrastructure of interdependent hosts, communication technologies, and trust relations. Incidents like the Chinese Great Cannon make it painfully clear: the security of end users hinges on the security of the surrounding infrastructure: routing, DNS, content delivery, and the PKI.

There are many competing, but isolated proposals to increase security, from the network up to the application layer. So far, researchers have focused on analyzing attacks and defenses on specific layers. We still lack an evaluation of how, given the status quo of the web, these proposals can be combined, how effective they are, and at what cost the increase of security comes.

In this chapter, we employ security data from the Web infrastructure and its third-party dynamics to develop a graph-based analysis that considers a complex attacker model affecting different Internet layers and a multitude of security mitigations from IPsec to DNSSEC and SRI. With this approach, we evaluate the security of billions of users against several attack scenarios ranging from small hacker groups to nation-state actors. Our security data model and analysis provide an explanation of which mitigations are effective for each scenario, why, and under which conditions. Analyzing the infrastructure of the Top 5k Alexa domains, we discover that the security mechanisms currently deployed are ineffective and that some infrastructure providers have a comparable threat potential to nations. We find a considerable increase of security (up to 13% protected web visits) is possible at a relatively modest cost, due to the effectiveness of mitigations at the application and transport layer, which dominate expensive infrastructure enhancements such as DNSSEC and IPsec.

4.1 Introduction

Billions of people use the web daily for business and private life. Given the success of the web as a platform, the impact of attacks on the web is enormous. Users can be unconsciously forced to visit a phishing website of their bank website, redirected to an exploit kit using drive-by download attacks, execute scripts to mine cryptocurrency, or perform DDoS attacks. Securing the user's activity on the Web is a serious challenge: not only do servers hosting a domain's content need to be protected from compromise, but the reliance of many sites on external JavaScript means that a compromised third party will affect the including site's security. Moreover, the internet's infrastructure plays a key role in securing a domain. This infrastructure covers resolution of domains to IP addresses and routing of IP packets between different hosts. Even the mechanisms to ensure confidentiality and availability like TLS rely on a Public Key Infrastructure (PKI) system. Securing a website, therefore, is not something a site operator can achieve on their own. Instead, actors like internet service providers, internet exchanges, name service providers, content delivery networks, and certificate authorities influence the whole ecosystem. Thus, the security of the web ecosystem hinges on the infrastructure and all involved actors as a whole.

In this chapter, we present a methodology to evaluate existing security proposals against mass attacks on the Web. Various proposals have been made to improve the security of the Internet infrastructure in terms of routing (IPsec [177]), name resolution level (DNSSEC [35]), website delivery (HTTPS [264], HSTS [154]), public-key infrastructure (certificate transparency [190], DANE [155]), and third-party JS inclusions (CSP [43], SRI [356]). They all raise the level of security, but which combination of proposals is the most cost-effective, *considering the current infrastructure*? Are some of them too costly to deploy or simply less effective than existing proposals? We thus identify the following research question:

***RQ2*:** How can we evaluate the cost-effective selection of mitigations for securing the Web given the current Internet infrastructure?

To close this research gap, we developed a methodology to investigate the effectiveness of security mitigation strategies based on the graph database system Neo4J[1]. In this

[1]`https://neo4j.com/`

chapter we make the following contributions:

- We represent Web entities (domains, NSs, ASes, etc.) and their relationships using a property graph and exploited Neo4j reachability queries to compute attack graphs and determine the impact and cost of different mitigations (Section 4.4).
- We define a threat model for web-based attacks, which covers both aspects of the underlying infrastructure and web attacks themselves and a defender model which considers different defensive actions, their associated costs, and potential dependencies for deployment (e.g., DANE requires DNSSEC to be secure) (Sections 4.5 and 4.6).
- We analyzed the effectiveness and cost of different mitigation strategies securing clients that visit the Alexa Top 5k considering three classes of attackers: cyber-criminal groups, malicious infrastructure providers for cloud services and name resolutions, and nation-state attackers (Section 4.7).
- We create a web-based GUI at mitigation-web.github.io to analyze and investigate optimal mitigation deployments with customizable costs.

Goals: We provide a methodology to evaluate the cost-effective selection of mitigations for the entire Internet infrastructure as the result of global policy that aims at making the web secure from mass attacks. We analyze the effect of mitigations that can avoid or limit the attacker's ability to affect user visits on websites. Our goal is to provide a framework to identify criticalities for the security of the Web due to dependencies on countries and infrastructure providers and help determine the mitigations that should be implemented as policies.

Non-goals: We do not focus on the greater goal of the attacker. The results of exploiting weaknesses of the Internet infrastructure can range from cryptojacking to phishing, attacks against password managers, or DDoS [156, 181, 211, 238, 298, 316] depending on the attacker's motivation and falls outside the scope of this work. We do not consider targeted attacks on specific individual hosts, software, or companies nor provide ad hoc defenses for targeted individuals. The discussion of the incentives that can lead to the application of the mitigations as global policies are outside the scope of this work.

4.2 Planning Attacks and Defenses

Planning is an area of AI dedicated to general-purpose mechanisms that automatically find a *plan*, when given a high-level description of the (relevant part of) the world properties: *propositions*, an *initial state*, and a *goal* condition (see [133] for an introduction). A plan is a sequence of *actions*, each described in terms of a *precondition* and a *postcondition*, from the initial state to a state that fulfills the goal condition.

Speicher et. al. proposed *Stackelberg planning*, which can be seen as a two-fold classical planning task [306]. Inspired by work on Stackelberg security games, the defender (leader) moves first and the attacker (follower) can fully observe the defender's action and can plan their best response accordingly. In our notion of Stackelberg planning, the actions of the attacker have an *attacker reward* which is used as an indicator of the severity of the attack. Instead of a *plan* leading to a *goal state*, the set of *attacker actions* maximizing the attacker reward is computed, *e.g.,* the number of compromised domains. To prevent attacks and thus lower the attacker reward, the defender can change the world state through the application of *defender actions*, also referred to as "mitigations" which are assigned a cost. The defender pursues the objective to simultaneously minimizes its own cost and the *attacker reward for the resulting state after applying the defender plan.*

If the attacker can, *e.g.,* get hold of 10 domains, the defender weighs the damage done against its own cost. We avoid fixing this weight by instead considering the *Pareto frontier*, *i.e.,* the set of all Pareto optimal defender plans. A plan is dominated by another plan, if the second plan is either cheaper (strictly lower cost) but as effective (lower or same attacker reward), or vice versa (lower or same cost, strictly lower reward). Any plan that is dominated by no other plan is Pareto optimal and thus part of the Pareto frontier. The Pareto frontier gives us the set of all defender plans that are economically reasonable, *i.e.,* optimal for their respective budgets. It also gives us a step function from budgets to the level of security, *i.e.,* the remaining attacker reward, that is achieved by the optimal plan for this budget. Our model introduced in Sections 4.5 and 4.6 can be seen as a Stackelberg planning task, but instead of using Speicher et. al.'s algorithm to compute the solution (i.e., the Pareto frontier), we developed a graph-based algorithm.

Speicher et. al [306] derived a general-purpose algorithm for Stackelberg planning, which was successfully applied to the security of the email infrastructure [308]. Their algorithm uses a diverse set of optimizations and pruning techniques to reuse information gathered across different mitigation scenarios and to discover when mitigations are applicable in no particular order. However, when applied on a more complex and rich case like the Web and despite using all available optimizations, experiments only scaled up to about 50 domains, exceeding the available memory of 88 Gb after several days of computation [329]. This is due to the problem size: reading the input file and initializing internal data structures already takes hours, even though computing the attacker plan is simple once these structures are in place.

Hence we developed a new approach based on the *Neo4j* graph database system that allows one to store data in the form of a property graph (PG), i.e. a graph with different types of nodes and edges, and easily query the database by exploiting the relationships between nodes. As discussed in Section 2.2.1, the security of the Web relies on relationships among several entities. The Web infrastructure (e.g. Figure 2.1) naturally maps into this type of representation. From the Neo4j property graph and the set of rules of our model, instead of enumerating all relevant attacker actions in an input file (like in [308, 329]), we generate an attack graph that captures their relationships and thus allows efficient computation of the attacker reward via reachability analysis and application of defender actions via the removal of edges. Neo4j's data structures are optimized for such queries. Moreover, by representing the security data presented in Section 2.2.1 and Section 2.2.2 as a property graph, we drastically improve the generation time of this attack graph.

We use Neo4j to store and analyze a larger set of domains. In contrast to the fine-grained deployment analysis via the Stackelberg planning algorithm [306], which considered the best-possible mitigation *per host*, we consider a fixed set of mitigation scenarios. This is not necessarily optimal, as the optimal deployment can be a mix of two solutions. On the other hand, policy decisions often do not afford a per-host policy. Hence, our global policies are more realistic to be carried out.

4.3 Notation

We use the same notation for predicates and rules presented in Chapter 3. In contrast to the model in Chapter 3, where the rules are relatively simple and compact, in this scenario we do not provide a formal model as the rules are too complicated to generate a proof. Indeed, as discussed in Sections 3.5 and 4.2 large-scale analysis for the whole Internet cannot be addressed formally.

Given the complexity of the model, for readability, we directly instantiate variables to a specific domain. For example, to describe all actions that compromise a website via XSS, we could use the following action schema with the variable x in post and precondition and we can limit the (value) domain of x to the set of (network) domains D:

$$\frac{x \in D \quad XSS(x)}{C^{\text{web}}(x)}$$

Using Neo4J, we can efficiently evaluate these properties and find all satisfying assignments from variables to nodes in the graph.

4.4 Methodology

We now present the approach to answer to ***RQ2*** and evaluate security mitigation strategies using our model and the security data collected.

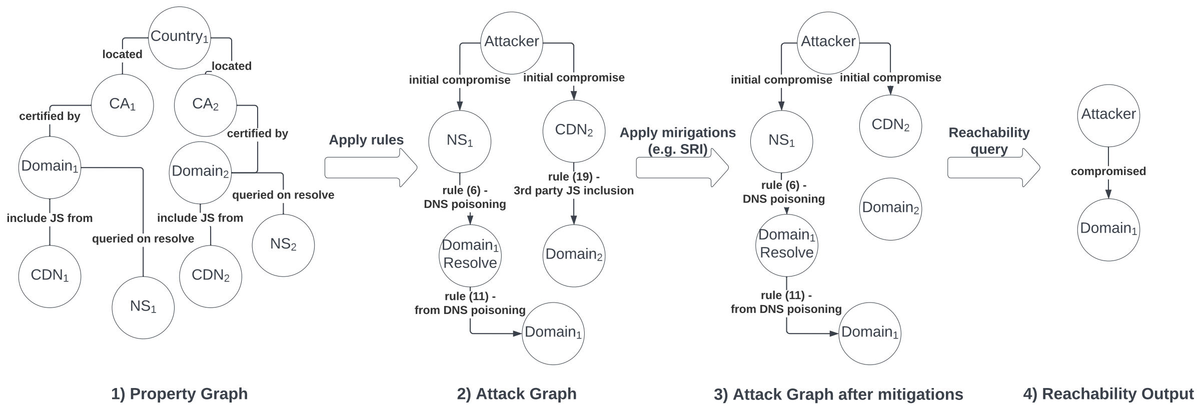
Figure 4.1 summarizes the procedure to perform a graph-based analysis. We start from the property graph describing the entire Web relationships. Starting from the attacker's initial assets we apply the rules of the model, that describe the attacker's actions, to generate an attack graph for the scenario. Finally, we apply a set of mitigations to remove some edges of the attack graph. We then queried the resulting attack graph using Neo4j to determine the reachability of the domains from the attacker node.

The attack graph is a directed graph with each node corresponding to a fact in the Stackelberg planning task and two nodes being connected if there is an attacker action that allows adding the latter (and only the latter) if the former is present. The initial assets are facts, and thus the graph would have multiple roots, however, to simplify reachability queries, we opted for a special root node `attacker` that connects to all initial assets and is the only node that is not a fact. The leaf nodes are the set of 'website compromised' nodes that are reachable when no mitigations are deployed, so that by removing edges, we can evaluate the impact of a countermeasure on the reachability of the leaves starting from `attacker`.

Note the difference to attack trees [291], where the root nodes describe a single goal and the parent-child relationship between subgoals can either be a conjunction (meaning all subgoals need to be achieved to reach the parent goal) or a disjunction. Our attack graphs are closer to Philips and Schwiler's attack graphs [257], where nodes describe the state of an attacker to a single goal. By contrast, we consider many goals and the state of the attacker corresponds to the set of nodes on the path(s) to the goal(s).

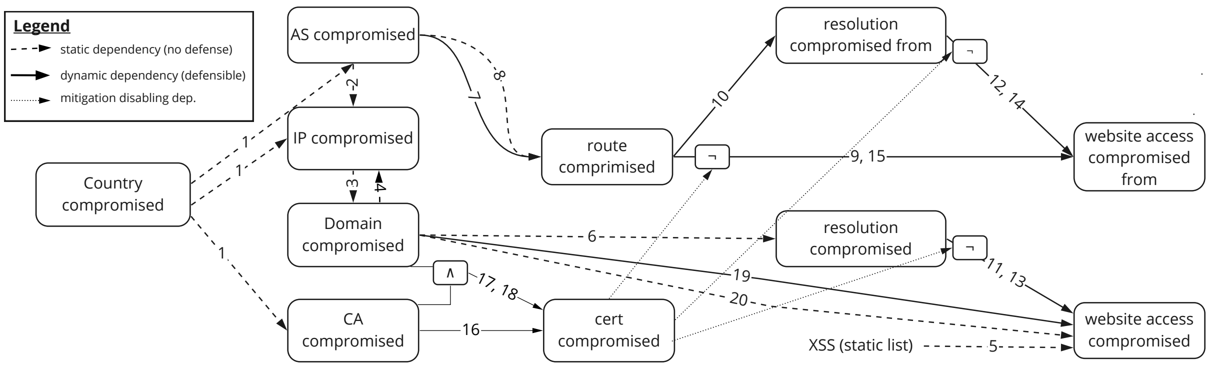
4.4.1 Rule Dependencies and Attack Graph Generation

To exploit Neo4J's strengths, we need to minimize the number of queries and computations outside the query evaluation. We exploit the structure of our threat model to this end. First, all rules have only a single postcondition. We can relate our rules in a dependency graph (Figure 4.2). Nodes are conditions, i.e., predicates with variables. Two nodes are connected if there is a rule with the first node as a precondition and the second as postcondition. Only rules B.17 and B.18 have two preconditions, which we indicate with the $\wedge$ symbol. Second, the dependency graph reveals that there is only a single loop (rules B.3, B.4) in this graph. Apart from this loop, every predicate can be derived with a bounded number of steps that corresponds to the length of the equivalent path



From the Neo4j property graph, that describes the Web Infrastructure, we generate an attack graph for an attack scenario based on the initial assets of the attacker and the rules of the model. A combination of mitigations is applied to the attack graph to disable edges. Finally, a reachability query in Neo4j is performed to determine the domains reachable from the attacker node.

Figure 4.1: Graph-based analysis for Stackelberg planning



The rules impose a hierarchy on the compromise predicates. A dashed edge indicates a static dependency, a solid edge a dynamic, i.e., defensible, dependency and a dotted edge an attacker action that can disable a mitigation.

Figure 4.2: Dependency rules graph

in our dependency graph. This allows us to express the attacker search with a bounded number of Neo4j queries that generate all predicates in the final state. The number of applications of B.3 and B.4 is unbounded in general, but in practice (and on our dataset) this fixpoint computation finishes after three steps. Disregarding the loop, we can use any topological order of the dependency graph to iteratively build the corresponding attack graph AG. At any step, we add the postconditions of the current rule r given that all possible instances of its preconditions are already present in the attack graph AG and that the graph conditions can be evaluated on property graph PG. The loop (B.3 and B.4) is handled separately.[2] In the resulting attack graph AG, a node represents an instantiation of a predicate and an edge represents an instantiation of a rule.

We generate one AG per attack scenario. By fixing the attack scenario, we can compute the effect of mitigations as a simple removal of edges and a reachability query in Neo4J. Algorithm 1 is used to translate a property graph into an attack graph exploiting the rule dependencies discussed before.

The algorithm applies to all threat models that can be described with a dependency graph, i.e., have a single positive postcondition and where mitigations only disable, but never enable attacker actions. It can handle loops but is most efficient if they concern only a small number of nodes in the dependency graph. Starting from the initial asset for the class of attackers, it traverses every rule in topological order w.r.t. the dependency graph (lines 3 and 12). For convenience, we introduce a starting note `attacker` that is connected to all the initial assets (line 1 and 2). For each rule, it formulates a query that generates the set of nodes (= compromise predicates) and edges (=actions) that represent applications of this rule valid for the property graph. Lines 6-9 handle the loop consisting of B.3 and B.4. The resulting graph AG after line 8 contains all attacker plans as paths starting from some 'initial asset' nodes.

4.4.2 Application of Mitigations

While the generation of the attack graph can be slow (several minutes), it allows a rapid computation of attacker success in given mitigation scenarios (order of seconds). As each edge in the attack graph corresponds to a rule, and mitigation predicates appear only in preconditions, the application of a mitigation corresponds to the addition or removal of

[2]For the general case of any dependency graph: We first compute all strongly connected components (SCCs) of the graph and replace any SSC with more than one node by a single placeholder node. Second, we sort the graph consisting of all the (placeholder) nodes and process them according to this order. If we encounter a placeholder, we perform the fixpoint computation of all the nodes which were originally replaced by the placeholder.

Algorithm 1: property graph to attack graph

```
   Input: property graph PG, initial assets attacker
   Output: attack graph AG
   // initialize AG
 1 Add to AG a single attacker node;
 2 Add to AG the initial assets attacker as nodes with an edge to the attacker node;
   // add compromised nodes
 3 for $r_i \in [B.1, B.2]$ do
 4     Add to AG the nodes in postcondition of $r_i$ to the nodes already in AG that are
         preconditions for $r_i$. The edge is labeled $r_i$;
 5 end
   // apply B.3 and B.4 until fixpoint is reached
 6 while fixpoint not reached do
 7     Add to AG the nodes in postcondition of B.3 to the nodes already in AG that are
         preconditions for B.3. The edge is labeled B.3;
 8     Add to AG the nodes in postcondition of B.4 to the nodes already in AG that are
         preconditions for B.4. The edge is labeled B.4;
 9 end
   // iteratively build graph
10 for $r_i \in [B.5, B.6, B.7, B.8, B.9, B.10, B.11, B.12, B.13,$
     $B.14, B.15, B.16, B.17, B.18, B.19, B.20]$ do
11     Add to AG the nodes in postcondition of $r_i$ to the nodes already in AG that are
         preconditions for $r_i$. The edge is labeled $r_i$;
12 end
   // add mitigations
13 mark all edges in AG as removable if a defender rule applies to it and they have no
     mitigation disabling dependency ;
14 for ca ∈{compromised CAs in AG} do
15     if ca not reachable for attacker then
16         mark all rules {B.9, B.15}, {B.12,B.14}, and {B.11,B.13} depending on ca as
             removable;
17 end
```

an edge in the attack graph.

For efficiency reasons, we apply mitigations in bulk, i.e., DNSSEC to all domains where it is both applicable and useful in removing edges. Let M be a set of mitigations (e.g., consisting of DNSSEC). To determine the cost and efficacy of applying M wherever possible, we query all edges in AG corresponding to rules which are disabled by an action in M and remove these edges. We say that a rule r is disabled by an action if the precondition of r includes the effect of m in negated form. We compute the cost of M by multiplying the number of domains for which we enabled DNSSEC with the cost of DNSSEC. The computation of the remaining attacker's successful paths is just a reachability query.

Using *transactions*, Neo4j permits us to store the unmodified attack graph, remove edges, and unroll this transaction later to reestablish the unmodified attack graph quickly.

The advantage of this approach is the fast computation of mitigation cost and attacker reward for a single set M. The downside is that for n classes of mitigations, we need to consider all 2^n combinations. This can be feasible for small n (e.g., in our case, $n = 9$). By contrast, classical Stackelberg planning computes the best options *for each host* instead of the best global policy, where, n additionally scales with the size of the attack graph.

The only mitigation-disabling predicate that cannot be statically computed, i.e., based on the property graph, is the compromise of a certificate, because any trusted Certificate Authority (CA) can issue a malicious certificate for a domain. As soon as the mitigations are known, however, the certificate compromise can be determined. In our graph, we have a limited number of CAs distributed over various countries, hence we can thus afford to compute all compromised CAs (for simplicity), and then determine which of the attacker rules B.9,B.15, B.12,B.14 or B.11,B.13 are being disabled. They are disabled (marked with the $\neg$-symbol in Figure 4.2) if the corresponding mitigation was selected and the 'cert compromised' predicate preventing the mitigation is not reachable for the attacker.

4.5 Threat Model

We focus on infrastructure attacks, *i.e.,* those that arise from physical, logical, and administrative dependencies in the Internet, as opposed to weaknesses in the protocol specification or in the implementation. We, therefore, assume that protocols and web mitigations achieve their stated goals, *e.g.,* provide a secure communication channel, but the attacker may break the trust assumptions, *e.g.,* when a *CA* is compromised.

Table 4.1: Attacker actions associated to class of attackers (nation-state (N), service providers (S), small hacker groups (H)), paraphrased (full definition in Appendix B), *Mitigation due to sneakiness assumption.

	#	attack vector	precondition	outcome	applicable mitigations	attacker class
initial compromise	(B.1)	attacker control	country compromised and entity (AS,IP,name server or CA) associated to this country	entity compromised	none	N
	(B.2)	attacker control	AS compromised and IP i belongs to AS	i compromised	none	N,S
	(B.3)	attacker control	IP i compromised and domain d resolves to i	d compromised	none	N,S,H
	(B.4)	attacker control	domain d compromised and d resolves to IP i	i compromised	none	N,S,H
routing	(B.7)	routing compromise	AS_2 potentially en route from AS_1 to AS_3 and AS_2 compromised	routing from AS_1 to AS_3 compromised	IPsec	N,S
	(B.8)	routing control	AS_1 compromised	routing from AS_1 to AS_2 compromised	none	N,S
DNS	(B.6)	DNS poisoning	name server d' queried when resolving d and d' compromised	resolution of d compromised	none	N,S,H
	(B.10)	DNS hijacking	name server d' queried when resolving d and d' in AS_2 and AS_1 geolocated in country and routing from AS_1 to AS_2 compromised	resolution of d from country compromised	DNSSEC on d'	N,S
certificate compr.	(B.16)	certificate spoofing	some CA is compromised	certificate of d can be forged	Certificate Transparency* (on d's CA); DANE (on d's authoritative NS)	N,S
	(B.17)	DANE record spoofing	some CA is compromised and d' authoritative for d and d' compromised	certificate of d can be forged	Certificate Transparency* (on d's CA)	N,S

	(B.18)	trust chain compromise	CA a is compromised and TLSA assumes trust in a	certificate of d can be forged	Certificate Transparency* (on d's CA)	N,S
content	(B.5)	XSS	XSS vulnerability on d	website on d compromised	none	N,S,H
	(B.9)	website MITM	Domain d resolves to IP in AS_2 and AS_1 geolocated in country and routing from AS_1 to AS_2 compromised	access to website on d from country compromised	HTTPS + HSTS + HTTPS-Redirect (unless certificate of d can be forged)	N,S
	(B.11)	from DNS poisoning	resolution of d compromised	website on d compromised	HTTPS + HSTS + HTTPS-Redirect (unless certificate of d can be forged)	N,S,H
	(B.12)	from DNS hijacking	resolution of d from country compromised	access to website on d from country compromised	HTTPS + HSTS + HTTPS-Redirect (unless certificate of d can be forged)	N,S
via CDNs/JS inclusion	(B.13)	from DNS poisoning	resolution of d' compromised and d includes JS from of d'	website on d compromised	SRI (res. from d'); secure incl. (res. from d') (unless cert. of d' can be forged); HTTPS + HTTPS-Redirect (unless cert. of d' can be forged); `upgrade-insecure-requests` on d (unless cert. of d' can be forged)	N,S,H
	(B.14)	from DNS hijacking	resolution of d' from country compromised and d includes JS from of d'	access to website on d from country compromised	SRI (res. from d'); secure incl. (res. from d') (unless cert. of d' can be forged); HTTPS + HTTPS-Redirect (unless cert. of d' can be forged); `upgrade-insecure-requests` on d (unless cert. of d' can be forged)	N,S

(B.15)	via routing	d includes JS from d' and AS$_1$ located in country and d' within AS$_2$ and routing from AS$_1$ to AS$_2$ compromised	access to website on d from country compromised	SRI (res. from d'); secure incl. (res. from d') (unless cert. of d' can be forged); HTTPS + HTTPS-Redirect (unless cert. of d' can be forged); `upgrade-insecure-requests` on d (unless cert. of d' can be forged)	N,S
(B.19)	third-party JS-inclusion	d includes from d' and d' is compromised	website on d compromised	SRI for resources from d'	N,S,H
(B.20)	website compromised	d is compromised	website on d compromised	none	N,S,H

Our threat model consists of a set of *attacker rules* that, given the higher number and complexity compared to the model in Chapter 3, we listed in a paraphrased form in Table 4.1 and formally defined in Appendix B. We define an entity (e.g. a NS, a route between ASes, etc.) in our model as compromised if the attacker can affect the integrity of the entity. For example, a route between two ASes is compromised if an attacker can pose as a MITM in the communication. Similarly, a NS is compromised if the attacker can tamper with the DNS response. These rules describe a layered model in which we depicted the different attacks that can be carried out for each layer: routing-level attacks can be used to compromise the integrity of packet transmission, DNS-level attacks can compromise the integrity of the name resolution and application-level attacks can compromise the content of the website. The combinations of the attacker's actions lead to a loss of confidentiality and integrity for the users visiting the websites. For example, an attacker that can perform a MITM attack can both actively inject/modify content (loss of integrity) or passively eavesdrop on the communication (loss of confidentiality).

4.5.1 Class of Attackers

We considered three classes of attackers with different capabilities: small cyber-criminal groups, malicious service providers, and nation-states. Each class has access to a given set of compromised entities, *e.g.,* ASes, websites, CAs, or NSs that translate into a subset of rules described in Table 4.1. Not all of the attack vectors are available to all classes of attackers as some are traits specific to particular attackers. In Table 4.1 we identified which classes hold the capability for each attack vector. This will be used in the analysis in Section 4.7. We underline that this assignment is not definitive as, e.g., small hacker groups can also compromise CAs, but our framework allows us to define different scenarios of adversaries targeting users of the web.

We evaluate the impact of an attacker in terms of the number of websites it can compromise, weighted by the number of visits to these web sites, i.e., the attacker plan maximizing $\sum_{i \in countries} Visits_{i,d}$ for $Visits_{i,d}$ being the estimated number of visits for the web site d from the Country i.[3]

By computing the maximum attacker reward, we can measure the potential impact of attacks on the Internet and the efficacy of the mitigations in scenarios characterized by the initial assets of the attacker and the set of rules available to the attacker.

For the class of attackers considered, stealthiness is of the utmost importance to avoid attribution and retaliation [104], in particular for service providers and countries. Therefore, for a first approximation, we ignored attacks that can be easily detected and that can result in global exposure to a company or country, *e.g.,* BGP hijacking attacks. Hence, our attacker is 'sneaky'. We discuss the limitations in Section 4.8.

4.5.2 Attacker Rules

The threat model is described in terms of attacker actions that are instances of the rules in Table 4.1. The predicates capture which entities (ASes, IPs, domains, CAs, NSs) exist, how they are related and which mitigations have been deployed, but also the state of the attack. The state of the attack is represented by the following predicates:

- An entity can be compromised globally, or for users from a country, in which case it can deliver malicious content.
- A route between two ASes can be compromised, in which case the attacker can inject/reroute packets on this route.
- The DNS resolution of a domain can be compromised (for all users or for users from a country), in which case the attacker can manipulate DNS queries for this domain.

The complete model and the list of predicates are presented in Appendix B along with the intuition for each rule. Here, we only consider an example for illustration. Say we consider China as an attacker mounting a Great Cannon-like attack, *i.e.,* Chinese authorities intercept requests to included JavaScript resources and modify their con-

[3]We use the number of visits per month as retrieved from Alexa, thus we are assuming the attacks to be stealthy and to persist for some time. Furthermore, we ignore countries that constitute less than 0.01% of the website's visitors.

tent [211]. Suppose users visit the popular website `www.diply.com`. By rule B.1, China controls the `AS714`[4], over which packets from Japanese users may be routed when contacting `a10-67.akam.net`, which is in `AS21342`. By rule B.7, this route is compromised. `a10-67.akam.net` is the authoritative NS for `cdn.diply.com`, the resolution of this domain is considered compromised by rule B.10. As `www.diply.com` includes JavaScript code from `cdn.diply.com`, this website is vulnerable to JavaScript injection after performing a DNS poisoning for `cdn.diply.com` (rule B.14). The injected JS can force visitors to perform a DDoS attack against target websites [211].

4.6 Mitigations

The defender model consists of a set of actions that aim to minimize the attacker's reward by implementing a set of mitigations. Each defender action has an associated cost and mitigates one or more attack vectors as summarized in Table 4.1. A mitigation can present some preconditions to be met before the deployment (*e.g.,* DANE requires DNSSEC, Certificate Transparency requires the presence of HTTPS, etc.). In Appendix B we formally defined the preconditions for each mitigation. In our analysis we allow a mitigation to be deployed only if its preconditions hold.

We now present the mitigations that can be applied at different levels of the Internet infrastructure.

4.6.1 Application Layer Mitigations

SRI CDNs are a major target for attackers, as thousands of websites often depend on a particular resource they host, *e.g.,* widely used libraries like jQuery. A modification of this resource can infect the users of the including website. With Sub Resource Integrity (SRI), the including website provides the hash value of each resource hosted on a third-party server with the script tag. The browser compares this hash value to the hash value of the retrieved file. If the values do not match, the browser does not execute the resource.

This type of attack is widespread and can be implemented in large scale as shown by the *Great Cannon* attack [28, 211]. The implementation of SRI for the resources retrieved from Baidu could have reduced the impact of this attack [333].

Although the adoption of SRI is growing [81], it is not suited for resources that change over time, *e.g.,* versioned JS libraries, or dynamically generated scripts.[5] This scenario

[4]Some prefixes are partially used at this location.

[5]To temporarily handle this mismatch, the external resource can be retrieved from a local repository.

is not uncommon, however, it is often caused by minor changes (e.g. recompilation) that can be easily avoided [312].

Other mitigations Another mitigation could be Content Security Policies (CSP) [309]. For a first approximation, we decided to not consider CSP for mitigating XSS in our model because the adoption is currently strongly limited by the required cooperation with third-parties [312], with the result that most of the deployments are insecure and enable inline scripts [73, 355, 357]. Given that CSP is mainly used to prevent inline XSS [120, 357] and does not prevent other attack vectors available for our classes of attackers (e.g. compromise of whitelisted CDNs), we are confident that CSP would not affect the overall results. We discuss the extension of the model in Section 4.10.

4.6.2 Transport Layer Mitigations

TLS The HTTP connection between a client and a website can be secured through TLS to achieve authenticity, integrity, and confidentiality. At the time of writing, HTTP is the default protocol in almost[6] all major browsers. As we assume users to not specifically ask for HTTP over TLS (HTTPS) connections, websites need to implement a redirect and set an appropriate HSTS header (see below) for this mitigation to be effective.

Redirects and HSTS While a secure redirect is not a mitigation in itself, it is necessary to provide a secure connection for the exchange of an HSTS policy through the `strict--transport-security` header. Indeed the header is ignored in an HTTP connection [154]. To ensure that any further access to the server is directly conducted over HTTPS, it is necessary to implement an HSTS policy.[7] An HSTS policy is an HTTP header that informs the browser that the specific domain and (if explicitly declared) its subdomains must be accessed via HTTPS for a certain period of time. All major browsers come with an *HSTS preload* list that contains a set of domains for which the browser automatically creates an HTTPS connection. However, it is required to keep a HSTS header to maintain the domain in the preload list.[8]

[6]Only the most recent version of Chrome [55] and Firefox in private mode [56] use HTTPS by default. Safari defaults to HTTP.

[7]Under rare circumstances, a redirect can increase security by itself: if a network injection attack is possible on an included resource, a redirect ensures that the malleable resource is not loaded because it would constitute mixed content (see Rule B.15).

[8]`https://hstspreload.org/#continued-requirements`

Secure inclusions and CSP upgrade-insecure-requests directive To secure inclusions from third-party websites, subresources should be loaded through a secure connection, either explicitly specifying the HTTPS protocol or using a Content Security Policy with an `upgrade-insecure-requests` directive. The latter informs the browser that all the site's insecure URLs must be replaced with HTTPS.

An attacker can exploit subresources retrieved through HTTP by conducting a MITM attack. This scenario is limited to the case in which the main web page is loaded over HTTP as currently, modern browsers block *mixed content* for active resources. We stress that different browsers handle mixed content differently, and outdated browsers might still be vulnerable to this attack. We reserve a closer look at how legacy browsers change the picture for future work and assume all browsers to block active mixed content.

4.6.3 Routing Layer Mitigations

IPsec To prevent attacks at the network layer from a malicious AS in the path between two ASes, packets routed between the two ASes can be encrypted and authenticated through a transport-level gateway-to-gateway tunnel. Various technologies provide this functionality, but to provide a concrete cost estimate [80], we chose IPsec with a gateway-to-gateway architecture [157, 177]. We assume the implementation of an IPsec connection to not be influenced by the geolocation of the endpoint and be the result of a private agreement between AS owners.

4.6.4 Resolution Mitigations

DNSSEC To prevent DNS spoofing attacks, DNS records can be authenticated with DNSSEC [35]. The adoption by end users is still very low [33], but it can be implemented in the recursive resolver of the ISP [229]. We assume this and that the route from the user to the recursive DNS resolver is secure. The latter assumption is necessary, as we do not have data on how the visitors reach their recursive resolver and the opposite assumption would render DNSSEC useless. As we will see (Section 4.7), DNSSEC achieves modest security improvements despite this over-approximation.

We consider this mitigation for all domains where all the parent domains up to the root already support DNSSEC. At the time of writing, DNSSEC is deployed in the root servers and more than 90% of the TLDs [164].

4.6.5 CA Mitigations

Certificate Transparency The authenticity of a web server on the Internet relies on digital certificates issued by certificate authorities. In the last years, this model showed many flaws including mistakenly issued certificates and CA compromise. Google proposed the Certificate Transparency (CT) [190] project as a measure to detect misissued certificates; this is done through a set of publicly available append-only certificate logs that contain all the certificates present on the Internet. CAs must submit the certificate to a log to receive a signed certificate timestamp required by the browser during the TLS handshake. Domain owners can verify the list of digital certificates issued for their domains and detect the presence of unauthorized ones. Chrome requires all certificates issued after 30 April 2018 to be compliant with the CT policy and Safari requires signed certificates timestamps. Given that Chrome and Safari alone cover more than 78% of the desktop browser market share [310], and that Mozilla is planning to include support for the CT project [217], we assume the entire CA ecosystem to be CT compliant. Given that our threat model considers stealthy attacks, we ignored the scenario in which an attacker issues a malicious certificate via a compromised CA. An attacker could potentially hide the forgery by controlling both a CA and a CT log. Currently, browsers control CT logs from different operators thus making the attack feasible only by controlling logs from different operators. For a first approximation, we ignored this scenario. Furthermore, distributed audit mechanisms like LogPicker [101] can prevent this attack. Nevertheless, we investigate the effect of DANE as an alternative to CT in a separate scenario.

Other mitigations DNS-Based Authentication of Named Entities (DANE) [155] is a DNS-level mitigation against vulnerabilities in the CA model [245]. DANE allows a client to retrieve an end-entity certificate or a certificate to be found in the path to (including) the trust anchor through DNS queries. This mitigation requires DNSSEC to be deployed. Depending on the implementation, DANE can amend or side step the CA model. We consider the case where DANE defines the website's current end-entity certificate in its record.

Although the adoption of DANE for email servers is growing, some challenges prevent the adoption for the Web PKI [194]. As of now, all major browsers do not automatically validate DNSSEC and DANE. We nevertheless assumed that this feature is implemented and evaluated its effect as an alternative to CT in case of the presence of a non-stealthy attacker in Section 4.7.

Other mitigations like the DNS Certificate Authority Authorization (CAA) allow domain owners to specify via CAA records the CAs that are allowed to issue certificates for the domain. However, this does not prevent a malicious CA to issue certificates [149]. We thus ignored this mitigation.

4.6.6 Mitigation Cost

Table 4.2: Mitigation cost per host. Let $r = 140\,\$/\text{h}$ the daily rate of an external consultant.

Mitigation	Cost per host	Comment
SRI	r*8h	Consultant cost for 1 day. Exists tools to support (*e.g.,* [170]). We do not consider any backup cost to handle mismatches of hashes. Although SRI requires CORS to be enabled for included resources, the cost for setting up this header is negligible , since it requires a single HTTP header to be set [312].
TLS	r*8h	Consultant cost for 1 day to modifying the web server configuration to allow HTTPS connections (including the effort of obtaining a certificate). We do not include the cost of the digital certificate, given free CAs like *Let's Encrypt*.
Redirect / HSTS	r*8h	Consultant cost for 1 day.
Secure inclusions / UR	r*8h	Consultant cost for 1 day to check that all subresources are available via HTTPS.
IPsec	$56,000 per link	Cost for a link speed of 10 Gb/s. Including the cost of two dedicated routers for $24,000 each [263] and the consultant cost for configuration and maintenance per year (about 80 consulting hours) [80].
DNSSEC	$366,342	Cost of deploying in all the authoritative NS managed by a company based on the maximum CAPEX from a survey [229], one of which appear in the Alex Top 100.
CT	$0	The CA ecosystem is already CT compliant and mitigation can only be applied if TLS is already deployed.
DANE	$4,000	Cost of creating `TLSA` record for the certificate, similar to [308]. Exists tools to automatically generate TLSA records (*https://ssl-tools.net/tlsa-generator*).

We gather the cost based on publicly available data and try to keep the cost model uniform, i.e., we do not take differences in cost of labor due to the location or structure of the company into account. For example, `youtube.com` and `google.com` belong to the same company, but only recently announced they will share infrastructure [12]. We stress that a uniform cost model, while being easy to convey, can never exactly represent the actual operating cost in such a diverse set of companies as the Alexa Top 5k. Moreover, what to include as a direct cost of a technology like DNSSEC is very much debatable. We focus on the immediate cost of mitigations and convert all personnel cost from time estimates into \$ by considering the cost for an external consultant[9] of $r = 140\,\$/\text{h}$. Table 4.2 lists the cost estimates. These values will be used to compute Pareto optimal defenses in Section 4.7. The investigation of optimal mitigation deployments with custom costs can be performed on our website `mitigation-web.github.io`.

4.7 Evaluation of Web Infrastructure Mitigations

We evaluated attacks on the Web carried by the different classes of attackers (a cyber-criminal group, large infrastructure providers offering, e.g., cloud services or name resolution, and nation-state groups) and the impact that the mitigations have in securing visitors on the Web. We model the purported threat by defining the set of assets initially under attacker control (B.1-B.3 in Table 4.1).

For each attack scenario, we generated the attack graph. We then ran the analysis on every combination of the mitigation strategies introduced in Section 4.6. We computed from this: the impact of the attacker, in terms of % of visits in the Top 5k that can be affected by the attack vectors in the status quo, the *current efficacy*, in terms of % of visits in the Top 5k protected by the mitigations *currently* deployed, the *potential efficacy*, in terms of % of visits that could be protected by the application of different mitigations strategies (without breaking websites' functionality) globally on the Web, and the cost of applying these strategies to the status quo. In Tables 4.3 and 4.4, we report this data for all sets of mitigation strategies we deem interesting – the totality of all 512 combinations would exhaust the space available here. For each scenario, we combined the set of combinations in a Pareto frontier. The frontiers are a visualization of all Pareto optimal combinations of mitigation strategy costs on the x-axis and the percentage of still affected visitors on the y-axis. We removed each combination that is dominated by others and plotted the remaining ones on a graph mapping cost to potential efficacy.

All these computations, including the Pareto frontiers, can be interactively explored at mitigation-web.github.io.

The resulting Pareto frontiers depend on the costs discussed in Section 4.6.6. Despite

[9]Hourly rate (US) for a Computer Security System Specialist level 2 via Deloitte, Ltd [95].

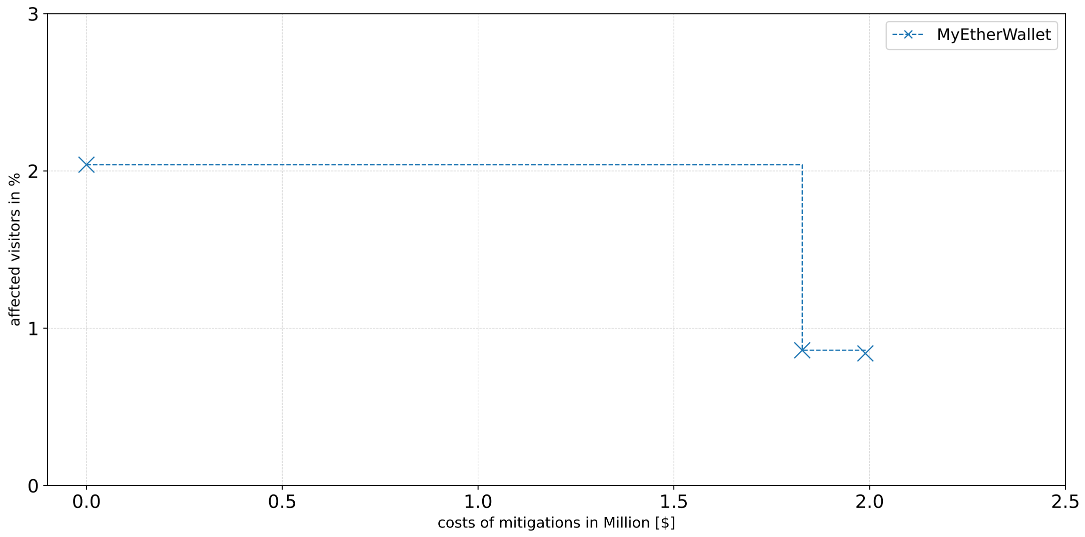
our best efforts to justify our cost assumptions, the empirical data available is incomplete and what needs to be taken into account is debatable. However, we can precompute each countermeasure's effect while also counting how often it is applied. The overall cost is the sum of these counts weighted by their cost and can be computed on the fly. The computation of the Pareto frontier is linear in the list of combinations once they are sorted by their cost. Stakeholders can modify the cost assigned to all countermeasures or determine the interval of costs in which the Pareto frontier does (not) change.

4.7.1 Cyber-criminal Group

In this threat scenario, shown in the leftmost column of Table 4.4, we consider a hacker group that can compromise NS resolutions and exploit XSS vulnerabilities, the most widespread type of vulnerabilities in web applications according to the OWASP foundation (see Figure 4.3 for the Pareto frontier). We identified a subset of domains with XSS vulnerabilities from Steffens et al. [313]. We took inspiration from the MyEtherWallet attack on the 24th of April 2018 [326] to evaluate the impact of an attack performed by cyber-criminal groups on the Web infrastructure. The original attack started by hijacking Amazon's *Route 53* name servers via BGP. The attackers rerouted requests to this name server to a malicious server that referred the users to a phishing website imitating MyEtherWallet. While our model excludes BGP hijacking as an attack vector due to the possible global exposure, we instead model the situation where the hacker group compromised the name servers directly. The initial asset is thus composed of more than 1500 Amazon's *Route 53* NS. With 2% of all page views on the Alexa Top 5k, the impact is already considerable. The attacker compromises the DNS resolution for a set of CDNs, like `cdn-1.tstatic.net` and `content.jwplatform.com`. This approach allows the attacker to compromise all websites that rely on these CDNs for the inclusion of JS resources. Furthermore, the Amazon *Route 53* DNS servers are queried for the DNS resolution of many domains, such as `reddit.com`, `twitter.com` or `dropbox.com`. IPsec, DNSSEC, and DANE have no effect because we assumed the DNS servers themselves to be compromised.[10]

The most effective countermeasure is to employ secure connections both for the website via HTTPS, HTTPS-Redirect and HSTS (abbreviated H3 in the following) and the external JS inclusions. Combining H3 with `upgrade-insecure-requests` (abbreviated UR in the following), we achieve the maximum increase in security. This matches the

[10]For the actual BGP-based attack, the attacker cannot sign in the nameserver's stead, hence DNSSEC/DANE could appear as a possible mitigation in the high-cost range of the Pareto frontier



Compromise of Amazon's route 53 NS affects 2% of the Top 5k Alexa visit. The application of HTTPS, Redirect, HSTS, and SRI reduces the affected visits below 1% with a minimal cost.

Figure 4.3: Pareto frontier for small hacker group attacker scenario

application-level countermeasures proposed by the developers in the aftermath [326]. In summary, enforcing endpoint-level defense is the optimal solution for this threat.

4.7.2 Infrastructure Providers

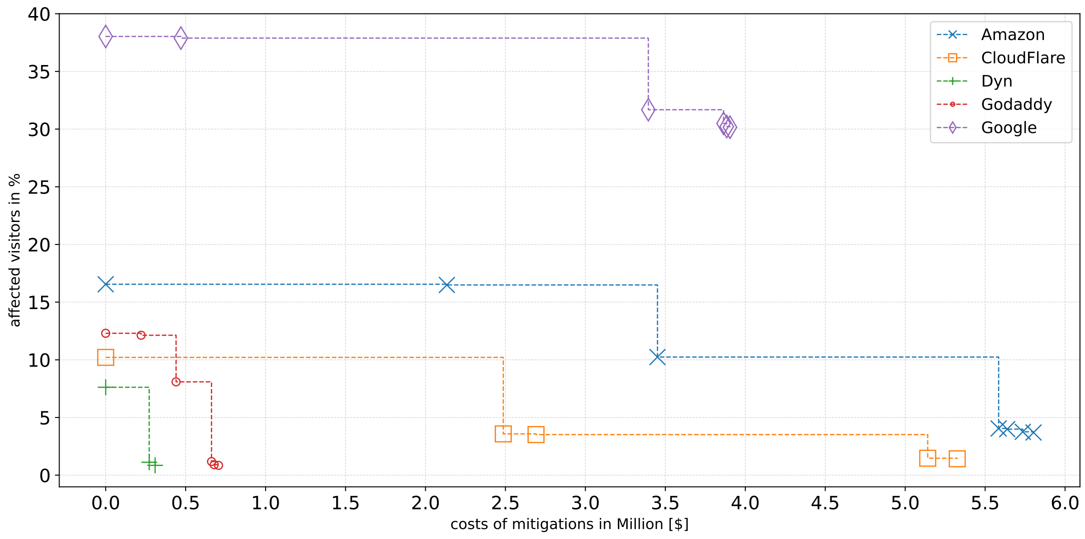
Next, we analyze the potential threat that the centralization of infrastructure in the Internet can pose to users in case an adversary gains control over them, and how to mitigate a potential attack. We choose some of the biggest infrastructure providers: Google, as a large provider of JS resources; Cloudflare and Amazon as two of the largest CDNs; Dyn, as one of the largest providers of DNS services and GoDaddy, as the largest domain registrar. The overall attacker success for each of the companies is shown in Table 4.3. Figure 4.4 shows the Pareto frontier for each company. The initial asset is generated starting from the owned ASes and CAs (if any) for each infrastructure provider. Amazon, Google, and GoDaddy control certification authorities that are accepted trust anchors in all major browsers. While CloudFlare and Dyn control ASes, but no CA. By propagating the rules B.2,B.3,B.4 we compute the entire asset for the attack scenario. For example for Dyn, we further add its 175 NSs serving about 576 domains in our dataset. In Table 4.3, Google is the strongest player, affecting about 38% of the page views on the Top 5K Alexa domains. While Dyn only affects roughly 8% of the visits. In terms of attack vectors, we observed that Google has a great impact on routing. As expected, Amazon is able to compromise 3rd party resources either directly or via DNS spoofing. CloudFlare's

Table 4.3: Percentage of affected visits, protected visits, and potentially protected visits and cost for infrastructure adversaries attacking the Alexa Top 5K. H3 is short for HTTPS, HTTPS-Redirect and HSTS, UR is short for CSP's `upgrade-insecure-requests`.

	companies (as attackers)									
Metric	Google		Amazon		GoDaddy		CloudFlare		Dyn	
Affected visits in status quo	38.03%		16.55%		12.3%		10.21%		7.62%	
Current efficacy in status quo (% of visits protected by the mitigations currently deployed)										
H3	0.05 %		0.05 %		0.05 %		3.15 %		4.46 %	
H3, SRI	0.05 %		0.08 %		0.05 %		3.15 %		4.46 %	
H3, UR	0.05 %		0.05 %		0.05 %		3.15 %		4.46 %	
CT	0.05 %		0.05 %		0.05 %		0.00 %		0.00 %	
CT, H3	0.05 %		0.05 %		0.05 %		3.15 %		4.46 %	
Potential efficacy (% of visits that could be protected by the mitigations) and deployment cost in $1000										
IPsec	0.00 %	2,072 k$	0.00 %	0 k$	0.00 %	4,424 k$	0.00 %	5,992 k$	0.00 %	0 k$
DNSSEC	0.00 %	39,931 k$	0.00 %	40,297 k$	0.00 %	41,030 k$	0.00 %	40,297 k$	0.00 %	39,931 k$
DANE	0.00 %	740 k$	0.00 %	740 k$	0.00 %	740 k$	0.00 %	0 k$	0.00 %	0 k$
SRI	6.35 %	3,393 k$	6.30 %	3,450 k$	4.21 %	440 k$	1.85 %	2,654 k$	0.00 %	42 k$
Sec. Incl.	0.00 %	10 k$	0.00 %	14 k$	0.00 %	3 k$	0.00 %	64 k$	0.00 %	38 k$
UR	0.00 %	5 k$	0.00 %	7 k$	0.00 %	3 k$	0.00 %	45 k$	0.00 %	11 k$
H3	0.15 %	470 k$	0.05 %	2,133 k$	0.17 %	221 k$	6.63 %	2,486 k$	6.49 %	272 k$
H3, CT	2.42 %	7,909 k$	9.01 %	8,880 k$	11.11 %	1,196 k$	6.63 %	2,486 k$	6.49 %	272 k$
H3, CT, SRI	7.60 %	3,884 k$	12.56 %	5,638 k$	11.18 %	685 k$	8.73 %	5,140 k$	6.49 %	314 k$
H3, CT, UR	2.69 %	8,430 k$	9.30 %	9,424 k$	11.40 %	1,263 k$	6.69 %	2,691 k$	6.76 %	309 k$
H3,CT,SRI,UR	7.87 %	3,904 k$	12.84 %	5,802 k$	11.44 %	705 k$	8.79 %	5,325 k$	6.76 %	328 k$

major attack vectors are routing attacks and content compromise. Similarly, GoDaddy can exploit routing attacks on JS inclusions and name resolution, while Dyn's major attack vector relies on DNS poisoning. The efficacy of currently deployed mitigations on securing visits is marginal (0.05% for Google and Amazon), showing that the current deployment is insufficient. However, if we apply a set of defenses globally, the potentially protected visits on the Top 5K increases to almost 8% for Google and 13% for Amazon.

Across the board (Table 4.3), we see that the deployment of lower-layer mitigations like IPsec, DNSSEC or DANE would add no additional security by themselves, even though they are often applicable, which is indicated by non-zero cost values. For Google, Amazon, and GoDaddy, we see that SRI has a tremendous effect, as those host or control access to popular JS libraries, e.g., jQuery. This effect is less pronounced for Cloudflare and zero for Dyn, as these exert less control via JS inclusion and, specifically for Dyn, HTTPS is already protecting many connections. H3+CT gives an inverse picture; the effect on Google is much weaker than SRI. It is important to underline that H3 deploys HTTPS, but does not assume CT compliance. As now all CAs support CT, H3+CT is



Goole affects 38% of the page views on the Top 5k Alexa, while Dyn, Cloudflare, GoDaddy, and Amazon range between 7% and 16%. By applying a combination of HTTPs, Redirect, HSTS, SRI and upgrade request the impact is reduced by 7%, 12%, 11%, 8%, and 6% for Google, Amazon, GoDaddy, Cloudflare, and Dyn respectively. IPsec, DNSSEC, and DANE are ineffective.

Figure 4.4: Pareto frontiers for infrastructure attacker scenarios

the more realistic mitigation, deploying CT at zero cost. As expected, H3 has only a very small impact in scenarios where the attacker controls a CA, highlighting the continued benefit of CT. We observed that H3 is always the first point in the Pareto frontier, confirming the intuition that securing access to the first party comes at a lower cost than protecting against JS inclusions. Securing third-party resources always achieves a stark improvement in security when combined with H3+CT, but comes at a high cost. Whether SRI, Secure inclusions or CSPs `upgrade-insecure-requests` are cost efficient depends on how websites include third-party resources and whether they are controlled by the attacker. For Dyn and CloudFlare, UR is the best choice, as the direct compromise of the third-party is less of an issue. By contrast, SRI *by itself* appears in the Pareto frontiers for Google, Amazon, and GoDaddy due to their direct control of CDNs. In all cases but Dyn, combining H3, SRI and UR achieves an increase over only H3 and SRI. This is because UR enables the deployment of H3 on domains with insecure JS inclusions, where a secure redirect would otherwise break functionality.

4.7.3 Nation-State Groups

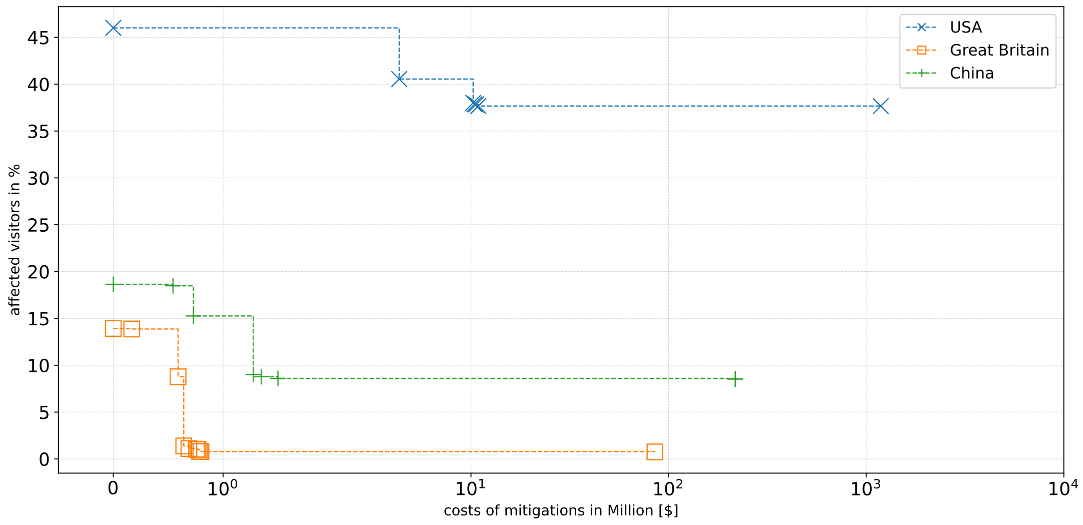
In this scenario (Table 4.4, right-side columns), we consider the potential of three states to mount an attack, assuming that local legislation permits such an attack (see Figure 4.5

Table 4.4: Percentage of affected visits, protected visits, and potentially protected visits and cost for hacker group and nation-state adversaries attacking the Alexa Top 5K. H3 is short for HTTPS, HTTPS-Redirect and HSTS, UR is short for CSP's `upgrade-insecure-requests`.

	hacker group		countries (as attackers)					
Metric	MyEtherWallet		US		CN		GB	
Affected visits in status quo	2.04%		46.01%		18.64%		13.93%	
Current efficacy in status quo (% of visits protected by the mitigations currently deployed)								
H3	1.55 %		0.00 %		0.26 %		0.00 %	
H3, SRI	1.55 %		0.00 %		0.26 %		0.00 %	
H3, UR	1.55 %		0.00 %		0.26 %		0.00 %	
CT	0.00 %		0.00 %		0.26 %		0.00 %	
CT, H3	1.55 %		0.00 %		0.26 %		0.00 %	
Potential efficacy (% of visits that could be protected by the mitigations) and deployment cost in $1000								
IPsec	0.00 %	0 k$	1.45 %	1,174,600 k$	9.43 %	216,104 k$	11.92 %	84,504 k$
DNSSEC	0.00 %	0 k$	0.01 %	52,386 k$	0.00 %	37,000 k$	0.00 %	12,455 k$
DANE	0.00 %	0 k$	0.00 %	740 k$	0.00 %	740 k$	0.00 %	740 k$
SRI	0.00 %	69 k$	5.46 %	4,331 k$	3.38 %	730 k$	5.15 %	590 k$
Sec. Incl.	0.00 %	67 k$	0.00 %	53 k$	0.00 %	23 k$	0.00 %	3 k$
UR	0.00 %	24 k$	0.00 %	35 k$	0.00 %	4 k$	0.00 %	3 k$
H3	1.18 %	1,827 k$	0.04 %	5,923 k$	0.15 %	544 k$	0.05 %	168 k$
H3, CT	1.18 %	1,827 k$	1.62 %	11,538 k$	9.53 %	1,515 k$	12.53 %	641 k$
H3, CT, SRI	1.18 %	1,897 k$	8.15 %	10,419 k$	9.81 %	1,398 k$	12.91 %	771 k$
H3, CT, UR	1.20 %	1,989 k$	1.64 %	12,280 k$	9.79 %	1,666 k$	12.82 %	688 k$
H3, CT, SRI, UR	1.20 %	2,014 k$	8.34 %	10,889 k$	10.04 %	1,499 k$	13.14 %	798 k$

for the Pareto frontier for each country). The Great Cannon attack, e.g., is believed to have been mounted from China. The initial asset is obtained starting from rule B.1 by including the NSs, ASes, domains, and IPs located in that country.

The US is the country with the highest attack potential: about 46% of the visits on the Top5K are affected. By applying different mitigations, this reward can only be reduced to 38%. Due to the importance of domains under US jurisdiction, many page views would be directly compromised. The potential impact of China and GB is much smaller. Where GB's attack potential can be reduced from about 14% to about 1%, China's attack potential can only be reduced from 19% to 9%, which can be explained by the relative autonomy of the Chinese Internet infrastructure. However, the single most effective mitigation is IPsec, being nearly as effective as the combination of H3, CT, SRI and, with marginal impact, UR. These mitigations are protecting foreign websites that



The US can impact up to 46% of the visits in the Top 5K Alexa. Nation-state attacks potential can be reduced by combining application-level mitigations like SRI, CSP upgrade requests, HTTPS, HSTS with low-level protection like IPsec. This is particularly effective for GB where the affected visits reduce from 14% to 1%.

Figure 4.5: Pareto frontiers for state-controlled attacker scenarios

rely on Chinese infrastructure for routing, resolution, or content distribution, but not Chinese websites. GB has an influence on foreign pages as well, but a larger share of them is able to deploy helpful countermeasures. In particular, GB is the best scenario to demonstrate the viability of IPsec, single-handedly reducing the attacker's success from 14% to 2%. This is likely because of the GB's access to transatlantic submarine cables. By contrast, infrastructure that is routed via the US and China is often situated in the same country, due to their size relative to their neighbors and China's stated goal of self-reliance.

From Table 4.4, SRI and then H3+SRI are the cheapest mitigation for the US, it is H3 and then SRI for China and GB. In all three cases, the optimal countermeasure is SRI, UR, H3, CT, and IPsec (USA, CN) or SRI, H3, and IPsec (GB).

DANE vs. Certificate Transparency To evaluate DANE, which proactively mitigates certificate forgeries, we considered a scenario where we artificially removed CT. This has the same effect as forgoing the sneakiness assumption concerning after-the-fact detection of certificate forgeries.

Note first that DNSSEC is a prerequisite to DANE and recall that it is not applicable on all hosts. We find that the improvement in deploying DANE (asserting the current end-entity certificate) in addition to DNSSEC is zero in all scenarios. The reason is as follows: DANE is only effective if, in addition to the domain and its NSs, all CDNs that provide JS inclusions deploy DNSSEC. The majority of JS providers do not. We inspected the remaining cases and, while DANE thwarts attacks based on certificate compromise, other attacks (mostly on JS inclusions) still apply. Even applying H3 where possible, the improvement from adding DANE remains zero.

4.7.4 Discussion

Overall, we find that the influence of the biggest players on the market, in particular Google, is significant and comes close to the adversarial capabilities of a state-sponsored attacker. At the same time, we find that regardless of the type of attacker we consider, securing the most popular domains can be primarily achieved by deploying endpoint mitigations such as HTTPS, HSTS, and SRI. This is sometimes augmented by the use of UR. Additionally, IPsec plays a significant role in securing against the countries but not against the service providers.

Moreover, deploying these comparatively cheap endpoint mitigations allows quartering the user's exposure against infrastructure attackers with a cost of less than $6 M. On average, this amounts to about $1,000 per domain. The exception is the Google scenario, where such a decrease is not possible. Likewise, there is little defense against the US.

Despite the sneakiness assumption, DNSSEC achieves little at a high cost. Even though theoretically, the amortized cost could make these countermeasures a viable alternative considering the number of domains, this is not the case. On the other hand, our analysis has indicated that IPSec is effective, although expensive, mitigation against China and GB.

4.7.5 Performance

The graph-based analysis algorithm discussed in Section 4.4 reduces the analysis effort for a given mitigation by precomputing the attack graph from the property graph. The runtime of this precomputation step (called 'attack graph generation' in Table 4.5) depends on the scenario of choice, ranging from about one minute for Dyn to 3 h for the US, the country with the largest attack graph. The size of the generated attack graph governs the time the status quo analysis takes, as it is a simple reachability query. Neo4j is optimized for such queries, hence, even for the US attack graph that contains about two million edges, the analysis takes less than a minute. This makes it feasible to analyze different mitigation scenarios (which remove edges from the attack graph) and analyze the efficacy of existing mitigations (which adds edges to the attack graph). We combine

Table 4.5: Performances for Alexa Top 5k. The property graph used in these scenarios has 70,975 nodes and 329,899 edges.

	attack graph generation			status quo (s)	applying mitigation (s)			current efficacy (s)		
scenario	runtime(s)	# nodes	# edges		min	med	max	min	med	max
US	9843.02	129,371	2,191,112	49.05	59.58	108.98	284.75	94.27	405.65	585.78
CN	558.57	43,812	252,343	6.36	9.87	24.85	35.66	10.91	52.13	83.06
GB	215.16	28,626	50,948	1.55	3.77	14.40	23.71	1.81	60.98	92.45
MyEtherWallet	48.36	12,568	26,010	0.28	1.10	9.96	19.27	0.24	166.70	243.98
Google	125.16	31,598	73,148	1.75	5.15	15.75	24.61	3.08	91.05	143.70
Amazon	979.73	62,131	167,317	8.45	14.16	24.33	34.83	11.60	76.74	115.64
Godaddy	111.44	27,652	33,770	0.89	2.92	13.65	22.20	1.30	40.88	75.76
CloudFlare	345.74	18,293	35,215	0.74	2.09	10.55	21.41	0.70	94.17	143.30
Dyn	33.15	5,152	5,387	0.1	1.43	9.79	18.72	0.08	77.85	116.25

the time to remove or add edges with the runtime of the reachability query and report the minimum, median and maximum. As expected, there is quite a range: queries that modify the graph are more expensive than reachability queries. Hence, the more edges a mitigation removes, the higher the runtime. Half the queries in the largest attack graph take less than two minutes. Computing the data needed for mitigation-web.github.io, i.e., generating the attack graphs and computing the potential and efficacy for all 256 combinations of 8 mitigations, took about 52 h in total. All computations were performed on an Intel Xeon E5-4650L @ 2.60GHz. Because a Neo4j Cypher query is always computed in a single thread, we only made use of one CPU core. Further, 32 Gb RAM was sufficient.

4.8 Limitations

As our analysis measures the efficacy of mitigations in terms of adversarial success, it needs to be as precise as possible, ideally capturing all attacks, and only those attacks. Dax and Künnemann outline how to establish soundness and completeness w.r.t. a Dolev-Yao attacker interacting with the protocol *according to specification* [94], but we consider this out of the scope and take this attacker model as granted.

Moreover, their results suggest a tight relation between rules in the attacker model and protocol-level security properties. On the one hand, this gives guidance for the formulation of new attack vectors. On the other hand, precisely describing protocol-level security properties is known to be difficult and often done in conjunction with verification. Like the Dolev-Yao model, our model assumes the absence of implementation-specific errors induced by the user, which could at best be estimated at this point.

In terms of ecological validity, our model must be evaluated on the overall results i.e. if it actually describes the reality of securing the Internet infrastructure. This is particularly challenging given that we aim to determine mitigations that should be implemented considering what-if scenarios where data about attacks are not (widely) available. We tried when possible, e.g. in the evaluation of the cyber-criminal group scenario, to compare our results with known deployed mitigations for similar situations. We leave for future work the investigation of more systematic approaches.

We weigh the domains by the number of visits to reflect their popularity. This is not a measure of the number of users that can potentially be infected, as the reward is additive and thus counts visitors that frequent two domains twice. Some domains likely share more users with each other (*thehackernews.com* and *wired.com*) than others (*google.com* and *bing.com*). To compute the number of infected users, we would need information about the intersection of visitors, ideally for all sets of Alexa-listed domains.

If the attacker has access to one of the NS potentially queried in the name resolution of a domain, the integrity of the resolution is considered compromised. As there can be more than one authoritative NS per domain and caching may prevent the iterative resolution, this is an over-approximation. Similarly, we consider a route between two ASes compromised if either of the endpoints is compromised, or if a compromised AS is potentially en route. Additional inaccuracy is introduced by the fact that routes change over time.

We assumed the attacker wants to avoid global exposure due to the forensic evidence. As a result, we consider attacks against the PKI as mitigated if the target domain's certificate was signed by a CA compliant with Certificate Transparency. We showed in Section 4.7 that this assumption does not impact the results by analyzing the case in which CT is disabled and DANE is applied instead. Similarly, we exclude BGP hijacking and attacks on the DNS root servers. For BGP hijacking, similar results can be obtained by considering attacks at the network layer. Furthermore, it would require assumptions on how the (sub-)prefix hijacking and the BGP routes propagate. We leave the implementation of these attacks for future works.

The threat model focuses on attacks that can lead an attacker to compromise the content of a web site as a result of physical and logical dependencies. Our model can be extended to describe additional web attacks, e.g. vulnerable libraries or server misconfigurations, that produce a *direct compromise* of the content of a web site with a structure similar to rule B.5 for XSS without impacting the methodology discussed in Section 4.4. We leave for future works this extension.

We *over-approximate* the efficacy of browser-level mitigations by assuming all users to use current browser versions. Our results thus have to be read either as a projection to the future (about the potential of these mitigation techniques) or as a security analysis for the share of users with recent browsers. This limitation can be overcome by determining which browser versions implement which mitigation and using per-website data on browser usage. We *approximate* the cost of mitigations by considering a uniform set of rules and assuming similar cost of labor in the web security sector. All our estimates consider only direct costs.

4.9 Related Work

The vulnerability of the Internet at the infrastructure level has been studied before [70, 141], including the European BGP topology [127], and web attacks [300], but the analysis of mitigations has been largely ignored. An exception is the analysis of the email infrastructure by Speicher et.al. [308]. They compare mitigations in the email setting and consider countries as defenders and attackers. A major difference is that most email communication is captured by considering all pairs from a small number of providers, which results in a drastically smaller problem size.

We lack analysis of the effectiveness of mitigation strategies to protect the Web at the different layers of the Internet.

Our analysis follows the Stackelberg planning methodology [306], which was originally proposed for mitigation analysis in simulated pentesting. This discipline is closely related to attack graphs, which were first introduced by Philipps and Swiler [257]. Like planning, attack graphs describe an attack (a plan) as a combination of atomic components (actions). Both aim at understanding threats that arise as combinations of atomic actions. There are many flavors of attack graphs, including the *monotonic* formulation, where only positive preconditions and postconditions are permitted [26, 135, 168, 178, 235, 246, 327]. The attacker task in our case is monotonic as well, it keeps gaining new assets, but never loses any assets during the attack.

In terms of formal mitigation analysis, our setting relates to game-theoretic security models, specifically to Stackelberg competitions, where the game consists of a single exchange of move and countermove. In our setting, each 'move' here consists of an entire (defender- respectively attacker-) action strategy. These have been studied to allocate physical defenses (*e.g.,* [325]), deploy air marshals in planes, place honeypots and security resources in network of computers and IoT devices [108–111, 117], and deactivate products or patch vulnerabilities in an enterprise network [118]. In particular, Serra et al. [118] employed a Stackelberg game to compute the trade-off between the impact of vulnerabilities and productivity in an enterprise network. They evaluated per-host protection on synthetic enterprise graphs with up to 30k edges. In contrast, we focus on the analysis of Pareto-optimal defenses on the entire Internet by focusing on global protections. Thus, we have a different trade-off between scalability and precision. We evaluated our algorithm on a snapshot of the Internet based on the Top 5k Alexa domains and attack graphs with more than 2M edges.

Methodologies to allocate defenses are mainly focused on small (local) networks and synthetic configurations.

Algorithmically, probabilistic defenses against an attacker with uncertainty raises the complexity of the attacker plan task considerably, which is not necessary for our use case: none of the mitigations rely on the adversary's uncertainty about its placement. Another line of research considers graphical security models that include defending nodes (*e.g.,* [182,183]), so-called attack-defense trees, but scale worse than Stackelberg planning [119].

4.10 Conclusions

We proposed a holistic approach to securing the users from Web-based attacks, based on an extensive model of attacks and defenses (with associated costs and security benefits), and an optimized graph algorithm. We analyzed the susceptibility of the top 5K Alexa domains against attackers ranging from cyber-criminal groups to infrastructure providers and nation-state actors. We find that large infrastructure providers are almost as powerful as nation-state attackers. We were able to compute solutions that significantly increase the security of the users. Interestingly, while significant effort has been spent to develop and deploy high-cost mitigations like IPsec or DNSSEC, our analysis highlights that the increase in security is enabled merely by the usage of cheap endpoint defenses like HTTPS, HSTS, and SRI.

Our approach is easy to extend and adapt, and thus provides a foundation for future analyses at the web scale. For example, it can be easily extended with additional mitigations like CSP. Likewise, new technological proposals to improve web security can immediately be added to the mitigation model to compete against existing technologies.

While our approach scales well in the size of the property graph, it does not scale well with the number of possible mitigations, limiting our analysis scenarios where mitigations are adopted globally, instead of host-by-host. Future work can investigate effective pruning techniques on the defender level like some of those presented in [306] and [334].

Chapter 5

Measuring SOC Analysts Investigation of Cyber attacks using the Entropy of Task Complexity

Security Operations Centers are critical socio-technical components of the security of an IT infrastructure. Unfortunately, it is still unclear how to measure the accuracy of the interplay between humans and technology when dealing with the alerts generated by cyber-attacks. In this chapter, we propose to model the complex tasks that a SOC analyst performs during the investigation of the kill chain of attack. With this approach, we explicit the underlying processes required when investigating cyber-attacks to better understand and measure the causal relations between SOC performance and technical skills or technological changes. We evaluated our model with an experiment with 368 students using network traces of intrusions to determine the impact on ticket accuracy of technical skills and technological changes in the SOC. We found that technical skills help but (simple) technological changes do not.

5.1 Introduction

IT infrastructures are constantly monitored by humans and machines to identify the presence of malicious activities on a network. The alerts generated by IDS are processed in a Security Operations Center (SOC) that filters false alarms from events that need further investigation and response. To effectively triage alerts, SOCs may use a hierarchy where higher tiers treat a subset of escalated incidents but analyze each incident more in-depth. Tier 1 analysts compose the largest group with generally less experienced analysts compared to other tiers. This group focuses on real-time monitoring of security incidents, triages a large number of alerts and logs per day, and does not have much time to investigate each alert. Tier 2 analysts are often more experienced analysts who analyze each incident in greater depth and make decisions on appropriate next steps to take if any (e.g. ignore, report to customers, engage in attack containment strategy, etc.). This in-depth analysis is possible because Tier 2 analysts only deal with incidents escalated from Tier 1 analysts [146]. Thus, the accuracy of the reported incident by the Tier 1 analyst is of critical importance to allow Tier 2 analysts to perform in-depth investigations. The lack of ground truth of cyber-attack makes SOC performance evaluation a hard task [273] and current performance metrics are detached from the real-world experience of analysts [180,321]. The misalignment between performance metrics and analyst tasks can create suboptimal security outcomes when SOC operations favor increasing metrics scores (e.g. by reducing the mean time-to-analyze) over performing accurate analyses of security events. This in turn can lead to analysts burnout [320] as well as making impossible comparisons across SOC configurations, or differently managed SOCs [273, 294]. A recent approach to solve this problem is to analyze cyber-attacks and defenders' performance in isolated and virtualized environments [273, 332]. Even in a controlled environment, it is hard to measure performance due to the different complexity of the operations executed by attackers and defenders. Evaluating the performance based only on the outcome (success or failure of detection) will likely produce imprecise results [332].

In this chapter, we propose a novel approach by extending Wood's classical theory of task complexity [359] and recursively defining the tasks and acts performed during the analysis of a cyber-attack. We investigated the following questions:

***RQ3a*:** How to model the task complexity of investigating a cyber-attack?

***RQ3b*:** How task complexity helps in comparing changes in accuracy due to technical (skills of analysts) and technological differences (different filtering of alerts)?

We focus on the process of analyzing traces (using alerts) of cyber-attacks to determine the phases of the intrusion from Tier 1 analysts. Our focus is on the complexity of the investigation of the attack itself. In this chapter we make the following contributions:

- we extended Wood's theory of task complexity by employing entropy to describe how information and acts are distributed and concur in the complexity of a task. We apply the model to describe the investigation of the kill chain of cyber-attacks by a SOC analyst (Section 5.3).
- we propose an approach to measure the accuracy of SOC analysts' incident reports

based on the correct and wrong content of a ticket produced during the investigation weighted by the associated complexity (Section 5.4).

- we evaluate how analysts' skills and SOC alert filtering affects the accuracy in a controlled experiment with 368 students using traces of real attacks (Sections 5.6 and 5.7).

We provide the low-level details to compute the complexity score in the Appendix C.0.1.

Non-goal: We do not aim to model the overall complexity of a SOC analyst's daily job activity that includes several tasks *not* related to the investigation and reporting of cyber-attacks [180, 322].

5.2 Background

To make the chapter self-contained we introduce the key concepts related to SOCs and task complexity.

Security Operations Centers Security operations centers (SOC) aim to monitor, detect, report, and, oftentimes respond to incoming cyberattacks or other security incidents in an organization. SOCs oftentimes employ (network) intrusion detection systems (NIDS), that capture network traffic and generate alerts on suspicious activities defined by an organization [372] and encoded in so-called signatures. Because of the generality of many of those signatures, (N)IDSs are known to generate a very high volume of alerts [371], of which the vast majority are either irrelevant (i.e. not indicating actual suspicious behaviour), or false positives [321]. In SOCs, human analysts are assisted by technology (in the form of correlation engines, also called SIEMs) with the role of identifying incidents from the noisy data [321]. Most SOCs employ a Tier-ed system, where Tier 1 analysts have the crucial role of distinguishing 'wheat from chaff' to pass on relevant information to the higher tiers on potential incidents in a monitored environment [321].

Much of the literature has focussed on the hardship of finding 'true positive' alerts in a sea of 'false positives' [16, 130]; yet, an often overlooked task of a (tier 1) analyst is that of *correctly investigating* a 'true positive' alert such that appropriate, complete, and timely information is passed on through to the higher tier analysts.

Correctly identifying and investigating a 'true positive' is a highly complex task, where the analyst pieces together information cues from multiple sources, and references multiple stages of an attack [273]. Despite it being key to determining the performance of a Tier 1 analyst, it is often hidden in aggregate metrics [294, 295] or overlooked altogether.

Indeed, as almost all alerts are ‘not interesting’ (see the ‘base rate fallacy’ first outlined in [39]), simply characterizing a Tier 1 performance on the basis of their ability to mark false positives as such can lead to misleading conclusions. On the other hand, the ability of Tier 1 to execute complex tasks involving multiple and diverse information sources should be captured to appropriately describe an analyst’s performance in identifying and characterizing the (few) alerts that matter.

Task Complexity and the Wood’s Basic Model One key factor that determines human performance is the complexity of the task. The seminal work of Wood [359] defined the complexity of a task as a function of its structure. Wood identified two task components:

- *acts*: are the actions required to create the product.
- *information cues*: are pieces of information utilized by an individual to perform decisions during a task.

From these elements, he defined two complexity dimensions: *component* and *coordinative* complexity. Wood’s component complexity is a direct function of the number of distinct acts and information cues needed to achieve a given task. The component complexity considers the redundancy of the acts and information cues (component redundancy). For example, if an act is performed multiple times to achieve a specific task, the component complexity is reduced to a single act. The component redundancy also applies in case the same knowledge, defined in terms of information cues, is used by different acts related to a task. An act takes part in the overall complexity if it requires one or more information cues. Wood’s coordinative complexity is a function of the relationship between information cues and acts typically as causal relations between acts, in which an act must be performed before another.

We started from the Wood’s model [359] to define a model for the SOC tasks complexity because it is the most widely used objective complexity model [203] and other attempts to describe task complexity [59, 152, 278] are based on the elements of Wood’s model [152, 203]. A limitation of Wood’s model is that focused on "simple" tasks and does not scale up to more complex and structured tasks, such as those of a SOC analyst. Consider a task in which there are dozens of distinct information cues that must be employed to perform a single act. For example, an analyst has commonly to relate alerts from different sensors and sources (e.g. alerts from a Network Intrusion Detection System with alerts from a Cloud environment) and evaluate them in the scope and context of the monitored environment together with relevant metadata such as information about the targeted systems (e.g. vulnerability data), and other network data and logs on net-

Table 5.1: Task Elements

Element	Definition	Example
Act	An act is an atomic action that composes a (sub-)task.	Analysis of the connections response status in an IDS log.
External Inputs (I_E)	Inputs not produced by any other sub-task associated with the task.	IP address in an alert, domain name of a malware C&C.
Total Inputs (I_T)	All inputs utilized by the acts composing a certain (sub-)task. Contains both I_E and the inputs that are the result of previous acts or sub-tasks.	IP address of a compromised machine in the network obtained from a previous sub-task and used to determine lateral movements.

work/system activities temporally adjacent to the event of interest. This task should have a higher complexity than a task with the same amount of information cues that can be split into several smaller *independent* subtasks with a subset of the inputs each. In Wood's theory, two tasks with the same number of inputs, but a different distribution of those over the acts, would have the same complexity. This can lead to a task underestimation that is the main source of burnout [320].

5.3 A New Theory of Task Complexity

We define task complexity by means of its entropy to measure the accuracy of SOC analyst incident reports on tasks of comparable complexity. High accuracy of incident reports results in more valuable information for the Tier 2 analyst analysis when the ticket is escalated. Similarly, low accuracy report results in a higher workload for the Tier 2 analysts, that must determine all the errors in the escalated tickets. Yet such analysis only makes sense if the tasks relative complexity can be assessed.

5.3.1 A Recursive Model of Task Complexity

To recursively evaluate the complexity of a task, we identify the following elements: *Acts*, *External Inputs* (I_E), and *Total Inputs* (I_T). Table 5.1 summarizes each element. We always satisfy the following constraints: $I_E \leq I_T$ and $\#Acts \leq I_T$.

A task can be composed of a sequence of sub-tasks and recursively, a sub-task can be split into several sub-sub-tasks. A (sub-)task is atomic if it is composed of a single act. A sub-task does not take part in the complexity if it does not require any input.

We considered component redundancy in which if the same sub-task is repeated multiple times to achieve a specific task, the sub-task is counted only once. An example is the analysis of malware, in which the executable is uploaded multiple times on different antivirus scanners. Indeed, the procedure can be automated e.g. using VirusTotal.

In contrast with Wood [359], we consider the reuse of the same input over different acts as part of the overall complexity as one needs to know that the information must be used multiple times. For example, a symmetric key to encrypt C&C communications is required to decode both client and server packets.

The classification of an input as an external one depends on the point where one computes the complexity. For example, when computing the I_E and I_T for a sub-task referring to the identification of the attack vector phase, one could consider all inputs to be external, even if some of them come from other sub-tasks. We consider an external input if it does not come from any other sub-task that the analyst must perform to identify the kill chain of an attack. Following the previous example, the information that an internal IP is compromised is necessary to identify the lateral movement phase of an attack. This information does not come from any external sources but it is the result of a previous sub-task of the investigation. We will then evaluate the correctness of sub-tasks depending on the successful achievements of previous sub-tasks whose output is required as input (see further Section 5.6).

Given the $\#Acts$, I_E, and I_T, we can compute two dimensions of complexity that aggregate and generalize the original Woods elements into different dimensions:

Information Complexity (IC) is obtained by the ratio of I_E and I_T. It is a value between 0 and 1. A value near 1 means that the information required to perform a task is obtained mostly from external sources and is not the result of internal processing of sub-tasks composing the task. Given a fixed value of I_T, the smaller the IC the more complex the task because it requires producing a higher amount of information to complete the task. While this definition seems, at first sight, counterintuitive, its entropy provides the right intuition as we shall see later in the section. As the produced information is necessary to perform subsequent acts in the task, it generates a causal relation between acts.

$$IC = \frac{I_E}{I_T} \tag{5.1}$$

Structural Complexity (SC) is obtained by the ratio of $\#Acts$ and I_T. It is a value between 0 and 1. A value near 1 means that there is a more uniform distribution of information (i.e. the inputs) over all acts. Given a fixed value of I_T, the smaller the SC, more complex the task as it requires each act to process more information on average. The reciprocal of the Structural Complexity describes the average number of inputs for each act.

$$SC = \frac{\#Acts}{I_T} \tag{5.2}$$

5.3.2 Entropy of Tasks Complexity

Shannon entropy defines the amount of information contained in a random variable which is linked to how difficult is to guess the result of the variable (in our case how much "surprise" a variable reading produces). A random variable with a probability p equal to 1 makes the task of guessing the value straightforward. Hence, from a complexity perspective, a value close to 1 is easy to guess and no complex analysis is needed.

We define the entropy of a task complexity over the Information and Structural Complexity to measure the amount of information to produce and employ in an act respectively.

$$H_{IC} = -(IC) * \log_2(IC) \tag{5.3}$$

$$H_{SC} = -(SC) * \log_2(SC) \tag{5.4}$$

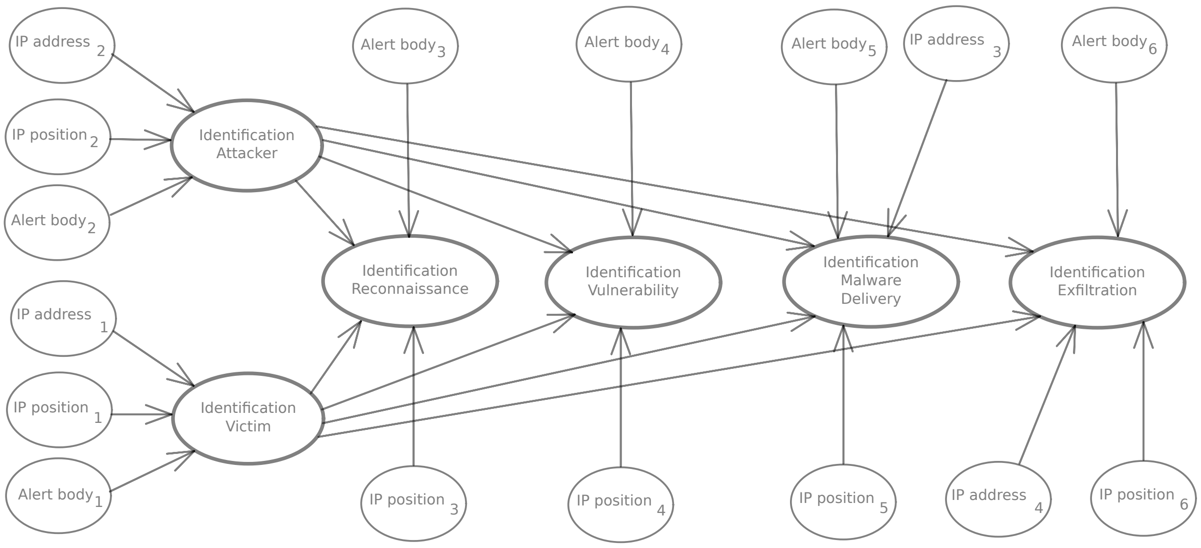
For example, a task with $H_{IC} = 0$ means that the amount of new information that the SOC analyst needs to produce is zero and can just rely on the external information provided by the IDS.

5.4 SOC Analyst Task Complexity

We now answer to ***RQ3a*** by instantiating the complexity of the investigation of cyber-attacks by a SOC analyst and using that to measure the accuracy of the incident reports and thus the effectiveness of technical and technological changes in the SOC.

Each SOC organization may include among the duties associated with a Tier 1 analyst different sub-tasks and unfortunately T1 analysts spend most of their time looking at irrelevant data [16, 322]. So our model focuses on the specific *task* of the analyst to identify an attack from the sequence of alerts issued by the SOC, i.e. when they figured out that they *might* be looking at an alert that involves an attack. Following the MITRE ATT&CK tactics framework [5], the analyst should eventually identify:

- victim
- attacker
- reconnaissance steps
- exploited vulnerabilities



The task of investigating cyber-attacks by a SOC analyst can be complex if observed externally with several different inputs to consider. However, it is decomposed into a sequence of sub-tasks in relation to each other. Each sub-task requires part of the overall input of the general task. Depending on the scenario, the sub-task can be further split into a sequence of acts. A SOC analyst activity includes also additional tasks not related to the investigation of the phases of an attack. The investigation typically starts from the identification of a victim or attacker machine and then pivots to determine all attack phases (reconnaissance, vulnerability exploited, malware C&C, data exfiltration). Different information (*IP address$_1$*, *IP address$_2$*, ...) must be processed by different sub-tasks (`Identif. Victim`, `Identif. Attacker`, ...)

Figure 5.1: SOC analyst investigation of cyber-attacks decomposed in sub-tasks with their causal relations

- malware delivery
- exfiltration steps

Figure 5.1 shows the investigation of cyber-attacks as a causal bayesian network [253]. Each phase is a sub-task that uses outputs produced by other sub-tasks as inputs for its acts. For example, the identification of the victim produces as output the exploited victim IP, which can then be used to identify the exploited vulnerability. Instances of attacks are dynamic and evolve but this does not change the types of the Tier 1 analyst's tasks. A type does not change over time thus we do not need to model their evolution [359].

Table 5.2 summarizes for each *sub-tasks* the I_E, I_T, output, and causal relations with other sub-tasks. There are many possible outputs for each sub-task. Here, we focus on the outputs indicated in bold in Table 5.2.

Table 5.2: SOC analyst sub-tasks complexity

In the most general scenario, for each sub-tasks, each alert body is unique. The same alert body may be reused for different sub-tasks. The reuse of inputs takes part in the overall complexity (Section 5.4). A sub-task can produce different outputs, we reported in bold the one used in the experiment (Section 5.6)

Sub-task	I_E	I_T	Output(s)	Causal Req(s)	Description
`Identif. Victim`	*IP address$_1$*, *IP position$_1$*, *alert body$_1$*	I_E(`Identif. Victim`) + I_T(*analyze body_alert$_1$*) − I_E(*analyze body_alert$_1$*)	**IP Victim**, system type, ...	/	Identified by an alert in which a destination/source receives/sends a suspicious packet.
`Identif. Attacker`	*IP address$_2$*, *IP position$_2$*, *alert body$_2$*	I_E(`Identif. Attacker`) + I_T(*analyze body_alert$_2$*) − I_E(*analyze body_alert$_2$*)	**IP Attacker**, geolocation, ...	/	Identified by an alert in which a source/destination sends/receives a suspicious packet to/from a victim.
`Identif. Recon`	*IP position$_3$*, *alert body$_3$*	I_E(`Identif. Recon`) + I_T(*analyze body_alert$_3$*) − I_E(*analyze body_alert$_3$*)	**Recon technique**, timing, ...	`Identif. Victim` OR `Attacker`	Identified by alert(s) in which the attacker sends probes to IP(s) of the network.
`Identif. Vuln.`	*IP position$_4$*, *alert body$_4$*	I_E(`Identif. Vuln.`) + I_T(*analyze body_alert$_4$*) − I_E(*analyze body_alert$_4$*)	**Vuln exploited**, sw affected, ...	`Identif. Victim` OR `Attacker`	Identified by an alert in which the attacker sends a packet containing an exploit to the victim.
`Identif. Malware Delivery`	*IP address$_3$*, *IP position$_5$*, *alert body$_5$*	I_E(`Identif. Mw Delivery`) + I_T(*analyze body_alert$_5$*) − I_E(*analyze body_alert$_5$*)	**IP Server**, malware family, ...	`Identif. Victim` OR `Attacker`	Identified by an alert in which the victim initiates a suspicious connection towards an external IP.
`Identif. Exfiltr.`	*IP address$_4$*, *IP position$_6$*, *alert body$_6$*	I_E(`Identif. Exfiltr.`) + I_T(*analyze body_alert$_6$*) − I_E(*analyze body_alert$_6$*)	**IP C&C**, exfiltrated info, geolocation, ...	`Identif. Victim` OR `Attacker`	Identified by an alert in which the victim initiates a suspicious connection towards an external IP.

Sub-task - Identification Victim The victim can be identified by alerts that describe suspicious activities directed to or from a machine in the network.

Example 5.4.1. *An alert reporting that a machine in the network has received a packet matching the signature of an exploit or an alert reporting that a machine has sent a packet containing credentials.*

The sub-task can be split into two *sub-sub-tasks*:

- *analyze* body_alert/log
- *analyze* the IP

The first sub-sub-task can be recursively expanded in terms of I_T, I_E, and *# Acts.*

Example 5.4.2. *The body of the alert can contain information like HTTP requests, user-agent, encoding, etc. (I_E). Furthermore, an analyst should first get the URL contacted, and then probe the content of the malicious website (#* Acts*).* The expansion depends on the specific scenario and we will instantiate them in Section 5.5.

The second sub-sub-task can be seen as an act that requires considering as information the IP address and its position (source or destination of the connection) in the alert. The IP position is important to contextualize the results of the analysis of the body alert.

There is no causal relation between the two sub-sub-tasks (*analyze* the IP and *analyze* body alert) as well as between other sub-tasks.

Sub-task - Identification Attacker The attacker can be identified by processing similar alerts as the one used for the previous sub-task. We have the same structure of the *Identification Victim.*

Example 5.4.3. *An alert reporting that a certain external source has sent to a destination in the network a suspicious packet containing an exploit or a probe or an alert reporting that a certain machine in the network has sent a suspicious packet containing credentials to an external machine.*

Sub-task - Identification Reconnaissance The (active) reconnaissance phase can be identified by processing alerts related to attacker's actions aimed at gathering information from the victim or the network.

Analysts can link the reconnaissance to an attack using either the attacker or the victim IP because not all scans are linked to attacks and different IPs can be employed in sequence to perform different steps of the kill chain (e.g. reconnaissance and exploitation can be performed using different machines) [198]. There is a causal relation between the sub-tasks (`Identification Victim`/`Identification Attacker` and `Identification Reconnaissance`). Thus, the victim/attacker IP information is not considered part of I_E but it is instead obtained from a previous sub-task.

In case of a scan, multiple packets are sent thus the analyst needs to observe n alerts and analyze n body_alert and n times the sender information. There is no causal relation between these n acts. Given that most of the SOC alerting software (e.g. IDSs) automatically cluster scan attempts into a single alert, the analyst does not need to analyze n distinct alert body, but a single alert that summarizes a scan attack. This alert body will contain additional information to use as an external input: the information that there have been several probes.

Example 5.4.4. *An alert reporting that a certain external source has (repeatedly) contacted IPs in the network on specific ports.*

Sub-task - Identification Vulnerability The vulnerability exploited during an attack can be identified by processing alerts related to the exploit phase performed by the attacker against the victim. Similarly to the reconnaissance sub-tasks, there is a causal relation with either the sub-tasks `Identification Victim` or `Identification Attacker`. In case the type of vulnerability requires multiple payloads (e.g. weak password or race-condition via brute force), multiple alerts are clustered by the SOC tool by adding in the body of the alert information about the brute-force attempt. Also, there is no causal relation between alerts related to e.g. different failed logins.

Example 5.4.5. *An alert reporting that an external source sent to a destination in the network a suspicious packet containing an exploit.*

Sub-task - Identification Malware Delivery The malware delivery phase can be identified by processing alerts related to a suspicious connection from the victim to an external IP belonging to the attacker.

Example 5.4.6. *An alert reporting that a second-stage malware or an exploit for privilege escalation was downloaded by an internal machine.*

The analysis of the alert can require a sequence of acts to resolve the domain contacted, access the website content, and decode the content to determine the presence of an exploit. Thus, there is a causal relation between the sub-tasks `Identification Victim/Attacker` and `Identification Malware Delivery`.

Sub-task - Identification Exfiltration The exfiltration phase can be identified by processing alerts with a structure similar to the delivery phase. The difference stays in the alert content analyzed by the SOC analyst.

Example 5.4.7. *An alert reporting a suspicious connection started from the victim towards an external IP, in which the source sends sensitive information e.g. credentials to an external machine.*

5.4.1 Accuracy of the SOC Analyst Task

We measure the accuracy using the overall correct (true information (T)) and wrong (false information (F)) content of the tickets referring to actual attacks using the sum of the entropy of the correctly and incorrectly achieved sub-tasks, weighted by their total input I_T. These values measure the accuracy of the SOC analyst in investigating attacks by looking at the information issued in the tickets. We underline that the values are computed on tickets that identified actual attacks because the unrelated tickets do not have a characterization of their complexity. Let $t \in tickets$, $p \in phases$ we define:

$$H(IC|T) = \sum_{t} \sum_{p \in Correct} I_{Tp} * H_{ICp} \tag{5.5}$$

$$H(IC|F) = \sum_{t} \sum_{p \in Wrong} I_{Tp} * H_{ICp} \tag{5.6}$$

$$H(SC|T) = \sum_{t} \sum_{p \in Correct} I_{Tp} * H_{SCp} \tag{5.7}$$

$$H(SC|F) = \sum_{t} \sum_{p \in Wrong} I_{Tp} * H_{SCp} \tag{5.8}$$

The outcome on each dimension (IC and SC) can be influenced by the skills of the analyst, as well as by the SOC tools available. We performed an experiment to determine the effect of these factors on the accuracy of SOC analyst tickets related to the investigation of a cyber-attack.

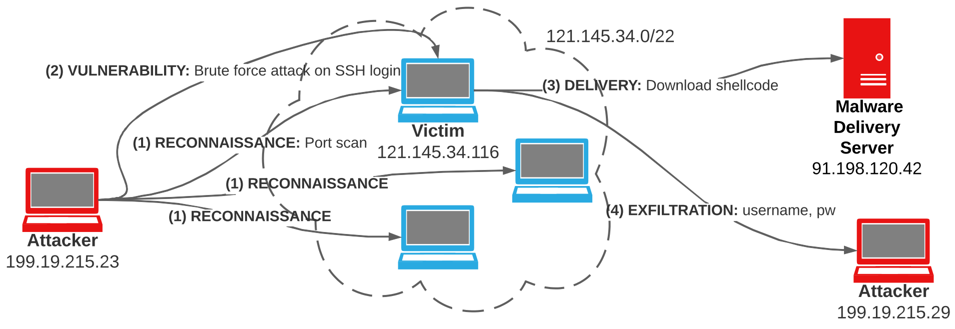
A question that can arise is how to handle path dependencies, in particular when measuring false information produced by SOC analysts where an early mistake in the investigation, can propagate to the subsequent steps. From Figure 5.1, we can observe that the only mistakes that can influence the next phases of the investigations relate to the identification of the attacker and victim, from which other phases of the attack are based (i.e. reconnaissance, vulnerability, malware delivery, and exfiltration). However, in our model, false information in the identification of a phase is considered independent from false information in the other parts of the tickets. The motivation is that, although e.g. a wrong attacker can lead to looking into unrelated alerts, the actual error is performed on the subsequent sub-task where the analyst failed to interpret correctly the information clues and determine that the alert is unrelated to the actual attack. For example, in the experiments (see Section 5.6), we observed tickets where the NS server was incorrectly identified as the attacker, and the subsequent vulnerability was incorrectly classified as a DNS RCE. In this case, although the incorrect classification of the attacker leads the analyst to look into DNS log requests, the false information in the vulnerability identification is due to the fact that the analyst did not correctly identify the connection as a benign DNS query.

5.5 Attack Scenarios Complexity

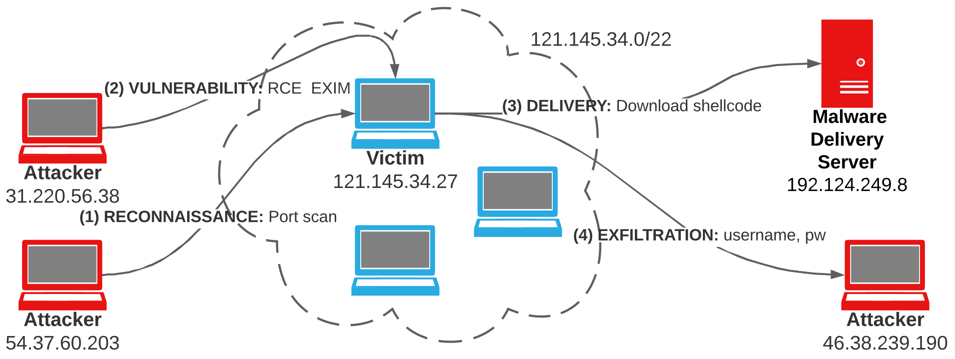
We instantiated the model of SOC task complexity (Section 5.4) with two attack scenarios injected in a platform that emulates a SOC infrastructure [273]. The scenarios' sub-tasks could be solved in different ways i.e. processing different alerts.

We present the two attack scenarios available in the tool [273] used for the controlled experiment.

Attack scenario MIRAI This attack reproduces the phases of an infection by the MIRAI botnet [30]. Figure 5.2a summarizes the phases of the attack. The malware scans several ports on the subnet `/22` of the internal network from the IP `199.19.215.23`. It



(a) MIRAI Attack Scenario



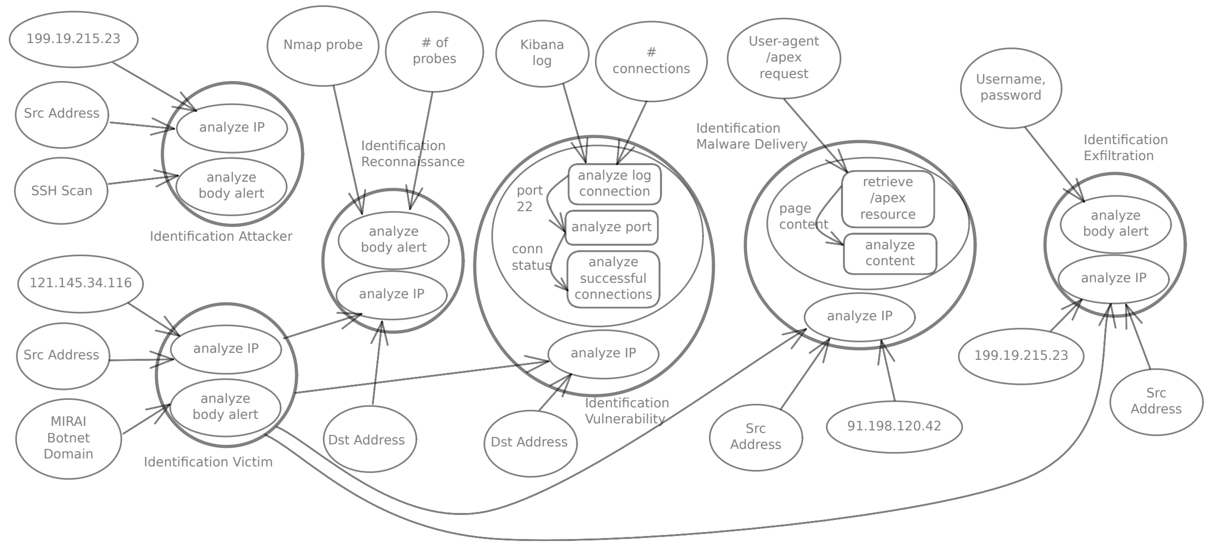
(b) EXIM Attack Scenario

Figure 5.2: The network architecture and sequence of phases of the attack scenarios

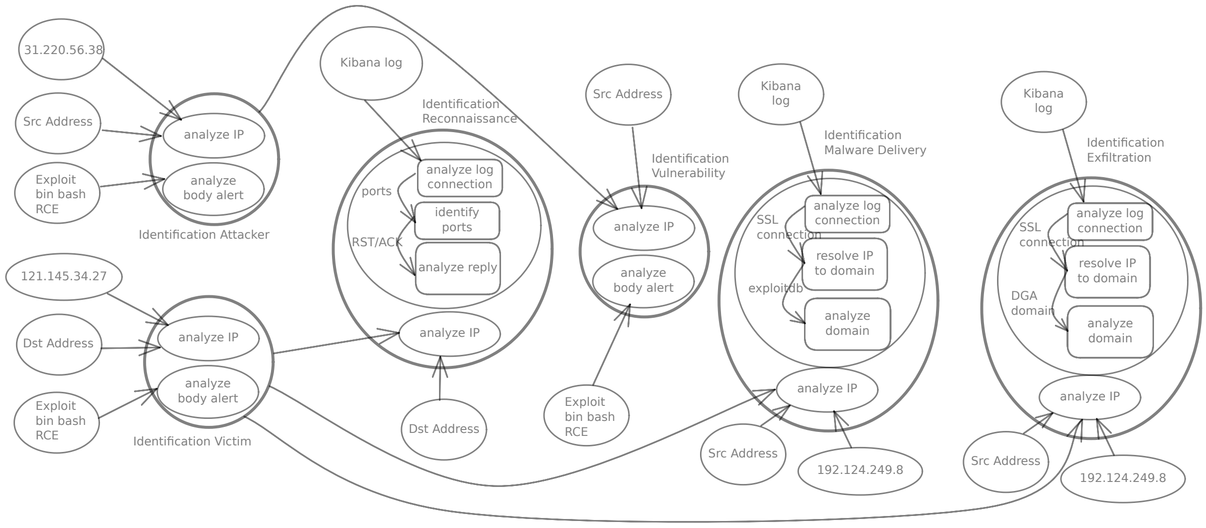
attempts a brute-force password attack on the SSH port. It eventually gets access to an internal machine with IP `121.145.34.116` (the actual victim IP cannot be shown for anonymity purposes). From this machine, it downloads a shellcode to escalate from an external domain *p.pi.fi* (IP `91.198.120.42`), and finally exfiltrates the credentials to the external IP `199.19.215.29`.

Attack Scenario EXIM This attack reproduces the exploitation of a Remote Command Execution vulnerability against the EXIM SMTP server [88]. Figure 5.2b summarizes the phases of the attack. The attacker performs a stealthy scan from the IP `54.37.60.203` to determine the vulnerable server on the IP `121.145.34.27`. The attacker sends a payload from another IP `31.220.56.38` to exploit the vulnerability on the server. It downloads an exploit from *exploit-db.com* (IP: `192.124.249.8`), and finally, it exfiltrates sensitive data to an external machine with domain *l6asd8cs-google-support.abc.xn-p2a.jetzt* (IP: `46.38.239.190`).

Table 5.3 summarizes the $\#$ *Acts*, I_T, I_E, and the entropy for the two complexity dimensions. The overall complexity and entropy are also computed for the entire attack



(a) MIRAI SOC analyst investigation



(b) EXIM SOC analyst investigation

Figure 5.3: Sub-tasks and causal relations in attack scenarios. Details in Appendix C.0.1

scenario. Figure 5.4 shows the entropy of the task complexity dimensions for MIRAI and EXIM. Each point represents a sub-task corresponding to the identification of one phase of the overall attack. Each phase can not only differ by the IC and SC but also on the total amount of information required (I_T). Figures 5.3a and 5.3b show the instantiation of Figure 5.1 for the MIRAI and EXIM attack scenario respectively.

Our model is based on the concept of objective task complexity where the task complexity is only defined by the task characteristics and is independent of the task performers. Therefore, although SOC analysts could interact with the task in different ways, for

Table 5.3: Task complexity of sub-tasks by attack scenario. Details in Appendix C.0.1

	MIRAI					EXIM					
Sub-task	# Acts	I_E	I_T	$H(IC)$	$H(SC)$	# Acts	I_E	I_T	$H(IC)$	$H(SC)$	Causal Requirements
Identification Victim	2	3	3	0.0	0.270	2	3	3	0	0.270	-
Identification Attacker	2	3	3	0.0	0.270	2	3	3	0	0.270	-
Identification Reconnaissance	2	3	4	0.216	0.347	4	2	5	0.366	0.178	1
Identification Vulnerability	4	3	6	0.347	0.270	2	2	3	0.270	0.270	1
Identification Delivery	3	3	5	0.306	0.306	4	3	6	0.347	0.270	1
Identification Exfiltration	2	3	4	0.216	0.347	4	3	6	0.347	0.270	1
Overall Task	15	18	25	0.236	0.306	18	16	26	0.299	0.255	4

example similarly to what was observed by Mantovani et al. [208], by skipping information and steps, the overall complexity of the task and sub-tasks do not change. Our model allows one to measure the overall performance that can be influenced by the approach and knowledge of the analyst as we will show in section 5.7.

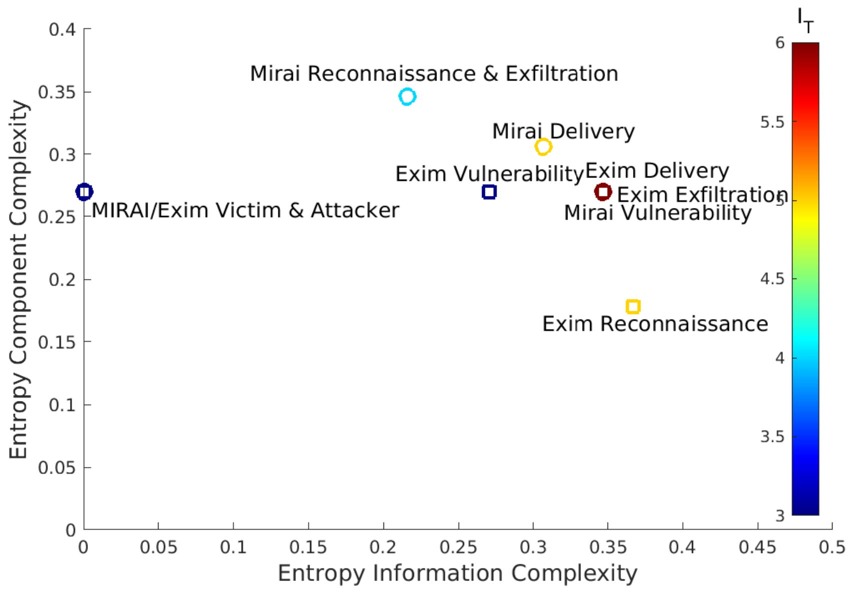
We classified tickets as related to either MIRAI or EXIM if at least one attacker IP/domain or victim IP is present in the ticket in the "Attacker IP" or "Victim IP" fields. All other tickets were considered unrelated to the scenarios. *Only* for the tickets that identify one of the two scenarios, the researchers analyzed the rest of the tickets to determine which sub-tasks were correctly or incorrectly performed. If a group issued two tickets for the same attack, only the most complete one was considered. Each sub-task reported in Table 5.2 is scored with one of the following values: OK, NO, and MAY. Table 5.4 and Table 5.5 show the general guidelines utilized for the scoring. The rules in Table 5.4 apply to tickets that contain at least one attacker or victim IP in the corresponding fields. While Appendix C.0.3 in the appendix describes the instantiation for the two scenarios.

5.6 Experimental Procedure

We describe the experiment structure, the demographics of the groups, and the scoring mechanism.

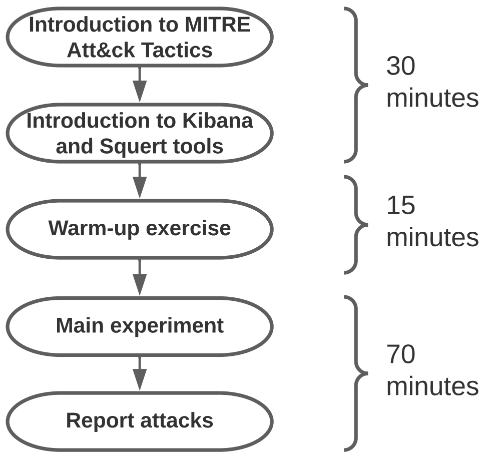
5.6.1 Experiment Structure

The experiment started with a short introduction to a compacted version of MITRE Att&ck Tactics (Reconnaissance, Initial Access, and Command & Control). Followed by an introduction to the tools used in the experiment (Kibana and Squert). Students practiced with the tools via a warm-up exercise. Figure 5.5 summarizes the timeline of the experiment.



The Entropy for Information Complexity (IC) and Structural Complexity (SC) for the MIRAI and EXIM sub-tasks with different values of I_T. Different sub-task can present a similar value in one complexity dimension but differ in the other. For example, the sub-task *Exim Vulnerability* and *Exim Exfiltration* present a similar structural complexity but the latter has a higher information complexity as it requires one to produce more information to perform the sub-task.

Figure 5.4: Entropy for MIRAI and EXIM attack scenarios


Introduction to MITRE Att&ck Tactics
Introduction to Kibana and Squert tools
Warm-up exercise
Main experiment
Report attacks
30 minutes
15 minutes
70 minutes

The experiment consisted of an introduction to the phases of an attack following the MITRE Att&ck framework, followed by an introduction to the tools, and a warm-up exercise. The main experiment consisted of 50 minutes of alert investigations and 20 minutes for reporting the attacks.

Figure 5.5: Experiment Timeline

Table 5.4: Guidelines scoring MIRAI and EXIM tickets: attacker and victim IPs

Score	EXIM Attacker	EXIM Victim	MIRAI Attacker	MIRAI Victim
OK	The reported attacker IPs include **only** the Attacker IP $\vee$ (the Attacker IP $\wedge \leq 2$ wrong IPs $\wedge$ correct explanation)	The reported victim IPs include **only** the Victim IP $\vee$ (the Victim IP $\wedge \leq 2$ wrong IPs $\wedge$ correct explanation)	The reported attacker IPs include (at least 2 out of 3 Attacker IPs $\wedge \leq 2$ wrong IPs) $\vee$ (1 Attacker IP and ≤ 2 wrong IPs $\wedge$ correct explanation)	The reported victim IPs include **only** the Victim IP $\vee$ (the Victim IP $\wedge \leq 2$ wrong IPs $\wedge$ correct explanation)
NO	The reported attacker IPs include (the Attacker IP $\wedge >2$ wrong IPs) $\vee$ (a mix of IPs from the two attack scenarios) $\vee$ (**only** wrong IPs but 'EXIM Victim' contains the Victim IP)	The reported victim IPs include (the Victim IP $\wedge >2$ wrong IPs) $\vee$ (a mix of IPs from the two attack scenarios) $\vee$ (**only** wrong IPs but 'EXIM Attacker' contains the Attacker IP)	The reported attacker IPs include (the Attacker IP $\wedge >2$ wrong IPs) $\vee$ (a mix of IPs from the two attack scenarios) $\vee$ (**only** wrong IPs but 'MIRAI Victim' contains the Victim IP)	The reported victim IPs include (the Victim IP $\wedge >2$ wrong IPs) $\vee$ (a mix of IPs from the two attack scenarios) $\vee$ (**only** wrong IPs but 'MIRAI Attacker' contains the Attacker IP)
EMPTY	The reported attacker IP is empty	The reported victim IP is empty	The reported victim IP is empty	The reported attacker IP is empty
MAY	Everything else	Everything else	Everything else	Everything else

Table 5.5: Guidelines scoring tickets phases

Score	Explanation	Example	Attributed Complexity
OK	(The reported attacker's IPs include **only** actual Attacker's IPs $\vee$ the reported victim's IPs include **only** actual Victim's IPs) $\wedge$ phase is right	Reported the C&C server IP for the exfiltration phase	$+H(IC)/H(SC)$
NO	(The reported attacker's IPs do not include any actual Attacker's IPs $\wedge$ the reported victim's IPs do not include any actual Victim's IPs) $\vee$ phase is wrong	Reported a wrong IP for the C&C server and the victim.	$-H(IC)/H(SC)$
MAY	Everything else	Reported the C&C server IP along with other wrong IPs.	0

The main experiment consisted of 50 minutes of live investigations followed by 20 minutes of reporting the attacks on an online platform. The students were allowed to access the tools during the reporting phase. We asked the group to fill out tickets with the attacks identified. The ticket form asks to insert the information in the output column of the sub-tasks in Table 5.2 *in bold* (*Identification Victim*, *Identification Attacker*, *Identification Reconnaissance*, *Identification Vulnerability*, *Identification Delivery*, and *Identification Exfiltration*). The groups were allowed to issue up to 5 tickets.

SOC Filtering Rules We employed two types of SOCs that differ only in the deactivation of a set of rules related to policy violations (e.g. due to the use of TOR or P2P connections) and *not* on the rules that are triggered by the attack scenarios. This is to simulate a less fine-tuned SOC environment that generates noisy alert output [346]. The SOC `Basic` contains all default rules from Suricata, while the SOC `Filter` contains all rules of `Basic` filtering out the ones related to policy violations. We injected into the two SOC the same network traffic that includes the two attack scenarios (MIRAI and EXIM). We assign groups randomly to the two SOC types.

Students Background and Skills Assessment Some days before the experiment, students were asked to individually fill out a short questionnaire to self-assess their background on a Likert scale from 1 to 5 regarding the following topics: *Web programming*, *Network Security*, *Network Analysis*, and *Hacking*. To reduce interpretations in the self-assessment, instead of the classical terms like "Familiar" and "Expert", we "instantiated" the possible options with more concrete entries. For example, for assessing the *Hacking* background we used "I have attended a couple of lectures in a course", "I am a professional pen tester (SANS certified or similar)...". Table C.2 in the Appendix shows an example of the instantiation.

In addition, we asked the students ten questions on technical aspects of the same topics of the background. The questionnaire is inspired by the SANS workforce test. We have considered using the SANS questionnaire directly but we have decided not to because the complete test requires over an hour to assess the participant's skills. Table 5.6 shows for each topic, the number of questions asked. Each question is structured as a multiple choice in which there is a unique right answer.

Table 5.6: Skill technical questions

Category	# of questions
Web programming	2
Networking	5
Network analysis	2
Vulnerability	1

5.6.2 Demographics and Ground Truth Determination

Participants Demographics Students belonged to two distinct degrees. The first degree, `Spec`, is enrolled in a Computer Science MSc and students were recruited from a cyber-security course that requires participants to have completed a set of prerequisite courses on information security topics like Network Security and Security Testing. The second degree, `Gen` did not require any prerequisite courses on security. Thus the `Spec` group presents a security-based educational background compared to the `Gen` group. Furthermore, students in the `Spec` attended a specific lecture and an exercise on IDS using Snort. Students were free to organize in groups of at most 2 members. We collected data in three rounds carried out in three consecutive academic years. The first and third experiments were carried out in presence, while the second experiment was carried out online due to the COVID-19 pandemic (we discuss the possible impact in Section 5.8).

Table 5.7 summarizes the final number of students, groups, and tickets divided by degree (`Spec`, `Gen`) considered in the analysis. We had 368 students in the experiment composing 184 groups. Table 5.9 shows the average background and skills of the groups as a sum of the individual assessments.

Scoring of Tickets For each round, two researchers manually inspected the tickets issued by each group and determine if the ticket identified or not one of the two scenarios (MIRAI or EXIM) by analyzing the IPs and domains identified by the group. The first two rounds were evaluated by two researchers, the third round was evaluated by one of the previous researchers and a third researcher who was not involved in the previous rounds. We ignored the tickets from groups composed of a single student or for which at least one student in the group did not submit the skill assessment questionnaire, for a total of 33 groups. For some of these cases, we identified misspelled student IDs. We manually investigated IDs and linked questionnaires to tickets issued if the ID does not differ by more than one digit. We assessed the agreement in the annotation of the sub-tasks by the two researchers using the Cohen κ. For the first two rounds, we obtained values that range between 0.97 and 1 for the different phases, for the third round the values range between

Table 5.7: Number of Students, Groups and Tickets issued by Degree

Degree	# students	# groups `Filter`	# groups `Basic`	# tickets
`Gen`	320	78	82	409
`Spec`	48	13	11	79
Total	368	91	93	488

Table 5.9: Groups Mean and Std Skills and Background

Overall Skills (Groups of 2)	`Spec`		`Gen`	
	(n=48)		(n=320)	
	Mean	Std	Mean	Std
Tested Assessment, range [0,20]	17.7	1.6	12.0	3.4
Self-assessment, range [2,10]				
Background Hacking	4.9	1.0	3.3	1.4
Background Web Programming	4.7	1.2	3.8	1.6
Background Network Analysis	5.5	1.1	3.4	1.3
Background Network Security	5.5	1.0	3.5	1.1

0.87 and 1. This indicates an almost perfect agreement [187] between the annotators. 6 tickets for which the two researchers did not eventually reach an agreement were dropped from the analysis.

5.7 Results

We performed the Kuder-Richardson 20 test to check internal reliability for the entire set of skills assessment questions ($\alpha = 0.72$). Our value meets the threshold as stated by Nunnally [237]. The Pearson correlation coefficient between the background and the technical questions shows a moderate correlation (ρ=0.49, $p < 0.0001$).

Among the various groups, 21 out of 184 groups (11.4%) issued a ticket related *only* to the EXIM scenario, 58 out of 184 (31,5%) issued a ticket related *only* to the MIRAI scenario, and 73 (39.7%) issued tickets related to both scenarios. We ignored tickets that did not refer to either the EXIM or MIRAI attack scenarios as our goal is to investigate the accuracy of the investigation of cyber-attacks and not on irrelevant alerts. Furthermore, the experiment potentially incentivized students to report as many attacks as possible due to the absence of negative scoring.

We report in Table 5.10 the descriptive statistics of the accuracy of the incident reports

Table 5.10: Descriptive statistics of the weighted IC and SC Entropies in tickets produced by groups identifying an attack scenario divided by true and false information outcomes

		$H(IC)$			$H(SC)$		
	Num	Mean	Std	Median	Mean	Std	Median
True Info		2.14	1.65	1.73	4.34	2.37	4.05
False Info		5.70	2.81	5.99	4.87	2.27	4.54

issued by the groups for the two attack scenarios as defined in Section 5.4.1.

We then answer to ***RQ3b*** by investigating to which extent technical skills and SOC play a role in producing more true information and fewer false information over the tickets related to the two attacks issued by a group.

Figure 5.6 shows the overall true and false information in tickets for IC and SC as a function of the groups' skill levels and by SOC type.

To evaluate the factors that influence the quality of the tickets, we run a linear regression model over the overall entropy outcomes of the SOC analysts as described in Section 5.4.1.

$$H = \beta_0 + \beta_1 Skills + \beta_2 SOC\ type + \beta_3 Degree \tag{5.9}$$

where:

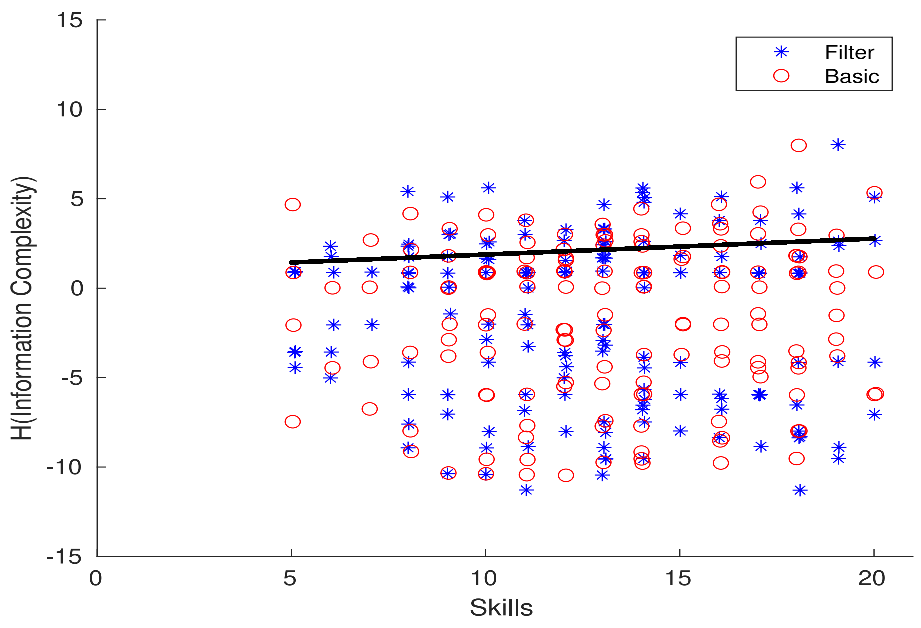
- H is one of $H(IC|T)$, $H(IC|F)$, $H(SC|T)$, and $H(SC|F)$.
- *Skills*: is a numeric value from 0 to 20 that describes the overall skills level of the group. This is obtained via the scoring of the skills assessment questionnaire as the sum of the skills of the group members.
- *SOC type*: is a categorical value that identifies the SOC type of the group (`Filter`, `Basic`)
- *Degree*: is a categorical value that identifies the degree of the group (`Gen`, `Spec`).

Table 5.11 and Table 5.12 show the results.

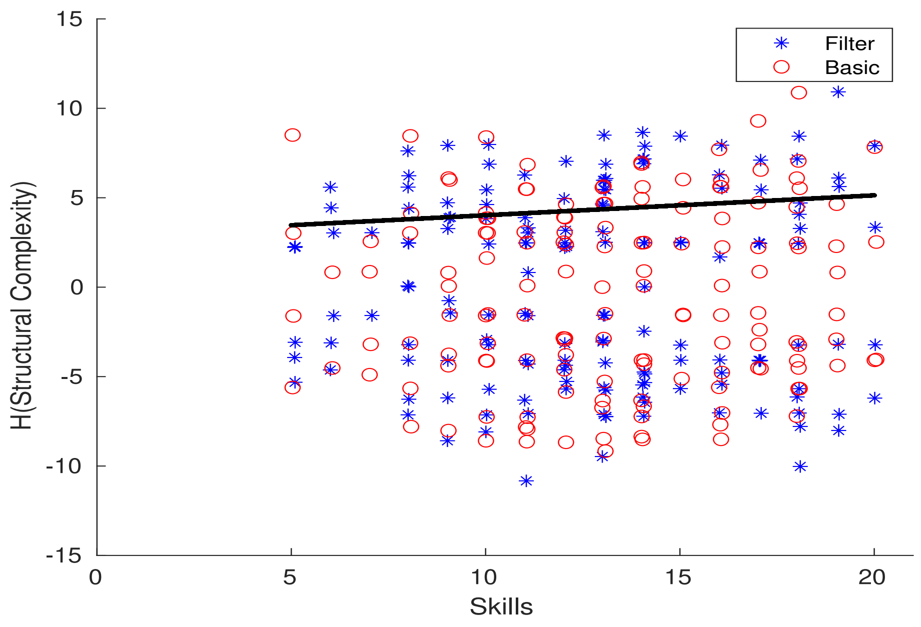
*Skills have a statistically significant effect on correct information in tickets with better information complexity (*IC*) and structural complexity (*SC*). SOC filtering rules have an effect only along the* SC *dimension.*

A possible explanation is that the filtering of irrelevant information by the SOC tool does not help the analyst in the parts of the tasks that require producing new information to continue the investigation. This dimension of complexity is instead influenced by the skills of the analyst in extracting the relevant information from the act and using it for the subsequent acts.

As shown in Table 5.12 skills and SOC do not play a role in reducing the wrong information contained in the tickets and escalated to Tier 2. Interestingly `Spec`-students



(a) Overall true and false information in tickets for Information Complexity (IC). The analysts' skill level is positively related to higher accuracy (true information) in the incident reports



(b) Overall true and false information in tickets for Structural Complexity (SC). The analysts' skill level is positively related to higher accuracy (true information) in the incident reports

Figure 5.6: The overall true and false content in the incident reports of each group by varying skills levels and SOC. For each group corresponds two points with a positive and negative value on the y-axis representing the true and false information outcome respectively

Table 5.11: Regression on total entropy by groups tickets containing true information referring to the attacks

	$H(IC\|T)$	$H(SC\|T)$
c	0.70	2.31$**$
Skills	0.09$*$	0.13$*$
SOC type (`Filter` $= 1$)	0.44	0.81$*$
Degree (`Spec` $= 1$)	0.01	-0.17
R^2	0.06	0.06
F statistic	3.17$*$	3.16$*$

$(*)\ p < 0.05$; $(**)\ p < 0.01$; $(***)\ p < 0.001$

Table 5.12: Regression on total entropy by groups tickets containing false information referring to the attacks

	$H(IC\|F)$	$H(SC\|F)$
c	4.60$***$	4.39$***$
Skills	0.06	0.03
SOC type (`Filter` $= 1$)	0.29	-0.03
Degree (`Spec` $= 1$)	1.12	1.16
R^2	0.04	0.04
F statistic	2.02	2.02

$(*)\ p < 0.05$; $(***)\ p < 0.001$

have a small impact in increasing the false information outcome on the *Structural* dimension. An explanation could be that more skilled and specialized people tend to be more paranoid.

Improving the skills of a SOC analyst or the accuracy of IDS rules do not seem to help in reducing the amount of wrong information from the ticket that is escalated.

We then investigated if the two SOC are equivalent in the amount of false information produced in the tickets related to an attack. From Table 5.12 we have that the overall false information content is not influenced by the SOC type. We ran a Wilcoxon TOST to determine if the accuracy on the two SOC is equivalent and reject the null hypothesis that the amount of false information outcomes in a ticket related to a real attack significantly differs from one SOC to the other:

$$H_{01} = H(complexity|FP \wedge \texttt{Filter}) < \delta_1 H(complexity|FP \wedge \texttt{Basic}) \tag{5.10}$$

$$H_{02} = H(complexity|FP \wedge \texttt{Filter}) > \delta_2 H(complexity|FP \wedge \texttt{Basic}) \tag{5.11}$$

where *complexity* is either IC or SC, δ_1=0.8 and $\delta_2=\frac{1}{0.8}$ as are common values used to compare products [219]. The max of the p values for the two tests is $\max(p_1, p_2)$=0.01 for both H_{IC} and H_{SC}. In other words, the two filterings are statistically equivalent.

An improved filtering mechanism is equivalent to a basic filtering mechanism from the perspective of the false information in tickets potentially related to actual attacks.

Better accuracy in the escalated tickets must be addressed with a mechanism that rewards on-point incident reports issued compared to many reports with little to no value related to the attack. A significantly more advanced filtering mechanism might therefore help in reducing the irrelevant tickets escalated to Tier 2 that are outside of the scope of this work and should be subject to a separate experiment.

5.8 Limitations

The experiment was performed in three runs carried one year apart from each other. The first and third experiments were performed in presence, while the second was performed online due to the restriction of the COVID-19 pandemic. The structure, tasks, and resources available to the students did not change between the runs. For example, the groups could search for any information on the Internet in all three runs. Nevertheless, we controlled if the different set-ups influenced the accuracy by performing a Wilcoxon rank sum test on the total true and false information outcome and no significant difference was found.

The model proposed describes only a part of the Tier 1 SOC analyst activity focusing on the investigation of cyber-attacks to report to a Tier 2 analyst. It does not describe the overall complexity of a SOC analyst's daily activity that includes, among others, the analysis of irrelevant alerts.

In terms of external validity, our results are obtained from experiments with students and not professional SOC analysts. There is a debate on whether the population of cyber-criminals differs from that of students and ethical hackers [63, 162, 163]. In the context of SOC, most Tier 1 positions are entry-level roles [373] for which the student population can be an acceptable approximation.

5.9 Related Work

We present the state-of-the-art related to Security Operations Centers and task complexity.

SOC A critical aspect in a SOC is to measure its performance [180, 321, 322]. Several works tried to investigate the procedure, performance, and needs of SOCs. Shah et al. [294] measured the effectiveness of SOC based on the time spent in an alert investigation and the simulation of different case studies describing the increase of alert arrival rate or decrease of alert service rate. Furthermore, they investigated mechanisms for a fair allocation of SOC resources based on customer agreement and priority of alerts [293]. Ganesan et al. [130] developed an optimization algorithm to allocate analysts to sensors subject to a set of constraints like the analysts' expertise, the number of sensors, and work-shift schedules. In contrast to an analysis of resource allocations on the SOC alerts, we instead focus on the complementary problem of analyzing the accuracy of the investigations on these alerts.

Kokulu et al. [180] identified a set of issues from interviews with SOC analysts and managers. In particular, current performance metrics like response time or the number of tickets issued are not effective in describing the performance of a SOC. Alahmadi et al. [16] found that human analysts are still fundamental to determine the validity of alerts and security tools are often unreliable and lack context for the investigation.

Sundaramurthy et al. [321] performed an anthropological study to determine conflicts in SOCs between the analyst needs and the metrics, tools, and rules imposed by the organization. In a related work [320], they modeled the burnout of SOC analysts in terms of factors related to automation, operational efficiency, and management metrics. In particular, they observed that an incorrect assignment of tasks based on skill levels, e.g. too complex tasks for entry-level or too repetitive tasks for expert analysts, causes burnout. Our model of task complexity can be used to discern complex tasks from repetitive tasks that can be automated. Van Ede et al. [340] developed a semi-supervised deep learning system to automatically correlate security events and reduce alert fatigue for SOC analysts. Chiba et al. [84] developed a tool to prioritize the analysis of domains to reduce repeated investigations.

We lack appropriate metrics to determine SOC analysts' accuracy during the investigation of cyber-attacks.

Modeling Task and Complexity Votipka et al. [349] and Wong et al. [358] investigated the procedures that reverse engineers and malware analysts employ to inspect software through interviews with experts. They modeled workflows performed during the analysis without a measure of task complexity. Mantovani et al. [208] performed an experiment with 72 reverse engineers to measure differences in their strategies. They measured performance in terms of time to obtain the binary flag. They observed that experts skip more blocks by identifying patterns and inspecting functions with a mix of forward and backward techniques. They provided a measure of code complexity using metrics like lines of codes and cyclomatic complexity. Similarly, Aonzo et al. [32] investigated static/dynamic features selection strategies by expert and novice malware analysts and compare them with ML systems. They observed similar choice of features by humans, although the performance significantly differ. However, both works neither measured the complexity of the overall task of reverse engineering (RE) the binary nor employed these metrics to measure the accuracy of the analysis.

Procedures for complex tasks, like malware analysis, lack a model to measure the complexity of their phases that can be used to compare the accuracy of the activity.

Our model of task complexity can be applied in these scenarios to investigate how the accuracy and the analysis process change depending on the complexity of the tasks of RE.

Liu and Li [203] categorized the literature on task complexity based on three viewpoints: structuralist, resource requirement, and interaction viewpoints. In the structuralist (e.g. [59,203,359]) viewpoint, task complexity is obtained by the structure of the tasks as a function of the number of elements required or the interactions between these elements. In the resource requirement viewpoint (e.g. [270]), task complexity is obtained, among the others, by the amount of cognitive, physical, and mental demands needed by the task performer. Instead, in the interaction viewpoint (e.g. [144]), task complexity is seen as a subjective term obtained from the interaction between the task and the task performer characteristics (e.g. knowledge, previous experience).

We lack the application of task complexity theory to complex scenarios related to cyber-security.

In this work, we build upon the structuralist viewpoint, and in particular, we extended Wood model [359] to determine a replicable measure for task complexity for SOC analyst tasks.

Controlled experiments in cyber-security tasks Several works tried to analyze relations between performance in security tasks and factors like years of expertise, background, and education [9,10,19,64,115,241] as well as the cognitive process employed [192]. Rosso et al. [273] built a tool to evaluate SOCs performance via the injection of synthetic attacks into the network traffic. We build upon this open-source tool to measure the accuracy of SOC analyst tickets using an extension of task complexity theory. In the context of secure development, Ruef et al. [279] found that more knowledge of programming languages does not reduce the risk of including security bugs in code. While Braz et al. [64] showed that instructing developers to focus on security issues improves the detection of vulnerabilities. Votipka et al. [348] found that development experience and security training (by means of MOOC participation) have no impact on the type of vulnerability inserted while coding, while it can help in discovering them [350].

We lack controlled experiments for SOC to determine factors that influence the accuracy of investigating cyber-attacks

5.10 Conclusions

In this chapter, we proposed a method to model the accuracy of SOC analyst investigation of cyber-attacks using a model of task complexity. We evaluated our model with three experiments with a total of 368 students to determine the impact on the accuracy of incident reports of the analyst's technical skills and technological changes in the SOC.

Our model can be applied in related tasks regarding malware analysis to evaluate the impact of task complexity in the analysis process [358]. Future works can investigate extensions of the model to deal with dynamic complexity. For example, an Incident Response team needs to interact with a dynamic environment in which the attack evolves over time, also in response to the defender's action [65]. Similarly to [208], the analysis can be extended by collecting more fine-grained data about the interactions of the analyst with the tools and alerts.

Chapter 6

A Graph-based Stratified Sampling Methodology for the Analysis of (Underground) Forums

Researchers analyze security data obtained by scraping underground forums to study abuse and cybercrime activities. Due to the size of the forums and the domain expertise required to identify criminal discussions, most approaches employ supervised machine learning techniques to automatically classify the posts of interest. However, human annotation is costly. How can we select samples to annotate that account for the structure of the forum? In this chapter, we present a methodology to generate stratified samples based on information about the centrality properties of the population and evaluate classifier performance. With this approach, we explicit relations between social network behavior of users and identification of suitables samples for the investigation of online criminal activities. We observe that by employing a sample obtained from a uniform distribution of the post-degree centrality metric, we maintain the same level of precision but significantly increase the recall (+30%) compared to a sample whose distribution is respecting the population stratification. We find that classifiers trained with similar samples disagree on the classification of criminal activities up to 33% of the time when deployed on the entire forum.

6.1 Introduction

Underground forums contain valuable information related to cybercriminal activities. However, this information is hard to retrieve given the large number of unrelated discussions in threads and posts. Current approaches rely on keyword searches and machine learning (ML) algorithms to identify and classify discussions. Many studies [15, 72, 259, 342] use supervised ML algorithms due to the increased accuracy where classifiers are trained on human-labeled data. Human labeling of data is a resource-intensive process, particularly as multiple annotators are required. For cybercrime forums, where jargon and specialised language abounds, annotators also require domain expertise. Therefore, there is a need for identifying the best use of limited resources. The choice of the sample data is a key feature that can impact the performance of the ML classifier. Current approaches randomly pick posts to annotate on a subset of the forum that is promising for the topic to investigate.

In this chapter, we investigate how the performance of the classifier is impacted by sampling methodology. In particular, we propose a methodology to generate stratified samples based on network centrality metrics and compare the classifier performances. We address the following research questions:

***RQ4a*:** What are the changes in performance for a ML classifier using different centrality metrics to generate stratified training samples?

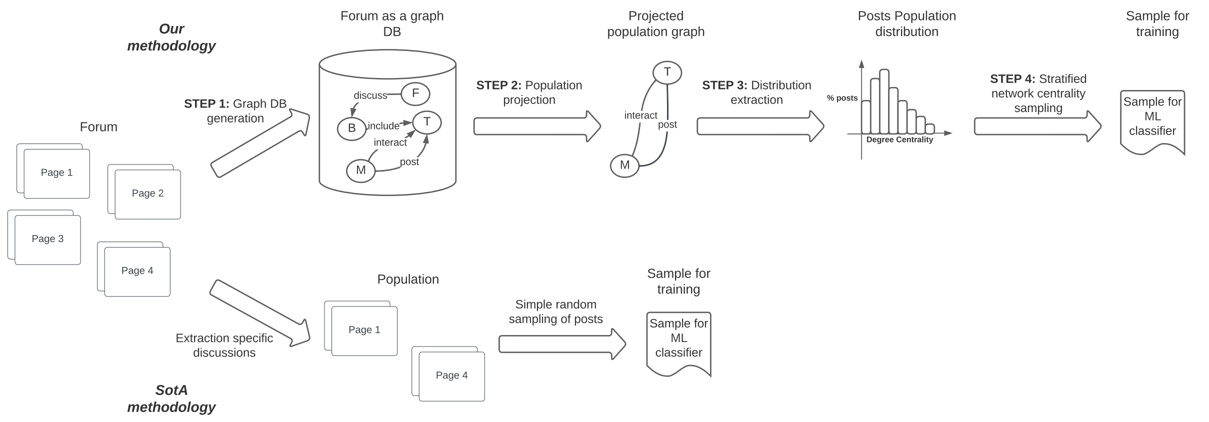
***RQ4b*:** What are the changes in performance using a different proportion compared to the population for the stratified training samples?

In this chapter we make the following contributions:

- A graph DB representing the structure and interactions in an underground forum. We release the anonymized structure to facilitate data analysis and future research. Due to ethical reasons, the access to the actual content stored in the graph (posts, thread, and member names) is subject to a formal data sharing agreement with the Cambridge Cybercrime Centre [1]
- A methodology for the generation of stratified samples based on graph metrics to train ML classifiers and for the validation of their performance on the population.
- An analysis of the impact on ML classifiers performance due to changes in the characteristics of the samples.

Non-goal: We are not interested in tuning the classifiers to obtain the best performance on a given sample. We do not aim to determine pitfalls in the design and implementation of experiments using ML systems [37, 255].

[1]https://www.cambridgecybercrime.uk/process.html


Our methodology
Forum as a graph DB
Projected population graph
Posts Population distribution
Sample for training
STEP 1: Graph DB generation
STEP 2: Population projection
STEP 3: Distribution extraction
STEP 4: Stratified network centrality sampling
Forum
Page 1
Page 2
Page 3
Page 4
discuss
include
interact
post
% posts
Degree Centrality
Sample for ML classifier
Population
Sample for training
Simple random sampling of posts
Extraction specific discussions
SotA methodology

Our methodology consists of: *1) Graph DB generation:* the forums (F), boards (B), threads (T), posts, and members (M) are represented in terms of nodes and edges to highlight relationships. *2) Population extraction:* a subgraph of the forum(s) is identified for the generation of samples for the topic. The resulting subgraph, projected based on some rules, is the population from which the sample is generated. *3) Distribution extraction:* A graph metric is applied to the sub-graph to determine the distribution of the feature on the population. *4) Sample generation:* A sample that respects the stratification is created. In contrast, the current SotA methodology filters the forum for interesting discussions and randomly extracts posts from the identified subset.

Figure 6.1: Methodology for stratified sampling of Forums

6.2 Ethical Considerations

This work requires individuals to read posts on a forum. We note the dataset is collected from the public Internet and is used for research on collective behaviour, without aiming to identify particular members. Given that users in underground forums hide behind a username, it is not possible to obtain consent from users as that would require us to identify them first. In accordance with the Menlo Report [102], we informed the ethics committee so that we could waive the requirement for informed consent. The ethics committee at the Department of Computer Science & Technology, University of Cambridge, considered and approved this research.

6.3 Methodology

We present our methodology to generate stratified samples based on centrality metrics from a population of members in a forum. Figure 6.1 summarizes our overall procedure in comparison with the current state of the art approach.

Step 1: Graph Database generation

We map a forum into a graph $G(V, E)$, in which $v \in V$ can be a board (B), a thread (T), or a member (M) node type and $e \in E$ can be one of the following relationships:

- Forum$\xrightarrow{discuss}$Board : a forum discusses one or more general topics defined in boards. Boards can cover a broad range of topics related to Online Gaming, Cryptography, Reverse Engineering, RATs, etc.
- Board$\xrightarrow{include}$Thread: a board includes one or more member-contributed topics related to the general topic of the board.
- Member$\xrightarrow{post\{content,post_type\}}$Thread: a member posts one or more comments inside a given thread. The comment is defined in the relationship property *content*. The post is also classified for its intent, for example, an offer, a request for services, an exchange, or a tutorial.
- Member$\xrightarrow{interact\{weight\}}$Thread: a member interacts with one or more threads with a certain frequency given by the number of posts as described by the property *weight*.

The schema of the graph DB is reported after the application of Step 1 in Figure 6.1. Nodes of the same type are connected through a different node type. For example, two members are connected if they post on the same thread or if they post in two different threads included in the same board. The *interact* edge is needed only for the graph analysis part. For performance reasons, this type of edge is created when the entire DB is generated. Be A_{ij} the adjacent matrix of *interact* and W_{ij} the matrix of weights associated with each *interact* relationships. For simplicity, in the sequel, we will use directly $G(V, E)$, A_{ij}, and W_{ij} as referring to the selected subset from the entire forum.

Step 2: Population projection

Given a topic of interest for the analysis of cybercrime activities on a forum (e.g. eWhoring [251], marketplace offered goods [301, 342], etc.) a ML classifier must be trained on a sample that contains, among the others, examples of the topic of interest. Given that most activities in the underground forums are legitimate [250] and only a subset deals with criminal activities, the sample is typically obtained from a sub-graph of the entire forum that is promising and representative for the study of the topic. This sub-graph represents the *population* from which a sample is generated for training and testing. The population is obtained following a selection rule, for example by identifying specific boards and threads that deal with the topic of interest via keyword searches.

In terms of graph representation, this approach traduces in a subset of the graph DB based on the selection rule.

Table 6.1: Centrality Metrics for Members node type

Centrality Metric	Formula	Description
Post-degree	$post\_centrality_i = \sum_j W_{ij} * A_{ij}$	Measure the amount of posts for a member i.
Thread-degree	$thread\_centrality_i = \sum_j A_{ij}$	Measure the amount of distinct threads a member i posts into.
Eigenvector	$eigenvector\_centrality_i = \frac{1}{\lambda_{max}} \sum_j A_{ij} * eigenvector\_centrality_j$	Measure the activity of a member i in highly participated threads.

Step 3: Distribution extraction

We compute for each member of the population a value describing its posting activity based on a centrality metric. We then compute the distribution of posts inducted by the member's value on the metric. The post is the unit of the *population* of interest for the analysis of cybercrime activities and composes the training sample for the ML model.

Centrality Metrics over Members

Table 6.1 summarizes the centrality metrics employed in the analysis and the meaning in the context of social network analysis. All metrics are computed for all nodes $v_i \in V$ where v_i belongs to the *Member* node type.

Post-degree centrality measures the amount of activity in terms of the number of posts of the members composing the *population*.

Thread-degree centrality measures the number of distinct threads with which the members interact. This metric differs from the post-degree centrality because it is used to discern members that are mainly active in a few threads from members that interact in different discussions.

Eigenvector centrality measures how much members participate in "hot" (highly participated) threads by looking at the importance of the thread nodes to which a member node is linked.

Distribution induced by centrality metrics

Once the distribution of the centrality metric over the members is obtained, we compute the distribution of posts in the population *induced* by the metric on the members and we normalize the distribution. The resulting distribution describes the percentage of posts associated with members of the population with a certain value for the centrality metric. For all metrics, we observed a few members with extreme values for a metric. For example, few members have a post-degree centrality greater than 10 000.

The skewed distribution induced by the centrality metric can affect the sampling mechanisms. The size of the bins of the distribution must be adjusted to avoid biased sampling due to sample size. Suppose one wants to generate a sample of S posts, the percentage of posts in each bin must be greater than $\frac{1}{S}$ such that at least one post can be picked from each bin. One would want to be able to pick at least 25 elements from each bin to have statistically significant results [13]. To achieve this, we transform the distribution using the logarithm in base 10 and merge together into the same bin posts to have a percentage of the overall sample size greater or equal than $\frac{25}{S}$ posts.

Step 4: Stratified Sample generation

Based on the distribution of posts inducted by a centrality metric on the members, we generate stratified samples whose distribution respect the characteristics of the population. For example, suppose that the distribution of the posts of the population based on the *post-degree* centrality is such that 70% of the posts are obtained from members with post-degree centrality less than 10, 20% of the posts are obtained from members with post-degree centrality less than 100, and 10% with less than 1 000. In particular, we generated 2 types of samples:

- *Proportional Sample*: that presents the same distribution (proportion) of the centrality metric as the population. The sample will be composed of 70% of its size of posts from members that posted less than 10 posts, of 20% of posts from members that posted less than 100, and of 10% of posts from members that posted less than 1 000.
- *Uniform Sample*: that presents a uniform distribution of the centrality metric. Along with the previous example, the new sample will present an equal number of posts from members that posted less than 10, less than 100, and less than 1 000.

The generation of the sample can be subject to more constraints. For example, a maximum number of posts to include in the sample or the need to include available annotated posts (belonging to the population) to reduce the manual effort of the annotation. In terms of annotation, a coding scheme and standardized procedure must be developed to label each sample. Table 6.2 summarizes an example of the coding scheme for Atondo Siu et al.'s [301] crime type classifier, which provides anonymized examples for each class. If some classes are rare and the sample does not include enough posts for them, we ignore

Table 6.2: Guideline Annotation Posts

Crime Type	Description	Anonymized Example
Not criminal	Unrelated to crime. *Including* sharing, selling of games points, skins, etc.	"Xbox One. Comment if you want to play.", "This post is to warm that the user X account was compromised by an hacker that also threat me"
Access to system	Exploitation of vulnerabilities (e.g. SQLi) where there is no innocent usage (e.g. pentesting). *Excluding* the use of malware.	"How to access a phone's text messages and calls without physical access to it."
Bots & Malware	Botnet, malware, and related services. *Excluding* social network bots.	"How to make my server file (of RAT) FUD????"
DDoS & booting	DDoS attack and stress testing. *Excluding* posts selling hosting with DoS protection.	"Would you be interested in investing in a SST service 100% money would be made back plus more."
Spam	Spam, email sharing, or marketing services. The technique employed must be clearly stated (e.g. use of Adfly). *Including* traffic generated, social network bots, request for views and subscribers.	"Earn passive money with clickbank"
Trading credentials	Trading accounts including gaming and social network. *Including* free accounts/credentials. *Excluding* sell of domains, accounts in which the seller is the owner of the domain or service the accounts belong to.	"Selling sickest kik"
VPN & hosting	VPN and hosting services. *Including* requests and offers of VPN.	"I am looking for someone to host OMCPool.net in return for a share in the profits."

these rare classes and label them as part of the main class. Each sample must be independently annotated by at least two annotators to avoid subjective interpretation of the text. A metric like Cohen's or Fleiss's κ must be employed to measure inter-annotator agreement. The posts with different annotations are reevaluated by all annotators jointly. The sample will be used to train the ML classifier.

Validation of ML classifiers performance

Once a ML classifier is trained on a sample, we evaluate its performance using an independent test sample that belongs to the same population.

To compare different sampling strategies, we directly run the classifiers on the entire population to determine the percentage of posts belonging to a class. We extract a random sample from the set of posts in which the two classifiers disagree respecting the stratification, annotate it, and evaluate the performance on that sample only. We then compute the Agresti Coull confidence interval [14] to determine the range of agreement between classifiers on each class. The Agresti Coull CI is used when the sample size is greater than 40 [67]. Differences in performance between the two classifiers are only due to the set of posts in which the classifiers disagree.

6.4 Forum Dataset

We relied on the CrimeBB dataset [252], a database of underground forums available upon request. We focused on `Hack Forums (HF)`, which is the largest and long-lived underground English-language forum, famous for the release of the Mirai botnet source code. We represented the `HF` database in a graph DB using Neo4j[2], a graph database that explicitly represents relationships between entities. The forum contains ≈680k members, with ≈42M posts over more than 4M threads.

We observed a user, likely the forum administrator, that presented an extreme number of posts in different threads. We observed that the posts were related to managing the forum and did not add any value to the analysis. We thus removed this member from the analysis. We further considered all threads and posts up to June 2018 to compare with a sample obtained from the related works [301].

6.5 Analysis

We computed the centrality metrics using the Neo4j Graph Data Science (GDS) Library. The GDS library exploits an in-memory graph projected from the DB to efficiently run graph algorithms on large graphs.

We evaluate the performance of the classifiers over samples obtained using our methodology on different centrality metrics and compare the performance with a sample obtained using random sampling from Atondo Siu et al. [301] from the same period of time. To reduce the manual effort of the annotation, we constrained the generation of the new samples by keeping as many entries from the random sample as possible that respect the distribution of the centrality metric considered.

[2]https://neo4j.com/

Table 6.3: Population graph Statistics

Selection Rule	# Nodes	# Edges
Subset of posts and members that are classified as 'Offer','Request','Exchange', and 'Tutorial' by [72]	$\approx$447k (member), $\approx$3M (thread)	$\approx$11.3M (post), $\approx$9.6M (interact)

We first identified the population from which the random sampling of posts from [301] has been extracted to compute the population centrality metrics. The sample in [301] is composed of several samples:

- *General HF Random Sample:* 500 posts extracted randomly from the entire HF.
- *Trading HF Random Sample:* 1 500 posts extracted randomly from all posts in HF classified as 'Offer', 'Request', 'Exchange', and 'Tutorial' by Caines et al. [72][3].
- *Currency HF Random Sample:* 2 000 posts extracted randomly from all posts in HF from members that published at least one post in the *Currency Exchange* board.

We focus the analysis on the *Trading HF Random Sample* because it can be easily described by the application of a selection rule and it is general enough to include discussions on different crime topics. We thus identified the population from which the sample was obtained:

- *Trading HF Population:* The subgraph obtained by the threads in HF and their *subset* of posts and members that are classified as 'Offer', 'Request', 'Exchange', and 'Tutorial' by Caines et al. [72].

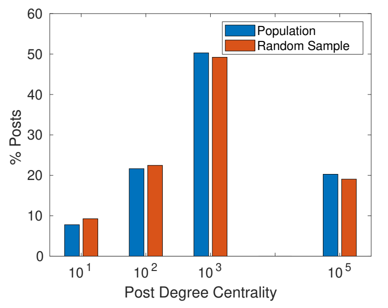
Table 6.3 summarizes the selection rule for the block and the characteristics of the graph population. Table 6.4 summarizes the samples for the analysis and their sampling strategy.

Figure 6.2 shows the distribution of the centrality metrics for the posts of the *Trading HF* population and the 1 500 posts of the Trading HF random sample from Atondo Siu et al. [301]. Each bin contains the posts associated with members whose centrality metric is strictly less than the bin value on the x-axis. The Trading HF random sample distribution is already similar to the population distribution because the sample size is large enough. For example, the probability p that a post belongs to a member with less than 10 posts is $\approx$8% (Figure 6.2a) for the population. Thus, the standard error of the sample proportion is given by $\sqrt{\frac{p(1-p)}{n}}$=0.0137 for n=1 500. We expect the sample proportion to be within

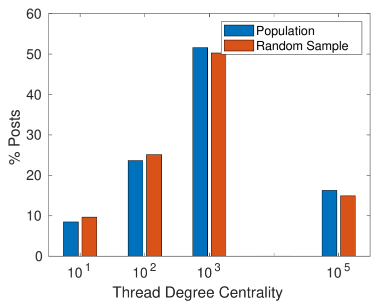
[3]The sample for training used in [72] is based on the following boards: *Beginner Hacking*, *Premium Sellers*, and additional 13 boards chosen at random (*Computer and Online Gaming; Cryptography and Encryption Market; Decompiling, Reverse Engineering, Disassembly, and Debugging; Domain Trading; Ebook Bazaar; HF API; Marketplace Discussions; Remote Administration Tools; Secondary Sellers Market; Shopping Deals; Web Browsers; Windows 10; World of Warcraft*)

Table 6.4: Samples from Trading HF Population

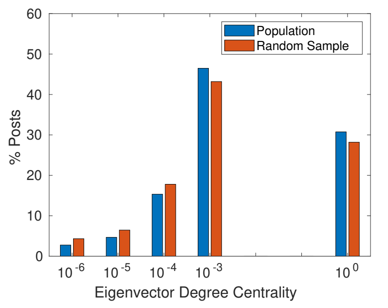
Sample Name	Sampling Strategy	# Posts
Trading HF random	Simple random sampling from population	1 500
Post-degree Proportional	Proportional stratified sampling from post-degree population distribution	1 500
Thread-degree Proportional	Proportional stratified sampling from thread-degree population distribution	1 500
Eigenvector Proportional	Proportional stratified sampling from eigenvector population distribution	1 500
Post-degree Uniform	Uniform stratified sampling from post-degree population distribution	1 500



(a) Distributions Post-degree centrality



(b) Distributions Thread-degree centrality



(c) Distributions Eigenvector centrality

Figure 6.2: Distribution of population and Trading HF random sample over different centrality metrics. The distribution is obtained by merging together bins to obtain bins with enough posts to be sampled given the sample size. All posts in a bin belong to members whose centrality metric is strictly less than the value on the bin.

0.92±0.0137 and that is the case for our Trading HF random sample that presents a probability $\hat{p}$=0.908.

6.5.1 Annotation and Classes

We obtained the 1 500 posts sample of the Trading HF random sample from Atondo Siu et al. [301] and used the same coding scheme (summarized in Table 6.2). We manually classified the posts in one of the classes of crime as described in Table 6.5. Atondo Siu et al. [301] considered a larger set of criminal types. However, by looking at the classes in Atondo Siu's sample, we did not find enough instances for all classes[4]. For

[4]Indeed the analysis in [301] is performed with the addition of the 500 posts from the *General HF* and the 2000 posts from the *Currency HF* random samples

Table 6.5: Crime type classes and # posts per sample

Crime Type	Description	Trading HF random	Post-degr. Propor.	Thread-degr. Propor.	Eigenvector Proport.	Post-degr. Uniform
Not criminal	Unrelated to crime	1041	1048	1050	1055	983
Access to system	Exploitation of vulnerabilities (e.g. SQLi)	57	56	53	56	78
Bots & Malware	Bots, malware, and related services	157	156	154	144	164
DDoS & booting	DDoS attack and stress testing	59	59	57	62	63
Spam	Spam, email sharing, or marketing services	46	43	46	42	61
Trading credentials	Trading accounts	106	104	105	103	115
VPN & hosting	VPN and hosting services	34	34	35	38	36

a first approximation, we ignored these rare classes and classify them as *not criminal*. This approximation does not affect the comparison of the performance between samples because all samples are annotated with the same annotation procedure. For all stratified samples, we reused as many posts as possible from the Trading HF random sample to reduce the manual effort.

Three researchers independently annotated the posts for the post-degree, thread-degree, and eigenvector *proportional samples*. Two of them were involved in the initial annotation of the Trading HF random sample. The new annotated posts for each sample were 34, 39, and 90 respectively. We report Fleiss's κ that measures the agreement among multiple annotators. The Fleiss's κ ranges from 0.68 to 0.79. A value greater than 0.6 indicates a substantial agreement [187]. Two researchers further annotated 432 posts to generate the *uniform samples* for the post-degree centrality metric. We report Cohen's κ that measures the agreement between two annotators. Cohen's κ is 0.74 for the uniform post-degree sample.

6.5.2 Training ML models

We evaluated the performance using the XGBoost model, a state-of-the-art model that showed promising results in classifying posts in underground forum [15, 72, 301] and it is less subject to overfitting of training data compared to other classifiers [301]. We consider as input to the classifier a text composed of the post content, the thread title, and the board title from the forum. We pre-processed the text to convert capitalized letters to lowercase, remove stop-words, tokenize the input, and lemmatize the words using the NLTK library [6]. We extract a vector of lexical features using *tf-idf*. Given that most posts in the sample are classified as *not criminal*, we re-sampled the training set using SMOTE to overcome imbalances among classes.

6.6 Performance

We now report the ML classifier performance trained on the different samples and apply our methodology to validate their performance on a representative sample of the population. For the crime type classes *DDoS* and *Spam* we observed that the classifier did not predict any post in 53% of the repeated stratified holdout for the Trading HF random sample due to the inability of the classifier to recognize samples of these classes. Similarly for the Post-degree, Thread-degree, and Eigenvector proportional samples, where classifier did not predict any post as *DDoS* or *Spam* in 36%, 46%, and 26% of the stratified holdout runs respectively. This behavior is probably due to a lack of posts for these classes in the sample that allows the classifier to extract significant characteristics. Any attempt to retrospectively get more data on those classes will bias the sample by construction and thus, to avoid unfair performance comparison, we ignored *DDoS* and *Spam* in the analysis.

Proportional Samples via Centrality Metrics

We now address ***RQ4a*** by looking at the performance of the classifier using the different centrality metrics to generate training samples. We compare the performance of the ML models using as the training set the Trading HF random sample from Atondo Siu et al. [301] and the post-degree, thread-degree, and eigenvector *proportional* stratified samples. For testing we used a new 500 posts test sample obtained randomly from the population, annotated by two researchers (Cohen's κ=0.75). We performed a repeated stratified holdout for the training and test sample with 30 different random seeds and average the results using the geometric mean. The geometric mean is best suited to summarize ratios [123]. The stratification in the holdout depends on the sample used for training. For example, when using the Trading HF random sample, the stratification is based on its class distribution[5], when using the post-degree sample the stratification is based on the population post-degree centrality distribution, and similarly for the other samples.

Table 6.6 shows the performance and the relative change considering the Trading HF random sample as the reference value in performance. From the results, we did not observe significant differences in the overall precision and recall compared to the Trading HF random sample. This can be explained by the fact that the Trading HF random sample already presents a distribution similar to the population distribution for all centrality

[5]This is the common approach performed by the state-of-the-art.

Table 6.6: Comparison XGBoost performance using Trading HF random and proportional stratified samples

	Precision				Recall			
Class	Random	Post-degr.	Thread-degr.	Eigenvector	Random	Post-degr.	Thread-degr.	Eigenvector
Not Criminal	0.78	0.77	0.78	0.78	0.93	0.94	0.93	0.93
Access to system	0.54	0.52	0.57	0.51	0.24	0.24	0.24	0.23
Bots & Malware	0.65	0.68	0.61	0.67	0.53	0.51	0.54	0.54
Trading credentials	0.59	0.61	0.64	0.60	0.58	0.56	0.57	0.53
VPN & hosting	0.54	0.54	0.54	0.54	0.23	0.23	0.23	0.27
Geometric Mean	0.61	0.62	0.62	0.61	0.44	0.43	0.44	0.44
Relative Change	/	+1.64%	+1.64%	+0.00%	/	-2.27%	+0.00%	+0.00%

Table 6.7: XGBoost performance using Post-degree uniform sample and relative change compared to Post-degree proportional sample

Class	Precision	Recall
Not Criminal	0.83	0.89
Access to system	0.52	0.40
Bots & Malware	0.64	0.66
Trading credentials	0.59	0.64
VPN & hosting	0.54	0.36
Geometric Mean	0.62	0.56
Relative Change	+0.00%	+30.23%

metrics (see Figure 6.2) and the change in the number of posts compared to the Trading HF random sample is small (2.2% for the post-degree, 2.6% for the thread-degree, and 6% for the eigenvector *proportional* stratified sample respectively).

Proportional vs Uniform Sample

To address ***RQ4b*** and determine if the distribution of the centrality metric plays a significant role in the ML performance, we trained the XGBoost model with a sample that significantly differs in the distribution of a centrality metric compared to the population. We generated using our methodology (Section 6.3) a *uniform* sample based on the post-degree centrality metric and compared it with the performance of the model trained with the same centrality metric but using the *proportional* sample.

Table 6.7 show the results and the relative change compared to the proportional sample, whose results are reported in Table 6.6). We observed that the overall precision using the uniform sample is the same as the proportional sample but in contrast, the recall significantly improves (+30.23%).

Table 6.8: XGBoost Agreement Trading HF Random and Post-degree proportional classifiers.

Class	#Agree	#Random only	#Proportional only	CI Agreement
Not Criminal	9 219 317	213 491	294 270	(0.947,0.947)
Access to system	120 068	25 986	32 058	(0.671,0.676)
Bots & Malware	561 925	103 996	67 620	(0.765,0.767)
Trading credentials	715 415	138 050	80 540	(0.765,0.766)
VPN & hosting	123 767	22 998	26 924	(0.710,0.714)

Agreement between classifiers

The differences in performance between the Trading HF random and the post-degree proportional stratified sample are relatively small if one only looks at Table 6.6. However, this difference can be significant when the classifier is deployed to classify an entire forum with millions of posts. We investigated the real impact of this small variation by running the classifiers trained on the Trading HF random and post-degree *proportional* sample on all population posts (Table 6.3). We then computed the proportion of posts that were classified the same by both classifiers and the Agresti Coull CI to determine the range of agreement. Table 6.8 shows the per-class agreement and disagreement for the Trading HF random sample and post-degree *proportional* sample. The agreement for the crime type classes ranges between 67% and 76%, thus although trained with very similar samples, the two classifiers differ up to 1 out of 3 posts for certain crime type classes when deployed on the entire forum.

We provide some anonymized examples of posts in which the Trading HF random and post-degree stratified sample disagreed. Table 6.9 shows the anonymized examples, the classification of each classifier, and the annotation.

We now investigate the performance on those posts in which the two classifiers disagree. To investigate the disagreement and determine which classifier performed better, we randomly picked for each class, 100 posts for which the classifiers disagreed. A total of 500 posts were manually re-annotated to create a new test set to measure the performance of the classifiers on the disagreement (Cohen's κ=0.80). This test sample allows one to investigate the region in which the boundaries of the classifier change due to the sample characteristics. Table 6.10 summarizes the performance of the classifiers on the test sample extracted from the disagreement posts. Although the overall performances are similar, in contrast to what was observed in Table 6.6 we have more significant differences in the precision and recall for certain crime type classes. For example, the Trading HF random sample has better precision for *VPN & hosting*, while the post-degree proportional sample

Table 6.9: Examples of Disagreement between trained models

Example	Post-degree Centrality	Thread-degree Centrality	Eigenvector Centrality	Classification Random	Classification Post-Degree	Annotation
"Help Keylogger. If you crypted you risk to made it unstable. It must be FUD to not be detected"	$3.8 * 10^2$	$3.5 * 10^2$	$2 * 10^{-4}$	Bots & Malware	Not criminal	Bots & Malware
"To have a botnet: take a mp3 file, infect it and upload on a hosting to download"	$7.0 * 10^1$	$6.8 * 10^1$	$3.3 * 10^{-4}$	VPN & hosting	Bots & Malware	Bots & Malware
"This post is to warm that the user X account was compromised by an hacker that also threat me"	$1.1 * 10^3$	$9.4 * 10^2$	$1.6 * 10^{-3}$	Not criminal	Trading credentials	Not criminal
"Hello, I updated this program to create a relaxing game. This is the virus scan for the program. Enjoy"	$5.3 * 10^2$	$3.8 * 10^2$	$2.1 * 10^{-4}$	Bots & Malware	Not criminal	Not criminal
"Hacking using IP. Where is the scanner link? Do I need to find it on the website?"	3	3	$6.04*10^{-7}$	Not criminal	Access to system	Access to system
"I have a social network md5 password. I need to decrypt it. Any idea?"	$1.5 * 10^1$	$1.5 * 10^1$	$1.8 * 10^{-4}$	Access to system	Bots & Malware	Access to system
"Hosting service - DDoS protection and VPS hosting with 24/7 support. We accept requests. You will enjoy it"	$1.7 * 10^2$	$1.4 * 10^2$	$1.2 * 10^{-3}$	Not criminal	VPN & hosting	VPN & hosting
"I am using these Proxies so I share with you [IP addresses]"	$1.3 * 10^1$	$1.2 * 10^1$	$6.2 * 10^{-4}$	VPN & hosting	Not criminal	VPN & hosting
"How many Netflix accounts can you sell for 5 cents?"	$1.3 * 10^2$	$1.1 * 10^2$	$1.8 * 10^{-4}$	Trading credentials	Not Criminal	Trading credentials
"I have thousands of accounts in [SOCIAL NETWORK] that have many contributions as requested by you"	$1.5 * 10^2$	$1.4 * 10^2$	$2.3 * 10^{-5}$	Not criminal	Trading credentials	Trading credentials

has better precision and recall for the *Trading credentials*. Given that the performance on the posts in which the classifiers agreed is the same, either both are right or both are wrong, we expect the corresponding classifiers to perform better in these classes when dealing with the entire forum because in the regions where the classifiers differ they have better performance. For example, the classifier trained with the post-degree proportional sample incorrectly classified as *VPN & hosting* the following anonymized example: "I

Table 6.10: XGBoost Disagreement Sample Performance.

	Precision		Recall	
Class	Random	Post-degree Proportional	Random	Post-degree Proportional
Not Criminal	0.36	0.39	0.53	0.57
Access to system	0.39	0.38	0.33	0.33
Bots & Malware	0.58	0.61	0.53	0.43
Trading credentials	0.41	0.56	0.49	0.65
VPN & hosting	0.68	0.53	0.47	0.49
Geometric Mean	0.47	0.48	0.46	0.48
Relative Change	/	+2.13%	/	+4.35

made lots of money by hosting Minecraft on a server I rent", while the classifier trained with the Trading HF random sample correctly did not classify it as related to VPN or hosting. Conversely the classifier trained with the Trading HF random sample classified as *Trading credentials* the following anonymized example: "Why would you want to unverify a Paypal account?", while the classifier trained with the post-degree proportional sample correctly did not classify it as related to trading credentials.

6.7 Limitations

In Section 6.5 we relied on a previously obtained classification of post types to identify the population of interest (*Trading HF* population). The population is thus affected by the precision and recall of the Caines et al. [72] classifier. However, this does not influence the overall results because the centrality metrics are computed by considering classified posts as the entire and only population.

The presence of time biases in which future posts are used to predict previous posts [255] is not an issue because the aim is to classify the entire set of posts available at a certain point in time.

The classifiers trained with samples are not tuned to obtain the best achievable precision and recall thus the results are not comparable with the related work that aim at proposing the best-fitted classifier on a given dataset.

6.8 Related Work

Analysis of Underground Forums Several works analyze underground forums to study specific areas of cybercrime. Common approaches rely on natural language processing (NLP) and supervised ML using random samples for training and testing.

In the context of the classification of posts, Portnoff et al. [259] proposed an automated classification of post type (buy, sell, and exchange currency), product offered/requested, and price. They also evaluated the performance of the supervised classifier by training and testing over eight forums. They observed that the tool performance significantly drops if used across forums. Similarly, Caines et al. [72] evaluated the performance of different statistical models and heuristics on labeling post type, author intent, and addressee in `Hack Forums`. Van Wegberg et al. [342] trained a SVM classifier to identify the type of listings (e.g. cash-out, malware, remote access tools (RATs), accounts, etc.) discussed in eight marketplaces and determine their associated revenue. Atondo Siu et al. [301] trained a supervised ML systems to classify posts in `HF)` into a class of crime (*Non-criminal*, *Access to system*, *Bots & Malware*, *eWhoring*, *Currency Exchange*, *DDoS*, *Identify Theft*, *Spam*, *Trading credentials*, *VPN*) and analyzed the digital currency utilized in each class. They observed that there was a massive shift to Bitcoin after Liberty Reserve was taken down, and there was a demand for exchanging PayPal.

In the context of cybercrime-as-a-service (CaaS), Akyazi et al. [15] measured the typology of CaaS services in `HF` via a supervised ML classifier and observed only a few CaaS categories discussed extensively (botnet, reputation escalation, and traffic-as-a-service). Sun et al. [318] investigated Concession-Abuse-as-a-Service and performed an investigation of the techniques employed in four underground forums. Bhalerao et al. [47] analyzed business-to-business interactions in two underground forums (`HF` and `Antichat`). They trained supervised ML classifiers to determine the product offered and the reply class (buying, selling, or other). From this classification, they built an interaction graph between members to determine the presence of supply chains in the criminal markets.

Several studies focus on predicting criminal activities based on forum discussions. Pastrana et al. [250] developed a SVM to classify posts of key actors in `HF` and predict potential which actors might be of interest to law enforcement. Van Wegberg et al. [341] investigated vendors and product characteristics to predict product success via regression analysis. They observed a positive correlation with features like the presence of a refund policy, customer support, and the use of vendor names. Sun et al. [319] studied the differences between private and public messages in underground forums and developed ML classifiers to predict presence of private interactions from public features. Yuan et al. [365] developed a tool to automatically identify new dark jargon in underground forum posts using NLP.

The availability of forum discussions over several years allows researchers to investigate the evolution of these ecosystems. Soska and Christin [305] performed a longitudinal analysis of 16 online marketplaces for two years to estimate the sales volume and the type

Table 6.11: SotA on Sampling Technique for supervised classification of criminal activities in forums

Sampling Technique	Papers
Simple Random	[342], [250], [72], [319], [47], [251], [301], [15], [318]*
Unspecified	[259], [305]
Stratified using network centrality metrics	Our work

* Stratified Random sampling

of products exchanged. They observed significant gross income for big marketplaces like Silk Road in the order of hundreds of thousands of dollars per day. Furthermore, they observed that the volume of sales was not significantly impacted by the law enforcement take-downs. Pastrana et al. [251] investigated eWhoring activities on underground forums for more than ten years. They trained a ML classifier to identify threads offering 'packs' and then determine the origins of the images, the main actors actively engaged in this activity, and their profits. They built a social network graph of members active in eWhoring discussions and identified the key actors using the h-index and eigenvector centrality. They observed how the interest of these users moves from gaming and hacking to market-related topics after the interaction in eWhoring threads. Allodi [17] investigated exploits traded in a Russian black market and their likelihood of exploitation in the wild. Vu et al. [351] performed a longitudinal analysis of the trading activities in HF and their evolution over different 'eras'. They observed how currency exchange and payments account for the majority of the contracts and payments are performed using Bitcoin and PayPal.

Table 6.11 summarize the papers by the sampling techniques employed to train classifiers.

Current approaches for supervised ML training rely on a sample obtained through random sampling from interesting discussions and do not employ information of the "population" of the social network.

Social Network Analysis of Underground Forums Several works investigate the properties of online social networks to identify influential actors and analyze topics of interests [160,250,251,351,370]. Motoyama et al. [224] analyzed social network dynamics from leaks of 6 underground forums. In particular, they generate social network relationships based on 'friend' requests, private messages, and thread discussions. They evaluated how the social degree of these relations impacts their trading. Garg et al. [131] employed *social network analysis* (SNA) to determine sub-communities inside forums, the correlation between centrality metrics for members, and the impact on the network structure of banned members. Pastrana et al. [250] employed SNA, logistic regression, and clustering to identify members that interact with known criminals and predict their likelihood of being involved in future criminal activities. Almukaynizi et al. [23] employed social network metrics as features to predict the exploitation of vulnerabilities discussed in forums.

SNA is extensively used to identify key actors in underground forums [159, 250, 367]. Our centrality metric approach is similar to Pete et al. [256], who constructed undirected graphs of six underground forums based on 6 months of observation, computed network statistics, and analyzed the network structure. Using centrality metrics they identified important members in the network and performed a qualitative analysis of the topics covered. In contrast to our work, their focus is on insights into the structures of different small forum snapshots. We instead propose a methodology to identify relevant samples for training ML classifiers based on the characteristics of the population of the entire forum, which can span several years of observation.

Social Network Analysis in underground forums is mainly focused on identifying key actors and field experts and how social relations influence the criminal activities in these forums. We lack an analysis of the use of social network characteristics, like centrality, to select representative posts from the population.

Sample Representativeness in Online Social Network Prior work has analyzed sampling techniques for online social networks to recover the properties of the network in case the entire social network cannot be used and a sample must be extracted (e.g. via Twitter API [223]) or to extract representative samples or samples with specific characteristics, e.g. high degree centrality [207]. These sampling techniques rely on random node extraction, random edges extraction, exploration via random walks [139, 368], and snowball sampling. Studies investigated how the different sampling techniques conserve the ranking of nodes, the visibility of groups [352], how robust different centrality metrics are [89] and proposed variants to preserve the properties of the original network, in particular ratio of nodes and edges and topology, based on hierarchical community and densification power law [364].

This line of research focuses on extracting a representative subgraph from a large network by trying to maintain the unknown properties of the population.

We are focusing on extracting sample elements exploiting known network characteristics of the entire population to improve ML classification.

6.9 Conclusions

We presented a methodology to generate new samples exploiting information about the centrality properties of the population and evaluate the performance of the classifiers. We observed a significant increase in recall using a uniform distribution of the post-degree centrality metric to generate the training sample. Other centrality metrics like thread and eigenvector did not differ in the overall performance. We also observed that the agreement between classifiers trained with similar samples can significantly disagree in their classification when the classifiers are deployed on the entire underground forum. We leave for future work further analysis using other distribution and centrality metrics like, for example between centrality, or other graph analysis techniques like for example clustering.

Chapter 7

Software Updates Strategies against Advanced Persistent Threats

Software updates reduce the opportunity for exploitation. However, since updates can also introduce breaking changes, enterprises face the problem of balancing the need to secure software with updates with the need to support operations. In this chapter, we propose a methodology to quantitatively investigate the effectiveness of software updates strategies against attacks of Advanced Persistent Threats (APTs). We modeled interactions between APTs behavior in terms of attack vectors and software vulnerabilities exploited and speed of deployment of updates. This allows one to investigate what-if scenarios for updating organizations software. We extracted security data from vulnerability, software repositories, and threat intelligence reports to create a manually curated dataset of APT attacks covering 86 APTs and 350 campaigns from 2008 to 2020. The dataset includes information about attack vectors, exploited vulnerabilities (e.g. 0-days vs public vulnerabilities), and affected software and versions. Contrary to common belief, most APT campaigns employed publicly known vulnerabilities. If an enterprise could theoretically update as soon as an update is released, it would face lower odds of being compromised than those waiting one (4.9x) or three (10.5x) months. However, if attacked, it could still be compromised from 14% to 33% of the times. As in practice enterprises must do regression testing before applying an update, our major finding is that one could avoid ≈33% of all possible updates restricting oneself only to versions fixing publicly known vulnerabilities without significant changes to the odds of being compromised compared to a company that updates for all versions.

7.1 Introduction

A recent study [184] shows that it takes more than 200 days for an enterprise to align 90% of their machines with the latest (not known to be vulnerable) software version given the need to perform regression testing [249].

Such behavior is rational because not all vulnerabilities are always exploited in the wild [62], and several authors have determined that the actual risk of slow updates against 'mass attackers' is limited [20, 51] and often due to specific types of vulnerabilities such as those traded in Black Markets [17] or with other predictable characteristics [82, 167]. Hence, risk analysis might be an effective approach when considering 'mass attackers' which might well be 'work averse' and stick to old exploits until they are no longer profitable [22]. However, many companies also face Advanced Persistent Threats (APTs). APTs are highly specialized professionals [275] that use a variety of customized strategies [195], often leveraging on spearphishing [337] and 0-days [83, 209] to maintain a stealthy profile [83]. In this scenario, slow updates do not seem appropriate.

Yet, not all the APTs are really *sophisticated* [314]. Some reports challenged some of these 'allegations' and observed that APTs often reuse tools, malware, and vulnerabilities [90,200,337]. These reports are based on threat intelligence data with few overlaps [61] thus capturing only partial information of the APT attacks [197, 314].

These conflicting claims may be due to the lack of a systematic study. Indeed, previous works on APTs analysis [83, 275, 338, 347] mostly reported a *qualitative* analysis of a handful of APTs. However, relying on qualitative estimations is known to produce risk miscategorization and wrong prioritization [29, 186] due to several factors like judgmental biases [186], agenda-setting, and framing [290]. Framing of individual reports can produce a distorted perception of the risk. We lack a broad view of the APT landscape that allows companies to correctly assess the advantages and disadvantages of current approaches to software updates. Data acquisition of APT campaigns, i.e. specific attacks conducted by APT groups, is currently a challenging task. Semi-automated approaches based on report parsing [189] proved to be too riddled with false positives because the associations between APTs and software vulnerabilities (identified by a CVE) are based on the presence of keywords and not on the semantics of the document. Here our research questions are as follows:

***RQ5a*:** What are the APTs characteristics that quantitatively describe the landscape of APT campaigns as observable from public reports?

***RQ5b*:** Given a quantitative description of both APT campaigns and software updates, how effective are different update strategies to protect against APT campaigns?

In this chapter we make the following contributions:

- We build a structured, manually verified database in Neo4j of 86 APTs and more than 350 campaigns based on an exhaustive search of over 500 technical reports and blogs and up to 22 different resources for each APT. The database [330] is available on Zenodo
- We present a methodology to quantify and compare the effectiveness and cost of software update strategies on historical data about campaigns
- We quantitatively evaluate the effectiveness and cost of different software updates strategies, in terms of the *conditional* probability of being compromised and the number of updates required for 5 widely used software products (*Office*, *Acrobat Reader*, *Air*, *JRE*, and *Flash Player*) for the Windows O.S.

Goal: We provide a quantitative analysis of the risk against APT to allow companies to make rational decisions on software updates.

Non-goal: We do not propose new mechanisms to detect and mitigate APTs attacks.

7.2 The Software Update Problem

If a company could only update for new functionalities, the choice would be obvious: why fixing what is not broken? Yet, companies must update for security reasons too. However, it is not uncommon that vulnerability fixes are merged with feature changes in a single update. Every time a new version of a software is published, one can

- *update* immediately;
- *wait* some time (e.g. for regression testing) and update;
- *skip* the update.

This choice may be influenced by asynchronous events related to the *reservation*, *disclosure*, and *exploitation* of software vulnerabilities in the *current* release.

Unfortunately, a company cannot fully decide *in advance* the configuration they will have when hit (or most frequently not hit) by an attacker as it depends on the attacker's choice. A company can only decide on the software updates strategy. To capture what can happen we introduce some terminology.

Table 7.1: Classification of Attack Scenarios

Scenario	Description
Unknown-Unknown/ Unpreventable (UU/U)	The vulnerability is exploited before a CVE was reserved, before its public disclosure, and before an update was released.
Unknown-Unknown/ Preventable (UU/P)	The vulnerability is exploited before a CVE was reserved, before its public disclosure, but after an update was released.
Known-Unknown/ Unpreventable (KU/U)	The vulnerability is exploited after a CVE was reserved, before its public disclosure, and before an update was released.
Known-Unknown/ Preventable (KU/P)	The vulnerability is exploited after a CVE was reserved, before its public disclosure, and after an update was released.
Known-Known/ Unpreventable (KK/U)	The vulnerability is exploited after a CVE was reserved, after its public disclosure, and before an update was released.
Known-Known/ Preventable (KK/P)	The vulnerability is exploited after a CVE was reserved, after its public disclosure, and after an update was released.

Terminology

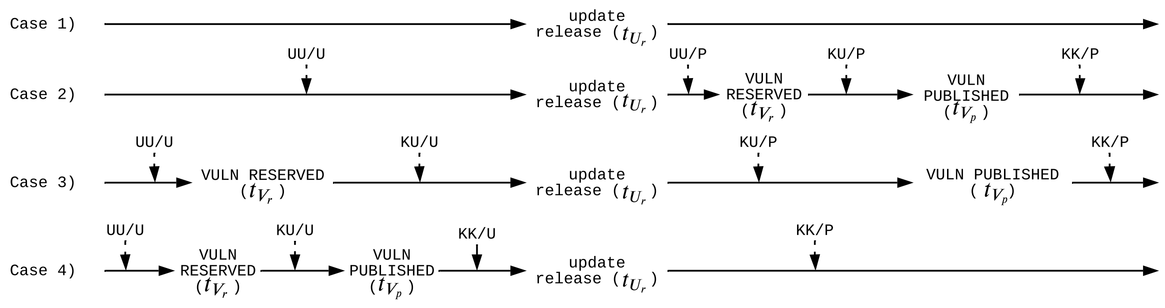
For each vulnerability we identified five instants of time:

- *Vulnerability Reserved time (t_{V_r})*: when the CVE entry for the vulnerability is reserved by MITRE;
- *Vulnerability Published time (t_{V_p})*: when the CVE for the vulnerability is published in NVD;
- *Vulnerability Exploited time (t_{V_e})*: when the vulnerability is observed to be exploited in the wild;
- *Update release time (t_{U_r})*: when an update that addresses the vulnerability is released.
- *Update deployed time (t_{U_d})*: when an update that addresses the vulnerability is deployed.

Table 7.1 shows how we can classify attack scenarios based on the instant of time t_{V_e} and its relative position with the other events: t_{V_r}, t_{V_p}, and t_{U_r}. Figure 7.1 summarizes the possible combinations of the different events.

7.2.1 The Software Update Strategies

To answer ***RQ5b***, we describe the update strategies, summarized in Table 7.2, for an enterprise based on what was discussed previously: *update*, *wait* and then update, or *skip*. It is important to underline that disabling automated updates is not uncommon in enterprise networks [226, 362]. This is mainly due to compatibility issues between the updated software and internal projects [57, 249] that can produce disruption of the


Case 1)
update release (t_{U_r})
Case 2)
UU/U
update release (t_{U_r})
UU/P
VULN RESERVED (t_{V_r})
KU/P
VULN PUBLISHED (t_{V_p})
KK/P
Case 3)
UU/U
VULN RESERVED (t_{V_r})
KU/U
update release (t_{U_r})
KU/P
VULN PUBLISHED (t_{V_p})
KK/P
Case 4)
UU/U
VULN RESERVED (t_{V_r})
KU/U
VULN PUBLISHED (t_{V_p})
KK/U
update release (t_{U_r})
KK/P

At the time when a software update is available, we have 4 cases: *Case 1)* there is no reservation and publication of vulnerabilities before and after the release of an update for the current version. In this case, there is no exploitation of the vulnerability. *Case 2)* after a software update is released, a vulnerability is reserved and disclosed for the current version. *Case 3)* before the release of a software update a vulnerability is reserved for the current version, but the disclosure happens after the update release. *Case 4)* the reservation and disclosure of a vulnerability for the current version happen before the release of an update. Different update strategies can be applied but are all constrained by the presence of a new release. The exploitation events (vertical lines) can happen at any instant of time asynchronously from the reservation-disclosure process and the release of updates. They are classified following Table 7.1

Figure 7.1: Combinations of vulnerability reservation, disclosure, and exploitation events with the presence of new updates.

enterprise work. In this case, delays are introduced to perform regression testing.

We considered the application of a software update with a certain delay, starting from the date on which the strategy bases its decision. We considered different update intervals to determine how the probabilities change if a more responsive approach is employed. We leverage update intervals data from SANS [286] based on a variety of enterprises (Government, Financial Services, Healthcare, and Consulting) and from Kotzias et al. [184] based on 28k enterprises. Table 7.3 shows the update intervals from SANS and maps them to our update strategies.

Immediate strategy: The enterprise updates its software as soon as a new version is available (t_{U_r}) and without delay. If multiple updates are released in the same time interval, the update takes the most recent one. The update is applied even if a vulnerability for the previous version is not present yet. This is the theoretical limit for the enterprise because it is bounded only by the release speed of the vendor. However, this approach is likely impractical because updates require some time to be deployed in an enterprise to not break other functionalities.

Planned strategy: The company updates its software to each new version with a delay from the release date (t_{U_r}). If multiple updates are released in the same time interval, the company takes the most recent one. This delay factors in the time for regression testing

Table 7.2: Update strategies

Each strategy represents an approach for updating the software. The *Immediate* strategy represents the upper bound achievable by an enterprise. The *Planned*, *Reactive*, and *Informed Reactive* are evaluated with different update intervals that represent different levels of responsiveness.

Strategy	Update Interval	Description
Immediate	/	Update to each newest version as soon as it is available without any delay
Planned	1, 3, 7 months	Update to each newest version but wait a delay before the deployment
Reactive	1, 3, 7 months	Update to the first new (non vulnerable) version only after the publication in NVD of a CVE, wait a delay before the deployment
Informed Reactive	1, 3, 7 months	Update to the first new (non vulnerable) version only after the reservation by MITRE of a CVE entry, wait a delay before the deployment

Table 7.3: Update Intervals from SANS [286]

Percentage of enterprises that update weekly, monthly, quarterly, or with other delays. We evaluated the strategies with these update intervals. Enterprises that update weekly are comparable to the *Immediate* strategy. We associated 7 months for the update interval of 'Other' from [184].

Update interval	% Enterprises	Update Strategy Correspondence
Weekly	24.9	*Immediate*
Monthly	57.5	*Planned/Reactive/Informed Reactive* within 1 month delay
Quarterly	7.7	*Planned/Reactive/Informed Reactive* within 3 months delay
Other	10.0	*Planned/Reactive/Informed Reactive* within 7 months delay

and update deployment. The delays are taken from Table 7.3. As in the *Immediate* strategy, the update is *not* triggered by the knowledge of vulnerabilities but only on the availability of a new update.

Reactive strategy The enterprise updates the software only *after* the publication of a vulnerability by NVD (t_{V_p}) with a delay taken from Table 7.3. The new version installed is the first non-vulnerable update available at that time.

Informed Reactive strategy The enterprise updates the software only *after* the *reservation* of a vulnerability by MITRE (t_{V_r}). The new version installed is the first non-vulnerable update available at that time. This strategy describes an enterprise that pays an annual subscription fee to get information about the non-publicly disclosed vulnerabilities from companies that provide 0-day data information (e.g. Exodus Intelligence, Zerodium). The strategy presents an update interval as the *Reactive* and *Planned* strategies.

7.3 Methodology

In this section, we present a methodology to evaluate the effectiveness and cost of update strategies. The definition of probabilistic risk assessment [121] is:

$$Risk = Pr(Compr|Attack) \cdot P(Attack) \cdot Impact \tag{7.1}$$

How to determine $P(Attack)$ is still an unsolved problem in cyber-security [21] while the *Impact* of cyber-attacks has received extensive discussion [27, 92]. We focus on $P(Compr|Attack)$ i.e. the *conditional* probability of being compromised given an attack (or campaign as used in this chapter). We propose a methodology to compute the *conditional* probability of being compromised in Eq. 7.1 by employing historical data about releases available, vulnerabilities, and their exploitation in campaigns. Table 7.4 overviews our methodology.

Table 7.4: Methodology overview

Step 1: Extract APT and software data	
INPUT	APT groups from MITRE Att&ck
OUTPUT	A set of campaigns in the form <*APT_name,CVE,date*>, <*APT_name,attack_vector,date*> and a set of software updates in the form <*sw,update,release_date*>

PROCEDURE Identify campaigns information and software releases:

- Collect resources describing campaigns for each APT based on Threat Actor Encyclopedia [328] and Internet searches using the MITRE APT name and "CVE" as keywords;
- Manually extract from resources the key information: *date* when campaign is observed, *CVE(s)* exploited, *attack_ vector* employed;
- For each CVE, automatically extract software and versions affected from NVD;
- Manually extract from software vendors website the *update* number and *date* of release.

Step 2: Instantiate update strategy

INPUT A set of software updates (<*sw,update,release_ date*>), update strategy (*Immediate*, *Planned*, *Reactive*, *Informed Reactive*), CVEs exploited in APT campaigns

OUTPUT A matrix that describes the application of updates for the software in the period [2008-2020]

PROCEDURE Create a matrix with rows identifying software versions and columns identifying months in [2008-2020] that determines the installed software version at a given time:

- Select the entry corresponding to the first vulnerable version available on 01/2008 (same for all strategies);
- Select another entry corresponding to a new version depending on the update strategy: on the release date of an update for the software (*Immediate*) or with a delay (*Planned*), on the publication (*Reactive*) or reservation date (*Informed Reactive*) of a CVE for the software with a delay;
- Consider availability of non-vulnerable updates at the time of publication of a CVE when computing delay for *Reactive* and *Informed Reactive.*

Step 3: Instantiate APT campaigns events

INPUT Set of events for different campaigns (<*APT_ name,CVE,date*>)

OUTPUT A set of matrices of campaigns. Each matrix describes the software versions targeted by a certain campaign in the period [2008-2020]

PROCEDURE For each campaign, create a matrix with rows identifying software versions and columns identifying months in the [2008-2020] that determine targeted software version at a given time:

- Extract the affected software versions from the CVEs;
- Select the entry of the affected software versions from the *date* of the campaign up to 2020.

Step 4: Generate pessimistic scenarios

INPUT A matrix that describes the application of updates for the software in the period [2008-2020]

OUTPUT	A matrix that describes the application of updates for the software in the period [2008-2020] and maintains both versions during the transition month
PROCEDURE	Update matrix to maintain the previous version in the month in which a new update is installed: • For each month in which the software version is updated to a new version, keep the entry corresponding to the previous version installed for that month only.

Step 5: Compute conditional probability of being compromised

INPUT	A set of matrices of update strategies and a set of matrices of campaigns events
OUTPUT	The conditional probability of being compromised given a set of campaigns are targeting you ($P(Compr|Attack)$) based on the update strategy, # of updates performed
PROCEDURE	Compute successful campaigns targeting installed software in the period [2008-2020]. For each matrix of update strategy: • Select a matrix of campaign events and compute the element-wise product of the matrix with the update strategy matrix to identify the intersection of installed and targeted software versions; • Sum rows of the resulting matrix to determine the months when a campaign is successful, save the campaign if successful; • Continue with another matrix of campaign events until no more campaigns. • Compute conditional probability as the number of successful campaigns divided by the number of matrix campaigns considered; • Compute the number of updates counting non-empty rows in the matrix of update strategy.

Step 6: Compare strategies effectiveness

INPUT	The successful campaigns for the different update strategies
OUTPUT	Confidence Intervals (CI) for update strategies
PROCEDURE	Compare the CI intervals of different update strategies: • Compute the Agresti-Coull 95% CI for the proportion of successful campaigns by update strategy; • Compare intervals, if they overlap update strategies are similar; • Compute pair-wise agreement of successful campaigns for pair of update strategies and Agresti-Coull 95% CI for the resulting proportion of agreement. The interval identifies the expected range of proportion of campaigns that succeed against both update strategies.

Step 1: Extract APT and software data

To collect data we analyzed both unstructured (technical reports, blogs about APT campaigns, and vendor's repositories) and structured (MITRE Att&ck and NVD repositories) public sources.

We do not perform open coding because the information in the reports is deterministic and already based on the MITRE industry standards on CVEs [1] and Initial Access Tactic[2]. The association of CVEs and attack vectors to a certain APT is based on explicit attribution in the consulted resources. Let us consider the following snippet from a Mandiant report referring to APT12[3]:

"*In June 2014, the [Arbor Networks] blog highlighted that the backdoor was utilized in campaigns from March 2011 till May 2014. Following the release of the article, FireEye observed a distinct change in RIPTIDE's protocols and strings. ...FireEye dubbed this new malware family HIGHTIDE.*
On Sunday August 24, 2014 we observed a spear phish email sent to a Taiwanese government ministry. Attached to this email was a malicious Microsoft Word document (MD5: f6fafb7c30b1114befc93f39d0698560) that exploited CVE-2012-0158. It is worth noting..."

we extracted the following information: *date*=08/2014; *CVE*=CVE-2012-0158; *attack vector*=spearphishing attachment.

The entries were then reviewed by a third researcher, not involved in the initial manual analysis, to resolve inconsistencies. Cohen's κ values are 1, 0.976, and 0.863 for the CVE, date, and attack vector respectively (42 disagreements over 652 entries) which show a good agreement among the raters. Total agreement on CVEs is unsurprising as CVEs are unique strings and are reported by copying and pasting the string into the data collection form. Such agreement would not happen between a manual rater and an automatic procedure as we already noted for DAPTSET [189] which is so riddled with false positives to be unusable. Simply, an automatic procedure will collect all CVEs including those that a human rater will see as clearly irrelevant (past campaign, related examples, etc.). Most disagreements are on the attack vector as the mapping of the natural description into the corresponding MITRE Att&ck category is sometimes amenable to interpretation (27 out of the 42 disagreements).

[1]`https://www.cve.org/About/History`
[2]`https://attack.mitre.org/tactics/TA0001/`
[3]`https://www.mandiant.com/resources/darwins-favorite-apt-group-2`

To resolve uncertainty among resources, we made the following *conservative* assumptions:

- if report A says CVE-1 is exploited by an APT campaign and report B says CVE-2 is exploited we mark both CVE-1 and CVE-2 as exploited by the APT in question.
- if report A says an APT campaign started on month X and report B says an APT campaign started on month Y we mark them as two distinct campaigns.

It is not uncommon that different security companies have non-overlapping information about APT campaigns [61,197]. We discuss the implications of this choice in Section 7.6.

Step 2: Instantiate update strategies

We employed a matrix representation to compare update strategies and APT campaigns.

Each strategy is represented as a matrix in which the rows represent the versions of the different products (e.g. *Acrobat Reader* 9.2, *Flash Player 11.0.1.152*) and the column a specific date with a month-base granularity (e.g. 12/2009). A matrix cell is 1 if, on that date, that version of the product is installed and otherwise 0. For a first approximation, we avoid considering the presence of multiple versions installed for the same product[4].

All strategies start from the same version, which is the oldest vulnerable version of a campaign that is available at the beginning of 2008. A strategy updates its version based on the release date of a new version, the publication date, and the reservation date of a CVE for the *Immediate* and *Planned*, *Reactive*, and *Informed Reactive* strategy respectively.

The first two strategies (*Planned* and *Immediate*) update when a new release for the software is available (w/ and w/o an update interval respectively). If multiple software versions are released on the same date, they will update to the newest consistent version.[5]

For the latter two strategies (*Reactive* and *Informed Reactive*) the next version installed, if available, is the *first most recent version* that is not affected by the CVE. We also considered the availability of updates based on the attack scenario in *Step 4*.

Identification of the outcome of attack scenarios

Depending on the availability of an update at the time of the publication of the CVE we have to discern two scenarios:

[4]We assume an update is applied on enterprise's machines at once.

[5]For example, if the current *JRE* version installed is *6u6* and a new update for JRE *5u13* is released after that, the update is ignored because it represents a downgrade of a major update.

- The release of an update is available *before* the publication of the CVE ($t_{U_r} \leq t_{V_p}$). The time when a company may decide to update because it is aware of the vulnerability is correctly computed from the time when a new vulnerability is published. This is the (implicit) assumption in [184, 226].
- The release of an update is available *after* the publication of the CVE ($t_{V_p} < t_{U_r}$). In this case, computing the time when a company may decide to update from the time of publication of the vulnerability will include an interval of time where a vulnerability for the version of a product is known but a non-vulnerable version has not been released yet ($t_{U_r} - t_{V_p}$). The time available to the company must be computed from the time when the release is available.

Step 3: Instantiate APT campaigns events

We created a matrix for each APT campaign with the same rows and columns of the update strategy matrix in *Step 2*. An entry is set to 1 if the version is affected by a CVE exploited by the campaign from the date when the campaign starts until 2020.

Step 4: Generate pessimistic scenarios

The updates and attacks have a month-based granularity because most of the resources *do not* contain information about the exact day in the month in which an update is published or a campaign is performed. We further discuss the limitation of these data in Section 7.6.

To balance possible interleaves between updates and campaigns within the same month, we performed two analyses: a pessimistic *APT-first* scenario and an optimistic *Update-first* scenario, that assume the campaign is executed before or after the update respectively.

To simulate the *APT-first* scenario, we create a new matrix from the update strategy matrix where we maintained the previous version also in the month in which the new update is installed. In other words, the two versions coexist in the month. Thus, we simulate the application of the update later in the month while allowing the APT to exploit the vulnerability. This is done by keeping selected the entry corresponding to the previous version also in the column in which we move to another version for each update strategy matrix generated in the *Step 2*.

Step 5: Compute conditional probability of being compromised

We evaluate at each instant of time, with a *month-base* granularity, the sequence of versions installed on a set of software products for each strategy and compare them with the software exploited by the APTs to determine the potentially successful campaigns.[6] We use the term *potentially successful* because the success of exploiting a vulnerability depends on the characteristics of the execution environment [93]. A campaign is considered successful if it exploits at least one of the software products considered. From the matrix of updates obtained from *Step 2,5* and the matrix of campaigns obtained from *Step 3*, we compute the *conditional* probability of being compromised given one is targeted by the campaigns at a given instant of time t_i. The probability is computed as the number of potentially successful campaigns at time t_i over the total number of campaigns active at that instant of time.

$$P(Compr|C, t = t_i) = \frac{|potentially\ successful\ campaigns_{t_i}|}{|active\ campaigns_{t_i}|} \tag{7.2}$$

where:

- $|potentially\ successful\ campaigns_{t_i}|$ is the number of active campaigns at time t_i that exploit at least one version of a product currently installed at that time.
- $|active\ campaigns_{t_i}|$ is the total number of active campaigns at time t_i.

We computed the $|potentially\ successful\ campaigns_{t_i}|$ by performing an element-wise product of the matrix of update strategy with each matrix describing an APT campaign. The resulting matrix identifies the versions that were installed *and* exploited by the campaign in a given month. With the sum of the rows of the resulting matrix, one obtains a vector of values ≥ 0 for each t_i.[7] If an entry at time t_i is > 0, then the campaign is included in $|potentially\ successful\ campaigns_{t_i}|$.

The overall percentage of potentially successful campaigns over the total number of campaigns in the entire interval of time is computed as:

$$\begin{aligned} P(Compr|C) &= \frac{|potentially\ successful\ campaigns|}{|campaigns|} \\ &= \frac{|\{C|\exists t_i : C \in potentially\ successful\ campaigns_{t_i}\}|}{|campaigns|} \end{aligned} \tag{7.3}$$

In other words, the total number of potentially successful campaigns is obtained from the

[6]For example, in 12/2009 the CVE-2009-4324, affecting *Acrobat Reader* up to version 9.2, is exploited in the wild. If at any time from 12/2009 an update strategy updates to one of these versions, then the campaign is potentially successful.

[7]Values can be > 1 if campaigns can exploit different products.

set of campaigns that could be successful in at least one instant of time t_i. If a campaign can succeed in several instants of time, it is counted only once in the period of interest.

The number of software updates is obtained from the matrix representing the strategy, by counting the number of rows that contain at least one non-zero entry in the columns.

Step 6: Compare strategies effectiveness

For each update strategy, we obtain from Eq. 7.3 a probability of being compromised based on the sample of campaigns considered. To predict the range in which the probability of being compromised for the entire population of campaigns resides we compute a confidence interval (CI). In case of binary outcomes (success, failure), we compute the Agresti-Coull confidence interval [14] that is recommended when the sample size is ≥ 40 [67]. From the CIs of different update strategies, we can then compare their performance. Two strategies are similar if their CIs significantly overlap.

We then determine the percentage of campaigns for which the two strategies behave in the same way by computing the proportion of campaigns that either succeeded or failed against both strategies. By computing the Agresti-Coull interval for the resulting proportion we obtain the range of similarity of the two strategies in terms of the percentage of campaigns that both succeed or failed against two update strategies.

7.4 Data Overview

For more than half of the APT campaigns saturation is reached with at most 5 distinct resources, while for some APTs we collected more than 15 and up to 22 different resources. Only for 11 APTs, we collected a single resource, which typically is a white paper containing detailed information about the APT's activity over an extended period.

We now answer ***RQ5a*** with a quantitative analysis of the attack vectors employed, the vulnerabilities exploited, and the software products targeted to understand the APT ecosystem.

7.4.1 Attack Vectors

We analyzed the attack vectors exploited in the different campaigns with the presence and absence of software vulnerability. Table 7.5 shows the different attack vectors and the number of campaigns in which are observed. We underline that a campaign can employ

Table 7.5: Attack vector campaigns and software vulns

	# of Campaigns	
Attack vector	**w/o vuln**	**w/ at least one vuln**
Spear phishing	130*	122*
Drive-by Compromise	15*	34*
Supply Chain Compromise	5*	0
Valid Accounts	3*	1
External Remote Services	3	0
Exploit Public-Facing Appl.	3*	7
Replic. via Remov. Media	0	1
Undetermined	38*	9*
Total	197 (190 unique)	174 (162 unique)

* Contains duplicates due to multiple attack vectors.

one or more attack vectors.[8]

We can observe that spear phishing is the main attack vector [337], present in 130 campaigns that do not exploit any vulnerability and 122 campaigns that exploit at least one vulnerability. Interestingly drive-by compromise is not only employed when a vulnerability is present but also used to facilitate campaigns that employ social engineering to trigger users to download malware.

We have 47 campaigns for which we do not know the attack vector. For 9 of them, the report identified the vulnerability exploited but not the attack vector.[9] If this information is not present in the report, we avoided making assumptions. For the remaining campaigns, the information about the attack vector was vague or missing.[10]

7.4.2 Popular Products and CVEs

We observed 118 unique vulnerabilities exploited by the APTs in at least one campaign between 2008 and 2020. Some CVEs are exploited in several campaigns by different APTs.

Table 7.6 shows the ten most targeted client-side applications and the ten most targeted server/O.S. products based on the exploited CVEs. A campaign is counted over different products if the CVE employed is applicable to different software products. For example, CVE-2012-0158 affects Office, SQL server, Visual Fox Pro, and Commerce Server.[11]

[8]For example, it is not uncommon to have campaigns that exploit both spearphishing and drive-by compromise.

[9]For example, some vulnerabilities (e.g. CVE-2012-0158) can be exploited via spearphishing techniques and drive-by compromise.

[10]For example, the Sony hack campaign in 2014 [236].

[11]We do not have information about the exact software targeted. For example, they could all have

Table 7.6: Top 10 client-side and Top 10 server-side/O.S. products exploited

The products are obtained from the CVEs exploited in a campaign. If a CVE affects multiple products, all the software are considered. Products are distinguished in *client-side*, *server-side* applications, and *O.S.*

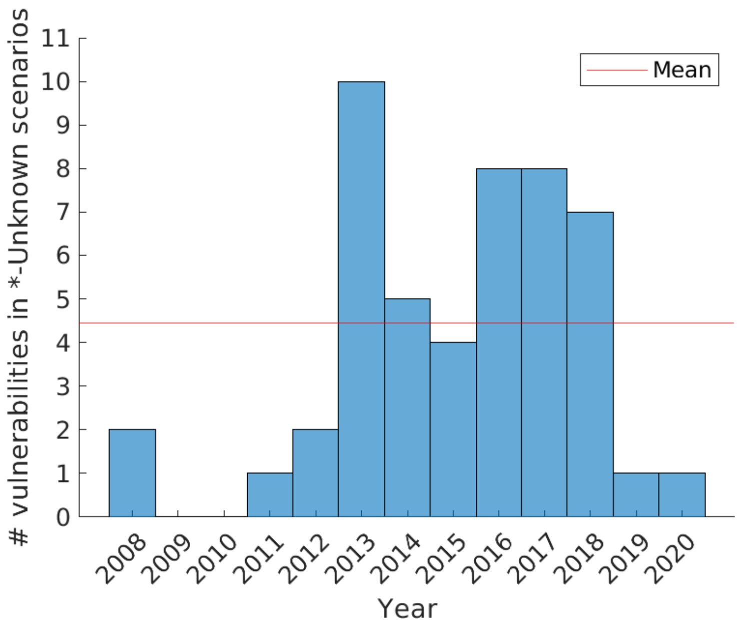
Vendor	Product	Software	# Campaigns (%)
Microsoft	**Office**	**Client**	**68 (41.9%)**
Microsoft	Windows 2008 Server	O.S.	49 (30.2%)
Microsoft	Windows 7	O.S.	43 (26.5%)
Microsoft	Windows Vista	O.S.	41 (25.3%)
Microsoft	Windows 2012 Server	O.S.	39 (24.0%)
Adobe	**Flash Player (EOL)**	**Client**	**35 (21.6%)**
Microsoft	Windows 8.1	O.S.	29 (17.9%)
Microsoft	Commerce Server	Server	19 (11.7%)
Microsoft	SQL server	Server	19 (11.7%)
Microsoft	Visual Basic	Client	19 (11.7%)
Microsoft	Visual FoxPro	Client	19 (11.7%)
Microsoft	BizTalk Server	Server	18 (11.1%)
Microsoft	Windows 10	O.S.	18 (11.1%)
Microsoft	Windows 8	O.S.	14 (8.6%)
Microsoft	IE (EOL)	Client	13 (8.0%)
Adobe	**Acrobat Reader**	**Client**	**11 (6.8%)**
Microsoft	.NET framework	Client	5 (3.1%)
Adobe	**Air**	**Client**	**5 (3.1%)**
Oracle	**JRE**	**Client**	**4 (2.5%)**
Oracle	JDK	Client	4 (2.5%)

Office is by far the major target of campaigns followed by *Windows O.S.* and *Flash Player*. This is coherent with the attack vectors previously observed as they are commonly exploited via spearphishing with malicious attachments.

APTs tend to "share" vulnerabilities during their campaigns. Only 8 APTs (`Stealth Falcon`, `APT17`, `Equation`, `Dragonfly`, `Elderwood`, `FIN8`, `DarkHydrus`, and `Rancor`) exploit CVEs that are not used by anyone else.[12] We are aware of vulnerabilities (e.g CVE-2017-0144) that are associated with `Equation` and used by other APTs, but we did not find enough information about the date when the vulnerabilities were employed. Roughly 35% of the APTs exploit CVEs observed in campaigns of other groups. 17 APTs share 4 or more vulnerabilities, while many APTs sharing a single vulnerability have only exploited that vulnerability during their campaigns (14 out of 20).

exploited Office.

[12]Only three APTs have exploited more than one vulnerability during all their campaigns.



The number of unique vulnerabilities employed in *Unknown-Unknown (UU)* and *Known-Unknown (KU)* attack scenarios grows significantly in recent years, compared to the first years of observation. On average around 5 distinct vulnerabilities per year are exploited by APTs.

Figure 7.2: Number of distinct vulnerabilities exploited over the years by different attack scenarios.

7.4.3 Evolution in Exploiting Vulnerabilities

Figure 7.2 shows the evolution of the number of unique vulnerabilities exploited in the **-unknown* attack scenarios[13] in our database. It represents a lower bound of the vulnerability exploited in the wild. Project Zero [260] collects information about 0-days in the wild by including also unattributed attacks. The mean number of distinct vulnerabilities exploited per year is roughly 5. We can observe how the numbers grew significantly in recent years. However, it can be influenced by the limited number of reports for campaigns in the early period (2008-2011), where it was less likely to report information about cyber-attacks. The drop for 2019/2020 is due to the natural delay of publicly reporting campaigns caused by the proximity of the period of data collection with the date of the campaigns themself. Thus, we expect the values to be higher if recomputed in the future.

Looking at the occurrence of a CVE in an APT campaign, the majority of the APTs prefer to exploit CVE already published, with few APTs as exceptions.[14]

[13]Either already reserved *(KU)* or not reserved *(UU)*.
[14]`Stealth Falcon, PLATINUM, APT17`

7.4.4 Software for Analysis of Update Strategies

As discussed in Section 7.3, the collection of update releases from vendors' websites is a manual procedure. Here, for a first approximations, we focus on collecting updates for a subset of all software targeted by APTs.

Table 7.6 shows the most targeted products by vendor. We decided to cover the most exploited client-side product for each vendor because (1) from Table 7.5 most of the campaigns exploit attack vectors directed to client-side software and (2) it is not uncommon to have products from these vendors in an enterprise computer. For Adobe, the *Flash Player* product is end of life (EOL) thus we decided to include the other two software products *Reader* and *Air*. Even if *Flash Player* is EOL in 2020 we still think it is interesting to see how different update strategies would affect the security of enterprises because it has been frequently exploited in the last years. Also, vendors' EOL of products, unfortunately, does not coincide with the disappearance from the field and end of exploitation as we are observing with Internet Explorer [176].

For a first approximation, we limited the analysis to the *Office* 2016 release only, as different releases (*Office* 2013, *Office* 365) can be seen as different products as they require buying a different license each. We considered the Knowledge Base (KB) updates from the Microsoft Update Catalog as the versions of the software. We assumed that KB updates for *Office* are cumulative, i.e. the package contains all previously released fixes.

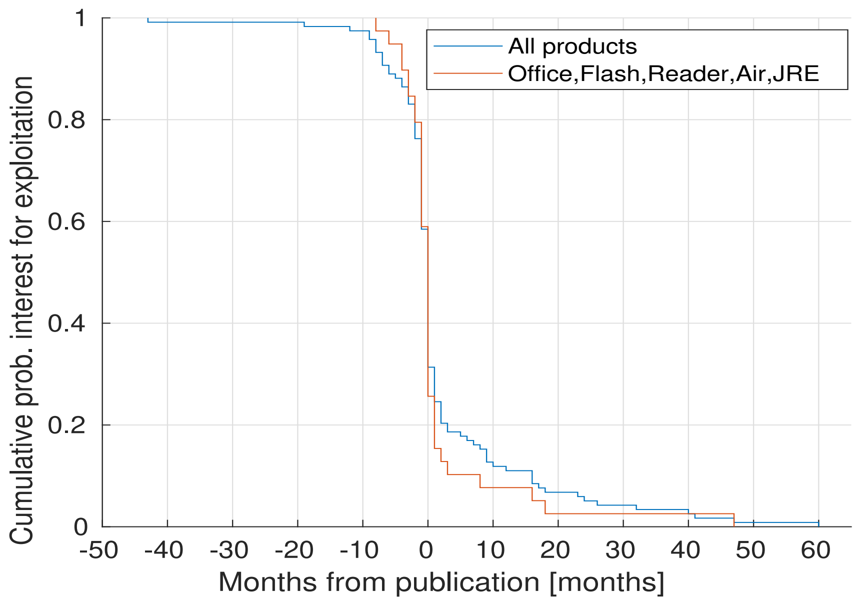
In summary, we collected releases of updates for 5 different software products from 3 different vendors: Office, Flash Player, Acrobat Reader, Air, and JRE. We considered only releases for Microsoft Windows O.S. as it covers at least half of the enterprise computers [315]. With this set of software products, we cover 44% of the campaigns (that exploit software vulnerabilities), 62% of the APT groups, and 33% of the CVEs.

7.5 Evaluation of Software Updates Mitigations

We now present an analysis of the speed of exploitation of individual vulnerabilities and the prevalence of **-Unknown* and *Known-Known* attacks in APT campaigns. We then quantitatively evaluate the effectiveness and cost of the different update strategies against the APT campaigns.

7.5.1 Survival Analysis

We performed a preliminary survival analysis on the vulnerabilities to compute the interval in months that passed from the publication of the CVE and the *first* campaign that



The survival is based on the *first* time the CVE is exploited in a campaign. More than half of the vulnerabilities are exploited for the *first time* within one month from the publication. However, there is high survivability of a small set of CVEs (roughly 10%) that are exploited after more than 1 year from the publication. If we consider only *Office*, *Flash Player*, *Reader*, *Air*, and *JRE* the behavior is similar.

Figure 7.3: Proportion of survival of CVE from publication (NVD) for all products and a subset (*Office*, *Flash Player*, *Reader*, *Air*, and *JRE*).

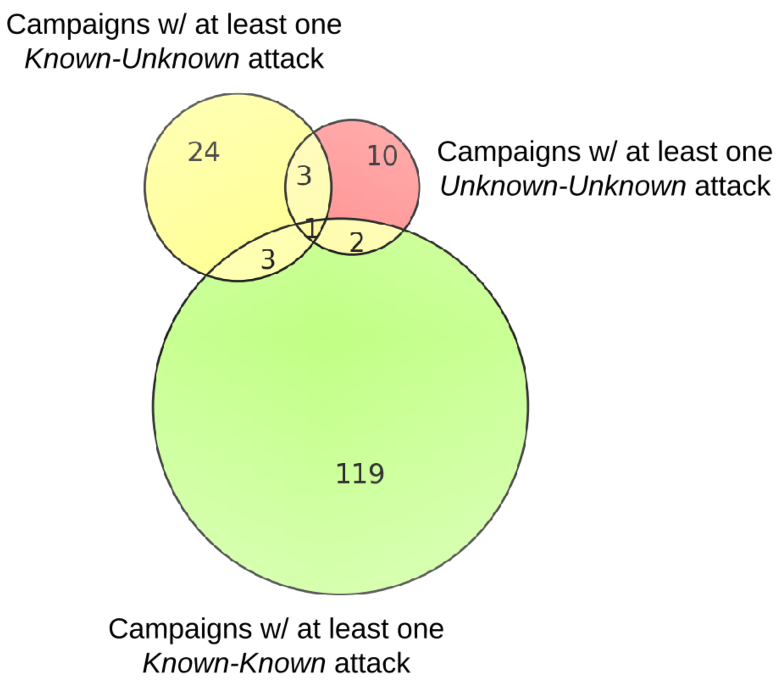
exploited the CVE (exploit age). Figure 7.3 shows the Kaplan-Meier plot for all the products in our database and for the set of products discussed in Section 7.4 (*Office*, *Flash Player*, *Reader*, *Air*, and *JRE*). We can see that roughly 40% of the vulnerabilities are exploited for *the first time* before the publication. This is coherent with what was observed by Chen et al. [82], where 49% of the CVEs are exploited before the NVD score is published. Furthermore, roughly 27% of the vulnerabilities are exploited *the first time*[15] within a month from the publication from NVD showing that APTs are fast to exploit new CVE [122]. Another interesting fact is that a significant number of vulnerabilities are exploited a few months before the NVD publication. This phenomenon can be partially explained because the observation of attacks in the wild brings software vendors to know about the vulnerability and thus the publication of a CVE. It is important to underline that this value does not mean that ≈40% of the campaigns are unpreventable because 1) **-Unknown* attacks can exploit several vulnerabilities[16] and 2) many of these CVEs are exploited multiple times from different APTs after months from their first exploitation.

[15] Among all the APTs.

[16] A famous example is Stuxnet.

If we only consider the vulnerabilities exploited the first time in *KK* attacks (as in [50]), we observe that roughly 47% of them are exploited within 30 days from their publications[17]. In contrast with previous results [227], we observed a long tail for part of the vulnerabilities, one out of 10 CVEs is exploited after one year from its publication, and 1 out of 20 after more than two years.

7.5.2 Classification of APT Campaigns



The majority of campaigns exploited at least one vulnerability in a *KK* attack (after publication by NVD and after reservation by MITRE). Only a few launched *UU* attacks (both before reservation by MITRE and before publication by NVD).

Figure 7.4: Classification of APT Campaigns.

Each APT campaign exploiting at least one vulnerability fits into one of these (possibly overlapping) groups:

- Campaigns with at least one *Known-Known (KK)* attack . In other words, the campaign exploited at least one vulnerability (either preventable or unpreventable) that was already present in the NVD database.

[17]Bilge et al. [50] observed a similar value of roughly 42%

- Campaigns with at least one *Known-Unknown (KU)* attack. In other words, the campaign exploited at least one vulnerability (either preventable or unpreventable) that was not present in the NVD database but an entry was already reserved by MITRE.[18]
- Campaigns with at least one *Unknown-Unknown (UU)* attack. In other words, the campaign exploited at least one vulnerability (either preventable or unpreventable) that was not even reserved by MITRE.

Out of 352 campaigns, less than half of them employ at least one vulnerability (Table 7.5). Figure 7.4 shows the resulting Venn diagram for the 162 campaigns of interest. 119 out of 162 campaigns employed only vulnerabilities in *Known-Known* attacks.

APTs heavily exploit known CVE to compromise their target. The prioritization of updates is thus a key factor that can significantly reduce the impact of APTs campaigns.

7.5.3 Effectiveness and Cost of Updates Strategies

We now answer ***RQ5b*** by applying our methodology (Section 7.3) to compute the overall probability of being compromised (Eq. 7.3) in the interval of time [Jan 2008-Jan 2020] with the updates strategies and update interval presented in Section 7.2 for the software discussed in Section 7.4.4. Table 7.7 summarizes the results in terms of the number of updates required, the conditional probability, and the odds ratio for the *optimistic* (*Update first*) and *pessimistic* (*APT first*) scenarios.

Updating the software as soon as a new release is available (*Immediate* strategy) provides the optimal lower-bound probability of being compromised. Even in this case, roughly 1 out of 4 campaigns can compromise the target. Although an immediate update can be applied in some critical situations, if we consider a more realistic approach in which the software is updated with some delay in the month (*Immediate* with *APT first*), the odds of being compromised increases by a factor of 5. What stands out is that the *Planned* strategy obtains the same probability of being compromised as a strategy that waits for the presence of public vulnerabilities (*Reactive* strategy). However, waiting to update when a CVE is published presents ≈33% fewer updates. Thus, if an enterprise cannot keep up with the updates and need to wait before deploying them, can consider being simply reactive. For the *Planned* strategy the number of updates decreases with bigger intervals because the updates are shifted outside of the period of observation. If a longer update interval is used, the probability of being compromised increases by a factor of ≈10 and ≈21 for 3 months and 7 months update intervals respectively. Comparing

[18]Thus, a small number of people known already some information about the vulnerability. E.g. vulnerability researchers.

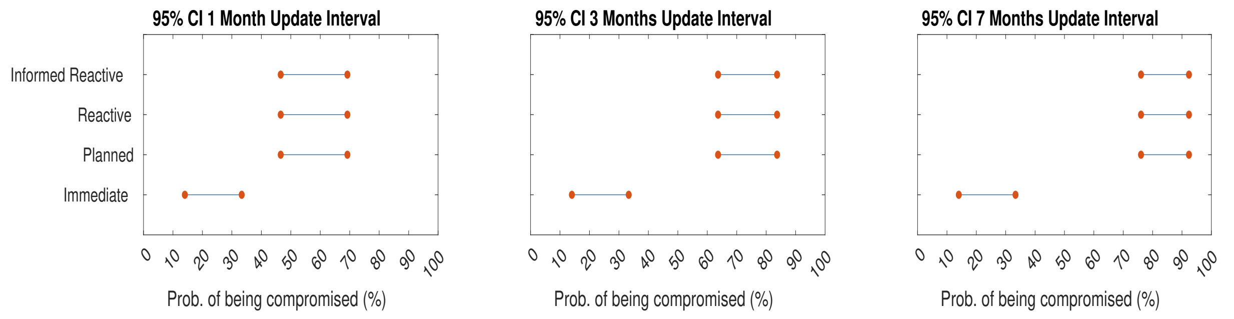
Table 7.7: Optimistic (*Update first*) and pessimistic (*APT first*) overall *conditional* probability of being compromised for different update strategies and update interval with the associated # of updates for the period [01/2008-01/2020]

Update Interval	Strategy	#Updates	Prob. (*Update first*	Odds — *APT first*)
/	Immediate	364	22.2-58.3%	1x-4.9x
1 Month	Planned	361	58.3-66.7%	4.9x-7.0x
	Reactive	247	58.3-66.7%	4.9x-7.0x
	Informed Reactive	257	58.3-66.7%	4.9x-7.0x
3 Months	Planned	354	75.0-77.8%	10.5x-12.3x
	Reactive	247	75.0-77.8%	10.5x-12.3x
	Informed Reactive	257	75.0-77.8%	10.5x-12.3x
7 Months	Planned	341	86.1-87.5%	21.7x-24.5x
	Reactive	244	86.1-87.5%	21.7x-24.5x
	Informed Reactive	254	86.1-87.5%	21.7x-24.5x

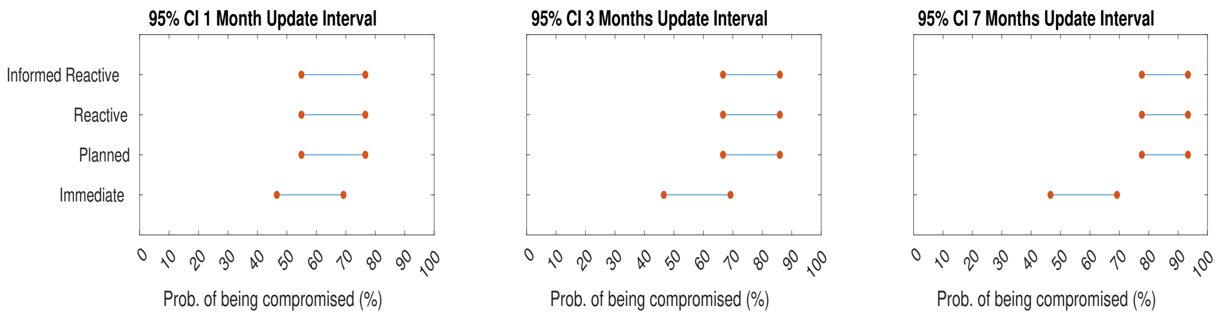
the *Reactive* and *Informed Reactive* strategies, there is no advantage in knowing about not publicly known vulnerabilities if the enterprise waits to update. We observed that to perform better than the *Reactive*, the *Informed Reactive* must apply immediately the update otherwise the advantage is lost even with a single month of delay.

We reported in Figure 7.5 the Agresti-Coull Interval for each update strategy for the different update intervals. The *Planned*, *Reactive*, and *Informed Reactive* strategies are identical as we see a significant overlap of the CI among these three strategies. In the case of an optimistic *Update-first* scenario, we observe that there is a clear difference between the *Immediate* and the *Planned* strategies, while this advantage is lost in the case of the pessimistic *APT-first* scenario. To evaluate the similarity we computed, for each pair of strategies, the proportion of campaigns that either succeeded or failed against both strategies. We estimate the Agresti-Coull CI for the resulting proportions. The results show that the *Planned* and *Reactive* behave in the same way for at least 94% up to 100% of the campaigns

Either you update always and *immediately to the new versions or just updating lately has the same risk profile but costs you a lot more than updating rarely [213].*



(a) *Update-first* (optimist) scenario. There is a difference between the *Immediate* and the other strategies. However, *Planned*, *Reactive*, and *Informed Reactive* behave similarly thus updating to each new version with some delay or relying on reserved CVE does not worth it.



(b) *APT-first* (pessimist) scenario. The *Immediate* and *Planned* present a similar behavior for the 1 month update interval but differ with bigger intervals. In the pessimistic scenario the *Planned*, *Reactive*, and *Informed Reactive* behave similarly thus updating to each new version with some delay or relying on reserved CVE does not worth it.

Figure 7.5: Agresti-Coull Interval (CI) for the update strategies with different update intervals

7.6 Limitations

The dataset obtained is based on publicly available reports. While this is just a small part of existing campaigns, this work is the first that tries to aggregate a manually validated dataset of APTs campaigns, CVE, and vulnerable products and it is a first step in the direction of an open and extensive dataset on APT campaigns.

The process to obtain information about campaigns was semi- automated but required manual effort to analyze and extract the key information about campaign dates, CVE, and attribution. We assume that this type of information reported by reputable security companies is not deliberately wrong, and our methodology strives to find multiple sources reporting the same campaign to control for possible errors. Since keyword-based automated searches (e.g. [189]) present limitations in the number of false associations that they generate, we decided that a manual approach would provide a more precise description of the APT ecosystem. Although the manual extraction of information from reports does not present difficulties, it can include erroneous matching of APT campaigns. To limit that, the manual analysis was performed by two researchers independently and inconsistencies were resolved by a third researcher.

We decided to ignore reports about campaigns where not enough information about the start and attribution was available. Thus, it is possible that certain vulnerabilities discussed in the reports are not included in the dataset.

We applied a conservative approach in extracting information from different reports reporting mutually disjoint CVEs exploited on the same date. Thus, potentially assuming fewer campaigns with a higher number of CVEs each. The probability of being compromised must be seen as an upper bound of what APT can achieve. However, the odds ratio between update strategies remains the same.

We relied on the NVD data as the industry standard but it is known to contain errors in the list of product names, CVE publication date [31], and vulnerable versions [105,230]. We leave for future work the application of these approaches to find inconsistencies. We relied on the data of observation of the campaigns as reported in the reports we consulted. This information could be wrong and detect only a more recent campaign. We tried, when possible, to find multiple resources about the campaign. The collection of release dates for the software discussed in Section 7.4.4 is collected manually given that vendors' repositories are not intended for past versions. Thus, the releases collected and employed in the evaluation might have errors and this could affect the *Immediate* and *Planned* strategies.

We used a month-based date granularity for the publication of the CVE, the release of new versions, and the date of the campaigns because the exact day in the month in which the campaign started is not known. This decision has a potential impact on the results. If a campaign for a CVE published on 29/01/2017 started on 01/02/2017 then in our case the exploit age is one month, even if the CVE is exploited a few days after the publication. However, the results we observed (e.g. exploit age of vulnerabilities) are coherent with previous observations of attacks in the wild [50], thus we think that the number of these cases is minimal and do not affect the results.

The same considerations apply to the results in Table 7.7: if a release is performed on 15/02/2019 and a campaign exploiting the software is executed on 03/02/2019, the month granularity would traduce both actions as performed on 02/2019. We thus considered two complementary scenarios: an *optimistic* scenario (*Update first*) and a *pessimistic* scenario (*APT first*). In the *Update first* the example above will traduce in the defender be able to update before the execution of the campaign. While in the *APT first* we assumed the

Table 7.8: State of the Art on APTs - Main Research Topics

Research Category	State of the Art	This work
APTs data sources	[195]	✓
Metrics for TI sources	[197], [61]	
Attackers characteristics	[83]*, [338]*, [337]	✓
Detection of attacks	[138], [369], [48], [79], [128], [210], [209], [66], [254], [132], [283], [297], [220], [296], [153], [150], [25]	
Game Theory	[158], [363], [283]	
Exploitation likelihood	[210], [363]	✓
Analysis of update releases	-	✓

* Performed high level analysis on few APTs campaigns.

opposite.

Finally, those companies that have an update interval that is less than a month will present a probability of being compromised that stays between the *Immediate Update-first* and the *Immediate APT-first* scenarios.

We assumed that a campaign will be carried on from the date when the campaign started up to the end of the observation (i.e. 2020). This causes an inflation of the number of campaigns that are active at a given instant of time in Eq. 7.2. However, we follow a conservative approach and assumed that if an APT has access to a vulnerability it will always be able to employ it given that one is under attack. We discuss extensions in the Section 7.8.

7.7 Related Work

Table 7.8 shows the research categories addressed by the state-of-the-art on APTs. The majority of the research activity focused on the detection of APTs campaigns while few papers tried to characterize their behavior, estimate the risk, and evaluate update strategies from real data.

APT and Metrics for Threat Intelligence sources Lemay et al. [195] presented a description of different resources about the activities of more than 40 APTs. Li et al. [197] utilized a set of metrics (Volume, Differential contribution, Exclusive contribution, Latency, Accuracy, and Coverage) to compare different public and private Threat Intelligence (TI) data feeds. They observed that in the majority of the data feeds there is no overlapping of Indicator of Compromise (IOC) and a high number of false positives. Similarly, Bouwman et al. [61] analyzed two paid TI and observed very few overlaps in the indicators for 22 APTs. The distinction is confirmed with a comparison with open TI data. Furthermore, they observed that TI data is employed in the decision process of companies, but there is a lack of metrics to determine the quality of these data. Several works [189, 199, 289] proposed a (semi-)automated approach based on report parsing to generate a database of IOC. However, merely relying on the results of the automated approach generates many false positives. For example in [189], we observed that CVEs are wrongly associated to the *admin@338* group in a report about the Poison Ivy malware, where several campaigns from different actors are described. We provide a manually curated database from which we can quantitatively evaluate the impact and cost of software update strategies.

Several studies evaluated the overlap among threat data feeds, showing poor accuracy. Mechanisms to semi-automatically extract information from reports are prone to false-positive.

Analysis of attackers characteristics Ussath et al. [338] analyzed 22 reports about APT campaigns and mapped them into the three phases of an attack (initial compromise, lateral movements, and Command&Control). They found that most of them employ social engineering techniques and living-off-the-land techniques. Furthermore, they noted that 0-day vulnerabilities are not exploited frequently by APTs. Chen et al. [83] studied 4 APT campaigns to analyze the phases of these attacks and determine possible countermeasures. Urban et al. [337] analyzed 93 APT reports (66 different APTs) and determined that spearphishing is the main attack vector. They then collected OSINT data like domain names and social media information of 30 companies to determine how much information is available to the adversaries. Additional works on APTs analysis focused on describing the phases of the attacks and possible countermeasures [275], the analysis of the malware employed in a few well-known campaigns [347], or the prevalence of living-off-the-land techniques in certain samples [41].

To the best of our knowledge, we are the first to analyze more than 350 campaigns exploiting 118 different CVEs from the inspection of more than 500 reports. This massive analysis makes it possible to draw significant conclusions on the efficacy of update strategies.

Although several works provided insights into the APT ecosystem, the analysis focused on a handful of campaigns that make it hard to draw significant conclusions on the characteristics of APTs.

Detection of attacks An orthogonal problem is detecting live APTs attacks once they get into the network. Different research proposed to employ machine learning [25,79,132], information flow tracking [66, 150, 153, 220], statistical correlation [292], and big data analysis [48,128,209,210].

Shu et al. [297] employed a temporal computational graph to perform threat hunting activities via graph patterns matching and analyzed a case study on a DARPA threat detection competition. Pei et al. [254] developed a framework to generate a multi-dimensional weighted graph based on log entries and identify attacks by the presence of dense connections among logs using unsupervised learning techniques. They evaluated it over 15 APTs campaigns.

The state-of-the-art focused mainly on the detection and response against APT attacks, while there is a lack of investigation on the orthogonal problem of prevention.

Game Theory Hu et al. [158] presented a two-layer attack/defense game to study APT attackers that make use of insiders and compute the best strategies for the attacker and defender. Sahabandu et al. [283] formulated a game-theoretic model to determine the optimal defender strategy in terms of tracking information flow (Dynamic Information Flow Tracking). Yang et al. [363] proposed a Nash game to model the response strategy and minimize the loss of an enterprise against lateral movements in the network of APTs in the network. We instead focus on the initial access phase of APTs campaigns and we evaluated the efficacy of software update strategies based on real data of attacks.

Game theory is extensively applied to find an optimal strategy against targeted attacks. However, these studies employ artificial data and networks.

Analysis of exploitation likelihood Many works employed ML and statistical methods to analyze vulnerabilities and predict the exploitation likelihood by joining data from resources like NVD, Exploit DB [62], historical data on attacks [20, 167], Dark Web forums [24], and Twitter [82, 281]. An extensive discussion of the academic literature on empirical cyber risk can be found in [360].

Other works investigated actual compromises using logs. Marchetti et al. proposed a framework to prioritize the internal clients of an organization that are most likely to be compromised by an APT using internal (network logs and flow records) and external (social media) data [210] and to detect data exfiltration using a set of host-based features and flow records analysis [209]. Similarly, Bilge et al. [51] and Liu et al. [204] employed supervised learning algorithms to determine machines at risk of infection from *internal* logs on binary file appearance, *external* data of misconfigured services (e.g. DNS or BGP), and malicious behaviors (e.g. spam or phishing).

We extend this line of research by proposing a methodology to evaluate the probability of being compromised by APTs and the cost associated with the update strategy.

Analysis of historical data about vulnerability and attacks as well as live information provided by logs and social platforms allows one to evaluate the exploitation likelihood.

Analysis of update releases From the client-side, Nappa et al. [226] proposed a systematic analysis of the update process and update delay on client applications, and performed a survival analysis of vulnerabilities based on data from Symantec. Similarly, Kotzias et al. [184] presented a longitudinal study of the update behavior for 12 client software and 112 server applications based on data from 28k enterprises. Sarabi et al. [288] employed Symantec dataset to model users' update delay as a geometric distribution and study 4 different products (Chrome, Firefox, Thunderbird, and Flash Player).

From the vendor-side, Arora et al. [36] analyzed vendors' patch behavior as a function of several factors like disclosure time, characteristics of the vendor, and severity of the vulnerability. Clark et al. [86] studied if agile methods produce a higher number of vulnerabilities in Firefox. They observed that rapid software releases do not increase the number of vulnerabilities in the code. Ozment and Schechter [247] analyzed the impact of legacy code on the number of vulnerabilities observed OpenBSD versions.

Similar to our work, Beres et al. [46] employed a discrete-event simulator to determine the exposure reduction produced by different security policies by varying update speed and mitigations. However, they modeled events like exploits and updates availability assuming fixed exponential functions looking at global trends observed by a security firm.

We present a quantitative evaluation of the effectiveness and cost of realistic update strategies by using historical data about APT campaigns.

Several works analyzed the update behavior of clients and vendors. However, there are only theoretical works on the efficacy of updates against targeted attacks for an enterprise.

7.8 Conclusions

In this work, we proposed *a methodology to quantitatively investigate the effectiveness and cost of software updates strategies against APT Campaign.* We applied the methodology to build a database of APT campaigns and presented an analysis of the attack vectors, vulnerabilities, and software exploited by 86 different APTs in more than 350 campaigns over 12 years. The database is publicly available on Zenodo [330].

In contrast to expectations, we showed that preventive mechanisms like updates can influence the probability of being compromised by APT. However, software updates based on wrong measures of risk can be counterproductive. Our analysis shows that a purely *Reactive* update strategy (wait until a vulnerability gets out) presents results very similar to a *Planned* strategy (always update to the newest version), but with ≈33% fewer updates. Furthermore, the *Informed Reactive* strategy, where updates are applied based on reserved information about not publicly known vulnerabilities (e.g. by paying for information on 0-days), does not produce significant advantages compared to the *Reactive* strategy and it is useless if the enterprise waits before applying the update.

In summary, for the broadly used products we analyzed, if you cannot keep updating always and immediately (e.g. because you must do regression testing before deploying an update), then being purely reactive on the publicly known vulnerable releases has the same risk profile than updating with a delay but costs significantly less.

Future work can extend the analysis to a more complete set of software products and evaluate a subset of campaigns by targeted enterprises, attacker preferences, or network exposure based on IDS alerts [21]. To achieve that, one would require to have company-specific information to move from a conditional probability to an absolute probability.

We also plan to extend the evaluation by considering campaigns as active only for a limited period. Further data about the lifetime of campaigns in the wild is required.

Chapter 8

Conclusions and Future Work

This dissertation has shown how to develop models and methodologies to quantitatively evaluate the effectiveness of security mitigation strategies using the security data available in different scenarios. Our results create an opportunity for individuals and companies to increase their security by selecting the most appropriate security mitigation strategy for their needs based on a systematic and quantitative evaluation of effectiveness and costs. For what concerns the privacy of users on the Web, we modeled tracking practices and evaluated the effectiveness of different filter lists of privacy extensions. Our approach allows a third-party independent verification of tracking claims and filter lists performance for individual users. In the complementary context of the security of Web users, we modeled several attacks and mitigations at different layers of the Internet. We evaluated the Internet security posture and observed that currently deployed security mitigations are ineffective. An increase in security can be achieved with relatively cheap mitigations at the application level like HSTS, HTTPS, and SRI. Future works can investigate the difference in security and privacy perceived by varying the point of collection of the security data and by extending the attack scenarios and available mitigations.

Our model of task complexity allowed us to employ security data about incidents to measure the accuracy of SOC analyst investigation. The model allows one to determine the impact on the performance of changes in the technical skills of the analyst as well as in the tools employed to investigate the intrusion. Future works can further expand the quantitative analysis of these socio-technical components' performance, for example on specific phases of an investigation like malware analysis or in a dynamic context like an incident response case.

To deal with unstructured data describing criminal activities we developed methodologies to represent and analyze their complex relationships. In the context of underground forums, we presented a methodology to generate samples for training ML classifiers based on the network characteristics obtained by the interactions of the members in the forum. We then showed how to evaluate ML performance on the entire forum to identify changes in performance from similar samples. In the context of APTs, we showed how current best practices on software updates are detached from reality as they lack a quantitative view of the APT landscape. Organizations cannot keep the pace of the updates and, while criticized for that, their behavior is rational as the risk of being compromised does not change. This work raised the interest of Trend Micro. We are planning to extend this work with an analysis of the effectiveness of the update strategies discerning targeted and untargeted attacks and by varying the victim profile.

This thesis is an attempt to incentivize the application of effective, in contrast to visible, defense mechanisms. Independently from the security data characteristics, be them fully structured and visible like blacklist feeds and software vulnerability repositories or unstructured and not visible like underground forum discussions and threat intelligence reports, we showed that with appropriate methodologies that rely on data models construction of the key processes and relationships between attackers and defenders from security data, we can measure the real effectiveness of security mitigation strategies. By making explicit the causal relations between threats characteristics, the targeted network, and the security mitigations available, we obtain results that are put in context and provide a transparent explanation of why certain mitigations succeed and others do not, why some are worth it, and if so under which conditions.

Bibliography

[1] Operators of online "virtual worlds" to pay $3 million to settle ftc charges that they illegally collected and disclosed children's personal information, 2011. `https://www.ftc.gov/news-events/press-releases/2011/05/operators-online-virtual-worlds-pay-3-million-settle-ftc-charges`. Accessed: 2022-11-01.

[2] Federal Trade Commission: Children's Online Privacy Protection Rule ("COPPA"), 2013. `https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule`. Accessed: 2022-01-11.

[3] European Parliament Regulation 2016/679: General Data Protection Regulation, 2016. `https://eur-lex.europa.eu/eli/reg/2016/679/oj`. Accessed: 2022-11-01.

[4] Google and youtube will pay record $170 million for alleged violations of children's privacy law, 2019. `https://www.ftc.gov/news-events/press-releases/2019/09/google-youtube-will-pay-record-170-million-alleged-violations`. Accessed: 2022-11-01.

[5] Mitre att&ck tactics, 2022. `https://attack.mitre.org/tactics/enterprise/`. Accessed: 2022-04-20.

[6] Natural language toolkit, 2022. `https://www.nltk.org/`. Accessed: 2022-11-07.

[7] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juárez, Arvind Narayanan, and Claudia Díaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proc. of ACM CCS-14*, 2014.

[8] Gunes Acar, Marc Juárez, Nick Nikiforakis, Claudia Díaz, Seda F. Gürses, Frank Piessens, and Bart Preneel. Fpdetective: dusting the web for fingerprinters. In *Proc. of ACM CCS-13*, 2013.

[9] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. You get where you're looking for: The impact of information sources on code security. In *Proc. of SSP-16*, 2016.

[10] Yasemin Acar, Christian Stransky, Dominik Wermke, Michelle L. Mazurek, and Sascha Fahl. Security developer studies with github users: Exploring a convenience sample. In *Proc. of SOUPS-17*, 2017.

[11] Alessandro Acquisti, Curtis Taylor, and Liad Wagman. The economics of privacy. *Journal of Economic Literature*, 54, 2016.

[12] Keumars Afifi-Sabet. Google is shifting youtube infrastructure to google cloud, 2021. `https://www.itpro.co.uk/cloud/cloud-computing/359785/google-is-shifting-youtube-infrastructure-to-google-cloud`. Accessed: 2022-11-01.

[13] A. Agresti, C.A. Franklin, and B. Klingenberg. *Statistics: The Art and Science of Learning from Data.* 2016.

[14] Alan Agresti and Brent A Coull. Approximate is better than "exact" for interval estimation of binomial proportions. *The American Statistician*, 52, 1998.

[15] Ugur Akyazi, Michel van Eeten, and Carlos H Gañán. Measuring cybercrime as a service (caas) offerings in a cybercrime forum. In *Proc. of WEIS-21*, 2021.

[16] Bushra A Alahmadi, Louise Axon, and Ivan Martinovic. 99% false positives: A qualitative study of soc analysts' perspectives on security alarms. In *Proc. of USENIX-22*, 2022.

[17] Luca Allodi. Economic factors of vulnerability trade and exploitation. In *Proc. of ACM CCS-17*, 2017.

[18] Luca Allodi, Tzouliano Chotza, Ekaterina Panina, and Nicola Zannone. The need for new antiphishing measures against spear-phishing attacks. *IEEE Secur. Priv.*, 18, 2020.

[19] Luca Allodi, Marco Cremonini, Fabio Massacci, and Woohyun Shim. Measuring the accuracy of software vulnerability assessments: experiments with students and professionals. *Empir. Softw. Eng.*, 25, 2020.

[20] Luca Allodi and Fabio Massacci. Comparing vulnerability severity and exploits using case-control studies. *ACM Trans. Inf. Syst. Secur.*, 17, 2014.

[21] Luca Allodi and Fabio Massacci. Security events and vulnerability data for cybersecurity risk estimation. *Risk Analysis*, 37, 2017.

[22] Luca Allodi, Fabio Massacci, and Julian Williams. The work-averse cyberattacker model: Theory and evidence from two million attack signatures. *Risk Analysis*, 42, 2021.

[23] Mohammed Almukaynizi, Alexander Grimm, Eric Nunes, Jana Shakarian, and Paulo Shakarian. Predicting cyber threats through hacker social networks in darkweb and deepweb forums. In *Proc. of CSSSA-17*, 2017.

[24] Mohammed Almukaynizi, Eric Nunes, Krishna Dharaiya, Manoj Senguttuvan, Jana Shakarian, and Paulo Shakarian. Proactive identification of exploits in the wild through vulnerability mentions online. In *Proc. of CyCon-17*, 2017.

[25] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. Atlas: A sequence-based learning approach for attack investigation. In *Proc. of USENIX-21*, 2021.

[26] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proc. of ACM CCS-02*, 2002.

[27] Ross Anderson, Chris Barton, Rainer Bölme, Richard Clayton, Carlos Ganán, Tom Grasso, Michael Levi, Tyler Moore, and Marie Vasek. Measuring the changing cost of cybercrime. In *In Proc. of WEIS-19*, 2019.

[28] Anonymous. Towards a comprehensive picture of the great firewall's DNS censorship. In *Proc. of FOCI-14*, 2014.

[29] Louis Anthony (Tony) Cox Jr. What's wrong with risk matrices? *Risk Analysis*, 28, 2008.

[30] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the mirai botnet. In *Proc. of USENIX-17*, 2017.

[31] Afsah Anwar, Ahmed Abusnaina, Songqing Chen, Frank Li, and David Mohaisen. Cleaning the nvd: Comprehensive quality assessment, improvements, and analyses. In *Proc. of DSN-21*, 2021.

[32] Simone Aonzo, Yufei Han, Alessandro Mantovani, and Davide Balzarotti. Humans vs. machines in malware classification. In *Proc. of USENIX-23*, 2023.

[33] APNIC. Dnssec validation rate by country. `https://stats.labs.apnic.net/dnssec`. Accessed: 2021-09-01.

[34] Waqar Aqeel, Balakrishnan Chandrasekaran, Anja Feldmann, and Bruce M. Maggs. On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement. In *Proc. of IMC-20*, 2020.

[35] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005.

[36] Ashish Arora, Ramayya Krishnan, Rahul Telang, and Yubao Yang. An empirical analysis of software vendors' patch release behavior: Impact of vulnerability disclosure. *Inf. Syst. Res.*, 21, 2010.

[37] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *Proc. of USENIX-22*, 2022.

[38] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. DREBIN: effective and explainable detection of android malware in your pocket. In *Proc. of NDSS-14*, 2014.

[39] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.*, 3, 2000.

[40] Mika Ayenson, Dietrich Wambach, Ashkan Soltani, Nathan Good, and Chris Hoofnagle. Flash cookies and privacy ii: Now with html5 and etag respawning. *Available at SSRN 1898390*, 2011.

[41] Frederick Barr-Smith, Xabier Ugarte-Pedrero, Mariano Graziano, Riccardo Spolaor, and Ivan Martinovic. Survivalism: Systematic analysis of windows malware living-off-the-land. In *Proc. of SSP-21*, 2021.

[42] Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *Proc. of SSP-06*, 2006.

[43] Adam Barth and Brandon Sterne. Content security policy 1.0. Technical report, W3C, 2015. URL: `http://www.w3.org/TR/2015/NOTE-CSP1-20150219/`. Accessed: 2022-11-01.

[44] Muhammad Ahmad Bashir, Sajjad Arshad, William K. Robertson, and Christo Wilson. Tracing information flows between ad exchanges using retargeted ads. In *Proc. of USENIX-16*, 2016.

[45] Muhammad Ahmad Bashir and Christo Wilson. Diffusion of user tracking data in the online advertising ecosystem. *Proc. of PETS-18*, 2018.

[46] Yolanta Beres, Jonathan Griffin, Simon Shiu, Max Heitman, David Markle, and Peter Ventura. Analysing the performance of security solutions to reduce vulnerability exposure window. In *Proc. of ACSAC-08*, 2008.

[47] Rasika Bhalerao, Maxwell Aliapoulios, Ilia Shumailov, Sadia Afroz, and Damon McCoy. Mapping the underground: Supervised discovery of cybercrime supply chains. In *Proc. of APWG eCrime-19*, 2019.

[48] Parth Bhatt, Edgar Toshiro Yano, and Per M. Gustavsson. Towards a framework to detect multi-stage advanced persistent threats attacks. In *Proc. of SOSE-14*, 2014.

[49] Ravi Bhoraskar, Seungyeop Han, Jinseong Jeon, Tanzirul Azim, Shuo Chen, Jaeyeon Jung, Suman Nath, Rui Wang, and David Wetherall. Brahmastra: Driving apps to test the security of third-party components. In *Proc. of USENIX-14*, 2014.

[50] Leyla Bilge and Tudor Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proc. of ACM CCS-12*, 2012.

[51] Leyla Bilge, Yufei Han, and Matteo Dell'Amico. Riskteller: Predicting the risk of cyber incidents. In *Proc. of ACM CCS-17*, 2017.

[52] Reuben Binns, Jun Zhao, Max Van Kleek, and Nigel Shadbolt. Measuring third-party tracker power across web and mobile. *ACM TOIT-18*, 18, 2018.

[53] Sarah Bird, Ilana Segall, and Martin Lopatka. Replication: Why we still can't browse in peace: On the uniqueness and reidentifiability of web browsing histories. In *Proc. of SOUPS-20*, 2020.

[54] Jasmin Christian Blanchette, Sascha Böhme, and Lawrence C. Paulson. Extending sledgehammer with SMT solvers. *J. Autom. Reasoning*, 51, 2013.

[55] Chromium Blog. A safer default for navigation: Https, 2021. URL: `https://blog.chromium.org/2021/03/a-safer-default-for-navigation-https.html`. Accessed: 2022-11-01.

[56] Mozilla Security Blog. Firefox 91 introduces https by default in private browsing, 2021. `https://blog.mozilla.org/security/2021/08/10/firefox-91-introduces-https-by-default-in-private-browsing/`. Accessed: 2022-11-01.

[57] Christopher Bogart, Christian Kästner, James D. Herbsleb, and Ferdian Thung. How to break an API: cost negotiation and community values in three software ecosystems. In *Proc. of FSE'16*, 2016.

[58] Dino Bollinger, Karel Kubicek, Carlos Cotrini, and David Basin. Automating cookie consent and gdpr violation detection. In *Proc. of USENIX-22*, 2022.

[59] Sarah E Bonner. A model of the effects of audit task complexity. *Accounting, organizations and society*, 19, 1994.

[60] Xander Bouwman, Victor Le Pochat, Pawel Foremski, Tom Van Goethem, Carlos H Gañán, Giovane CM Moura, Samaneh Tajalizadehkhoob, Wouter Joosen, and Michel Van Eeten. Helping hands: Measuring the impact of a large threat intelligence sharing community. In *Proc. of USENIX-22*, 2022.

[61] XB Bouwman, Harm Griffioen, Jelle Egbers, Christian Doerr, Bram Klievink, and MJG van Eeten. A different cup of ti? the added value of commercial threat intelligence. In *Proc. of USENIX-20*, 2020.

[62] Mehran Bozorgi, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proc. of SIGKDD-10*, 2010.

[63] Sergey Bratus. What hackers learn that the rest of us don't: notes on hacker curriculum. *IEEE Security & Privacy*, 2007.

[64] Larissa Braz, Christian Aeberhard, Gül Çalikli, and Alberto Bacchelli. Less is more: Supporting developers in vulnerability detection during code review. In *Proc. of ICSE-22*, 2022.

[65] Berndt Brehmer. Dynamic decision making: Human control of complex systems. *Acta psychologica*, 81, 1992.

[66] Guillaume Brogi and Valérie Viet Triem Tong. Terminaptor: Highlighting advanced persistent threats through information flow tracking. In *Proc. of NTMS-16*, 2016.

[67] Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval Estimation for a Binomial Proportion. *Statistical Science*, 16, 2001.

[68] Sam Buss and Grigori Mints. The complexity of the disjunction and existential properties in intuitionistic logic. *Annals of Pure and Applied Logic*, 99, 1999.

[69] Samuel R Buss and Pavel Pudlák. On the computational content of intuitionistic propositional proofs. *Annals of Pure and Applied Logic*, 109, 2001.

[70] Kevin R. B. Butler, Toni R. Farley, Patrick D. McDaniel, and Jennifer Rexford. A survey of BGP security issues and solutions. *Proc. IEEE*, 98, 2010.

[71] Aaron Cahn, Scott Alfeld, Paul Barford, and S. Muthukrishnan. An empirical study of web cookies. In *Proc. of WWW-16*, 2016.

[72] Andrew Caines, Sergio Pastrana, Alice Hutchings, and Paula J Buttery. Automatically identifying the function and intent of posts in underground forums. *Crime Science*, 7, 2018.

[73] Stefano Calzavara, Alvise Rabitti, and Michele Bugliesi. Content security problems?: Evaluating the effectiveness of content security policy in the wild. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proc. of ACM CCS-16*, 2016.

[74] Stefano Calzavara, Sebastian Roth, Alvise Rabitti, Michael Backes, and Ben Stock. A tale of two headers: A formal analysis of inconsistent click-jacking protection on the web. In *Proc. of USENIX-20*, 2020.

[75] Stefano Calzavara, Tobias Urban, Dennis Tatang, Marius Steffens, and Ben Stock. Reining in the web's inconsistencies with site policy. In *Proc. of NDSS-21*, 2021.

[76] Michele Campobasso and Luca Allodi. Impersonation-as-a-service: Characterizing the emerging criminal infrastructure for user impersonation at scale. In *Proc. of ACM CCS-20*, 2020.

[77] Davide Canali and Davide Balzarotti. Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In *Proc. of NDSS-13*, 2013.

[78] Yinzhi Cao, Song Li, and Erik Wijmans. (cross-)browser fingerprinting via OS and hardware level features. In *Proc. of NDSS-17*, 2017.

[79] Saranya Chandran, P. Hrudya, and Prabaharan Poornachandran. An efficient classification model for detecting advanced persistent threat. In *Proc. of ICACCI-15*, 2015.

[80] Mike Chapple. How expensive are ipsec vpn setup costs?, 2017. `http://searchsecurity.techtarget.com/answer/How-expensive-are-IPsec-VPN-setup-costs`. Accessed: 2021-06-01.

[81] Bertil Chapuis, Olamide Omolola, Mauro Cherubini, Mathias Humbert, and Kévin Huguenin. An empirical study of the use of integrity verification mechanisms for web subresources. In *Proc. of WWW-20*, 2020.

[82] Haipeng Chen, Rui Liu, Noseong Park, and VS Subrahmanian. Using twitter to predict when vulnerabilities will be exploited. In *Proc. of SIGKDD-19*, 2019.

[83] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *Proc. of Communications and Multimedia Security*, 2014.

[84] Daiki Chiba, Mitsuaki Akiyama, Yuto Otsuki, Hiroki Hada, Takeshi Yagi, Tobias Fiebig, and Michel van Eeten. Domainprio: Prioritizing domain name investigations to improve SOC efficiency. *IEEE Access*, 10, 2022.

[85] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David R. Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. A longitudinal, end-to-end view of the DNSSEC ecosystem. In *Proc. of USENIX-17*, 2017.

[86] Sandy Clark, Michael Collis, Matt Blaze, and Jonathan M. Smith. Moving targets: Security and rapid-release in firefox. In *Proc. of ACM CCS-14*, 2014.

[87] Cloudflare. Understanding how facebook disappeared from the internet, 2021. `https://blog.cloudflare.com/october-2021-facebook-outage/`. Accessed: 2022-08-30.

[88] Cve-2018-6789, 2018. `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-6789`. Accessed: 2021-06-23.

[89] Elizabeth Costenbader and Thomas W Valente. The stability of centrality measures when networks are sampled. *Social networks*, 25, 2003.

[90] Crowdstrike. The french connection: French aerospacefocused cve-2014-0322 attack shares similarities with 2012 capstone turbine activity, 2014. https://www.crowdstrike.com/blog/french-connection-french-aerospace-focused-cve-2014-0322-attack-shares-similarities-2012/. Accessed: 2020-11-01.

[91] Adrian Dabrowski, Georg Merzdovnik, Johanna Ullrich, Gerald Sendera, and Edgar R. Weippl. Measuring cookies and web privacy in a post-gdpr world. In *Proc. of PAM-19*, 2019.

[92] Savino Dambra, Leyla Bilge, and Davide Balzarotti. Sok: Cyber insurance - technical challenges and a system security roadmap. In *Proc. of SSP-20*, 2020.

[93] Stanislav Dashevskyi, Daniel Ricardo dos Santos, Fabio Massacci, and Antonino Sabetta. TESTREX: a testbed for repeatable exploits. In *Proc. of CSET'14*, 2014.

[94] Alexander Dax and Robert Künnemann. On the soundness of infrastructure adversaries. In *Proc. of IEEE CSF-21*, 2021.

[95] Deloitte. Deloitte contractor site hourly rates. `https://www2.deloitte.com/content/dam/Deloitte/us/Documents/public-sector/us-fed-contractor-site-hourly-rates-10172014.pdf`. Accessed: 2018-10-30.

[96] Henry DeYoung, Deepak Garg, Limin Jia, Dilsun Kirli Kaynar, and Anupam Datta. Experiences in the logical specification of the HIPAA and GLBA privacy laws. In *Proc. of WPES-10*, 2010.

[97] Giorgio Di Tizio, Michele Armellini, and Fabio Massacci. Software updates strategies: a quantitative evaluation against advanced persistent threats. *IEEE Trans. Software Eng.*, 2022.

[98] Giorgio Di Tizio, Alice Hutchings, and Fabio Massacci. A graph-based stratified sampling methodology for the analysis of (underground) forums. *Submitted to IEEE Trans. Inf. Forensics Secur.*, 2022.

[99] Giorgio Di Tizio, Leon Kersten, Martin Rosso, Luca Allodi, and Fabio Massacci. Measuring soc analysts investigation of cyber attacks using the entropy of task complexity. *Submitted to IEEE Trans. Inf. Forensics Secur.*, 2022.

[100] Giorgio Di Tizio, Patrick Speicher, Milivoj Simeonovski, Michael Backes, Ben Stock, and Robert Künnemann. Pareto-optimal defenses for the web infrastructure: Theory and practice. *ACM Trans. Priv. Secur.*, 2022.

[101] Alexandra Dirksen, David Klein, Robert Michael, Tilman Stehr, Konrad Rieck, and Martin Johns. Logpicker: Strengthening certificate transparency against covert adversaries. *Proc. Priv. Enhancing Technol.*, 2021, 2021.

[102] David Dittrich and Erin Kenneally. The menlo report: Ethical principles guiding information and communication technology research. Technical report, US Department of Homeland Security, Aug 2012.

[103] Trinh Viet Doan, Roland van Rijswijk-Deij, Oliver Hohlfeld, and Vaibhav Bajpai. An empirical view on consolidation of the web. *ACM Trans. Internet Tech.*, 22, 2022.

[104] DOJ. Four chinese nationals working with the ministry of state security charged with global computer intrusion campaign targeting intellectual property and confidential business information, including infectious disease research, 2021. https://www.justice.gov/opa/pr/four-chinese-nationals-working-ministry-state-security-charged-global-computer-intrusion. Accessed: 2022-11-01.

[105] Ying Dong, Wenbo Guo, Yueqi Chen, Xinyu Xing, Yuqing Zhang, and Gang Wang. Towards the detection of inconsistencies in public security vulnerability reports. In *Proc. of USENIX-19*, 2019.

[106] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proc. of ACM CCS-17*, 2017.

[107] Tudor Dumitraş and Darren Shou. Toward a standard benchmark for computer security research: The worldwide intelligence network environment (wine). In *Proc. of EuroSys-11 Workshop*, 2011.

[108] Karel Durkota, Viliam Lisý, Branislav Bosanský, and Christopher Kiekintveld. Approximate solutions for attack graph games with imperfect information. In *Proc. of GameSec-15*, 2015.

[109] Karel Durkota, Viliam Lisy, Christofer Kiekintveld, and Branislav Bosansky. Optimal network security hardening using attack graph games. In *Proc. of IJCAI-15*, 2015.

[110] Karel Durkota, Viliam Lisý, Christopher Kiekintveld, Branislav Bosanský, and Michal Pechoucek. Case studies of network defense with attack graph games. *IEEE Intelligent Systems*, 31, 2016.

[111] Durkota, Karel and Lisy, Viliam and Kiekintveld, Christofer and Bosansky, Branislav. Game-theoretic algorithms for optimal network security hardening using attack graphs. In *Proc. of AAMAS-15*, 2015.

[112] EasyList. Easylist overview, 2022. `https://easylist.to/`. Accessed: 2022-30-08.

[113] Gabriel Ebner, Stefan Hetzl, Giselle Reis, Martin Riener, Simon Wolfsteiner, and Sebastian Zivota. System description: Gapt 2.0. 2016.

[114] Peter Eckersley. How unique is your web browser? In *Proc. of PETS-10*, 2010.

[115] Anne Edmundson, Brian Holtkamp, Emanuel Rivera, Matthew Finifter, Adrian Mettler, and David A. Wagner. An empirical study on the effectiveness of security code review. In *Proc. of ESSoS-13*, 2013.

[116] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proc. of ACM CCS-16*, 2016.

[117] Antonino Rullo et al. Pareto optimal security resource allocation for internet of things. *ACM Trans. Priv. Secur.*, 20, 2017.

[118] Edoardo Serra et al. Pareto-optimal adversarial defense of enterprise systems. *ACM Trans. Inf. Syst. Secur.*, 17, 2015.

[119] Orly Stan et al. Heuristic approach for countermeasure selection using attack graphs. In *Proc. of IEEE CSF-21*, 2021.

[120] Sebastian Roth et al. 12 angry developers - A qualitative study on developers' struggles with CSP. In *Proc. of ACM CCS-21*, 2021.

[121] Barry Charles Ezell, Steven P Bennett, Detlof Von Winterfeldt, John Sokolowski, and Andrew J Collins. Probabilistic risk analysis and terrorism risk. *Risk Analysis*, 30, 2010.

[122] FireEye. Think fast: Time between disclosure, patch release and vulnerability exploitation - intelligence for vulnerability management, part two, 2020. `https://www.fireeye.com/blog/threat-research/2020/04/time-between-disclosure-patch-release-and-vulnerability-exploitation.html`. Accessed: 2020-10-01.

[123] Philip J Fleming and John J Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Communications of the ACM*, 29, 1986.

[124] Imane Fouad, Nataliia Bielova, Arnaud Legout, and Natasa Sarafijanovic-Djukic. Tracking the pixels: Detecting web trackers via analyzing invisible pixels. *CoRR*, 2018.

[125] Imane Fouad, Cristiana Santos, Arnaud Legout, and Nataliia Bielova. My cookie is a phoenix: detection, measurement, and lawfulness of cookie respawning with browser fingerprinting. *Proc. Priv. Enhancing Technol.*, 2022, 2022.

[126] Electronic Frontier Foundation. Privacy badger is changing to protect you better, 2020. `https://www.eff.org/deeplinks/2020/10/privacy-badger-changing-protect-you-better`. Accessed: 2022-30-08.

[127] Sylvain Frey, Yehia Elkhatib, Awais Rashid, Karolina Follis, John Vidler, Nicholas J. P. Race, and Christopher Edwards. It bends but would it break? topological analysis of BGP infrastructures in europe. In *Proc. of EuroS&P-16*, 2016.

[128] Ivo Friedberg, Florian Skopik, Giuseppe Settanni, and Roman Fiedler. Combating advanced persistent threats: From network event correlation to incident detection. *Computers & Security*, 2015.

[129] Ebner Gabriel. Herbrand construction for automated intuitionistic theorem proving. In *Proc. of FISP-18*, 2018.

[130] Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. Optimal scheduling of cybersecurity analysts for minimizing risk. *ACM Trans. Intell. Syst. Technol.*, 8, 2017.

[131] Vaibhav Garg, Sadia Afroz, Rebekah Overdorf, and Rachel Greenstadt. Computer-supported cooperative crime. In *Proc. of FC-15*, volume 8975, 2015.

[132] Ibrahim Ghafir, Mohammad Hammoudeh, Vaclav Prenosil, Liangxiu Han, Robert Hegarty, Khaled M. Rabie, and Francisco J. Aparicio-Navarro. Detection of ad-

vanced persistent threat using machine-learning correlation analysis. *Future Generation Comp. Syst.*, 89, 2018.

[133] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice.* Morgan Kaufmann, 2004.

[134] Arpita Ghosh, Mohammad Mahdian, R Preston McAfee, and Sergei Vassilvitskii. To match or not to match: Economics of cookie matching in online advertising. *ACM Trans. on Economics and Computation*, 3, 2015.

[135] Nirnay Ghosh and S. K. Ghosh. An intelligent technique for generating minimal attack graph. In *Proc. Workshop on Intelligent Security*, 2009.

[136] Ghostery. Ghostery faq, 2022. `https://www.ghostery.com/faq`. Accessed: 2022-30-08.

[137] Phillipa Gill, Vijay Erramilli, Augustin Chaintreau, Balachander Krishnamurthy, Konstantina Papagiannaki, and Pablo Rodriguez. Follow the money: understanding economics of online aggregation and advertising. In *Proc. of IMC-13*, 2013.

[138] Paul Giura and Wei Wang. A context-based detection framework for advanced persistent threats. In *Proc. of ICCS-12*, 2012.

[139] Minas Gjoka, Carter T. Butts, Maciej Kurant, and Athina Markopoulou. Multigraph sampling of online social networks. *IEEE J. Sel. Areas Commun.*, 29, 2011.

[140] Andreas Goerdt. *Efficient interpolation for the intuitionistic sequent calculus.* Techn. Univ., Fak. für Informatik, 2000.

[141] Sharon Goldberg. Why is it taking so long to secure internet routing? *Commun. ACM*, 57, 2014.

[142] Richard Gomer, Eduarda Mendes Rodrigues, Natasa Milic-Frayling, and MC Schraefel. Network analysis of third party tracking: User exposure to tracking cookies through search. In *Proc. of WI-IAT-13*, 2013.

[143] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale. In *Proc. of WWW-18*, 2018.

[144] Cleotilde Gonzalez, Polina Vanyukov, and Michael K Martin. The use of microworlds to study dynamic decision making. *Computers in human behavior*, 21, 2005.

[145] Roberto Gonzalez, Lili Jiang, Mohamed Ahmed, Miriam Marciel, Rubén Cuevas, Hassan Metwalley, and Saverio Niccolini. The cookie recipe: Untangling the use of cookies in the wild. In *Proc. of TMA-17*, 2017.

[146] Eric T. Greenlee, Gregory J. Funke, Joel S. Warm, Ben D. Sawyer, Victor S. Finomore, Vince F. Mancuso, Matthew E. Funke, and Gerald Matthews. Stress and workload profiles of network analysis: Not all tasks are created equal. In *Advances in Human Factors in Cybersecurity*, 2016.

[147] Enrico Gregori, Alessandro Improta, Luciano Lenzini, Lorenzo Rossi, and Luca Sani. On the incompleteness of the as-level graph: a novel methodology for BGP route collector placement. In *Proc. of IMC-12*, 2012.

[148] Chris Grier, Kurt Thomas, Vern Paxson, and Chao Michael Zhang. @spam: the underground on 140 characters or less. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proc. of ACM CCS-10*, 2010.

[149] P. Hallam-Baker, R. Stradling, and J. Hoffman-Andrews. DNS Certification Authority Authorization (CAA) Resource Record. RFC 8659, November 2019.

[150] Xueyuan Han, Thomas F. J.-M. Pasquier, Adam Bates, James Mickens, and Margo I. Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. In *Proc. of NDSS-20*, 2020.

[151] Alina Hang, Emanuel von Zezschwitz, Alexander De Luca, and Heinrich Hussmann. Too much information! user attitudes towards smartphone sharing. In *Proc. of NordiCHI-12*, 2012.

[152] Craig M Harvey and Richard J Koubek. Cognitive, social, and environmental attributes of distributed engineering collaboration: A review and proposed model of collaboration. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 10, 2000.

[153] Wajih Ul Hassan, Adam Bates, and Daniel Marino. Tactical provenance analysis for endpoint detection and response systems. In *Proc. of SSP-20*, 2020.

[154] J. Hodges, C. Jackson, and A. Barth. HTTP Strict Transport Security (HSTS). RFC 6797 (Proposed Standard), November 2012.

[155] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard), August 2012.

[156] Geng Hong, Zhemin Yang, Sen Yang, Lei Zhang, Yuhong Nan, Zhibo Zhang, Min Yang, Yuan Zhang, Zhiyun Qian, and Haixin Duan. How you get shot in the back: A systematical study about cryptojacking in the real world. In *Proc. of ACM CCS-18*, 2018.

[157] R. Housley. Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). RFC 4309 (Proposed Standard), December 2005.

[158] Pengfei Hu, Hongxing Li, Hao Fu, Derya Cansever, and Prasant Mohapatra. Dynamic defense strategy against advanced persistent threat with insiders. In *Proc. of INFOCOM-15*, 2015.

[159] Cheng Huang, Yongyan Guo, Wenbo Guo, and Ying Li. Hackerrank: Identifying key hackers in underground forums. *Int. J. Distributed Sens. Networks*, 17, 2021.

[160] Jack Hughes, Ben Collier, and Alice Hutchings. From playing games to committing crimes: A multi-technique approach to predicting key actors on an online gaming forum. In *Proc. of APWG eCrime-19*, 2019.

[161] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of CTI sources. In *Proc. of ACSAC-17*, 2017.

[162] Alice Hutchings. Crime from the keyboard: organised cybercrime, co-offending, initiation and knowledge transmission. *Crime, Law and Social Change*, 2014.

[163] Alice Hutchings and Richard Clayton. Exploring the provision of online booter services. *Deviant Behavior*, 37, 2016.

[164] ICANN. Tld dnssec report, 2021. `http://stats.research.icann.org/dns/tld_report/`. Accessed: 2021-08-01.

[165] Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kâafar, Noha Loizon, and Roya Ensafi. The chain of implicit trust: An analysis of the web third-party resources loading. In *Proc. of WWW-19*, 2019.

[166] Umar Iqbal, Zubair Shafiq, Peter Snyder, Shitong Zhu, Zhiyun Qian, and Benjamin Livshits. Adgraph: A graph-based approach to ad and tracker blocking. In *Proc. of SSP-20*, 2020.

[167] Jay Jacobs, Sasha Romanosky, Idris Adjerid, and Wade Baker. Improving vulnerability remediation through better exploit prediction. In *Proc. of WEIS-19*, 2019.

[168] Sushil Jajodia, Steven Noel, and Brian O'Berry. Topological analysis of network attack vulnerability. In *Managing Cyber Threats: Issues, Approaches and Challenges*. Springer, 2005.

[169] Andrew Jaquith. *Security metrics: replacing fear, uncertainty, and doubt*. Pearson Education, 2007.

[170] Joshua Mervine Justin Dorfman. How to implement sri in your build process, 2016. `https://hacks.mozilla.org/2016/04/how-to-implement-sri-into-your-build-process`. Accessed: 2021-03-01.

[171] Vasiliki Kalavri, Jeremy Blackburn, Matteo Varvello, and Konstantina Papagiannaki. Like a pack of wolves: Community structure of web trackers. In *Proc. of PAM-16*, 2016.

[172] Poul-Henning Kamp. The software industry is still the problem: The time is (also) way overdue for it professional liability. *ACM Queue*, 19(4), 2021.

[173] George Karantzas and Constantinos Patsakis. An empirical assessment of endpoint detection and response systems against advanced persistent threats attack vectors. *Journal of Cybersecurity and Privacy*, 1, 2021.

[174] Aqsa Kashaf, Vyas Sekar, and Yuvraj Agarwal. Analyzing third party service dependencies in modern web services: Have we learned from the mirai-dyn incident? In *Proc. of IMC-20*, 2020.

[175] Kaspersky. Equation group: questions and answers, 2015. `https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064459/Equation_group_questions_and_answers.pdf`. Accessed: 2022-11-01.

[176] Kaspersky, 2020. https://securelist.com/ie-and-windows-zero-day-operation-powerfall/97976/. Accessed: 2020-12-01.

[177] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005.

[178] Hyunyoung Kil, Seog-Chan Oh, Ergin Elmacioglu, Wonhong Nam, and Dongwon Lee. Graph theoretic topological analysis of web service networks. *World Wide Web*, 12, 2009.

[179] Amit Klein and Benny Pinkas. DNS cache-based user tracking. In *Proc. of NDSS-19*, 2019.

[180] Faris Bugra Kokulu, Ananta Soneji, Tiffany Bao, Yan Shoshitaishvili, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. Matched and mismatched socs: A qualitative study on security operations center issues. In *Proc. of ACM CCS-19*, 2019.

[181] Radhesh Krishnan Konoth, Emanuele Vineti, Veelasha Moonsamy, Martina Lindorfer, Christopher Kruegel, Herbert Bos, and Giovanni Vigna. Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense. In *Proc. of ACM CCS-18*, 2018.

[182] Barbara Kordy, Piotr Kordy, Sjouke Mauw, and Patrick Schweitzer. ADTool: security analysis with attack-defense trees. In *Quantitative Evaluation of Systems*, 2013.

[183] Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Patrick Schweitzer. Foundations of attack-defense trees. In *Formal Aspects in Security and Trust*, 2010.

[184] Platon Kotzias, Leyla Bilge, Pierre-Antoine Vervier, and Juan Caballero. Mind your own business: A longitudinal study of threats and vulnerabilities in enterprises. In *Proc. of NDSS-19*, 2019.

[185] Deepak Kumar, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, J. Alex Halderman, and Michael Bailey. Tracking certificate misissuance in the wild. In *Proc. of SSP-18*, 2018.

[186] Howard Kunreuther. Risk analysis and risk management in an uncertain world 1. *Risk Analysis: An International Journal*, 22, 2002.

[187] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, 1977.

[188] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser fingerprinting: A survey. *CoRR*, 2019.

[189] Giuseppe Laurenza and Riccardo Lazzeretti. daptaset: A comprehensive mapping of apt-related data. In *Proc. of FINSEC-19*, 2019.

[190] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962 (Experimental), June 2013.

[191] Ben Laurie. Certificate transparency. *ACM Queue*, 12, 2014.

[192] Lucas Layman, Sylvain David Diffo, and Nico Zazworka. Human factors in webserver log file analysis: a controlled experiment on investigating malicious activity. In *Proc. of HotSoS-14*, 2014.

[193] Hyeonmin Lee, Md Ishtiaq Ashiq, Moritz Müller, Roland van Rijswijk-Deij, Taejoong Chung, et al. Under the hood of {DANE} mismanagement in {SMTP}. In *Proc. of USENIX-22*, 2022.

[194] Hyeonmin Lee, Aniketh Gireesh, Roland van Rijswijk-Deij, Taekyoung Kwon, and Taejoong Chung. A longitudinal and comprehensive study of the DANE ecosystem in email. In *Proc. of USENIX-20*, 2020.

[195] Antoine Lemay, Joan Calvet, François Menet, and José M. Fernandez. Survey of publicly available reports on advanced persistent threat actors. *Computers & Security*, 72, 2018.

[196] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016. In *Proc. of USENIX-16*, 2016.

[197] Vector Guo Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. Reading the tea leaves: A comparative analysis of threat intelligence. In *Proc. of USENIX-19*, 2019.

[198] Xigao Li, Babak Amin Azad, Amir Rahmati, and Nick Nikiforakis. Good bot, bad bot: Characterizing automated browsing activity. In *Proc. of SSP-21*, 2021.

[199] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem A. Beyah. Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *Proc. of ACM CCS-16*, 2016.

[200] Secure List, 2016. `https://securelist.com/cve-2015-2545-overview-of-current-threats/74828/`. Accessed: 2022-11-01.

[201] Bin Liu, Anmol Sheth, Udi Weinsberg, Jaideep Chandrashekar, and Ramesh Govindan. Adreveal: improving transparency into online targeted advertising. In *Proc. of HotNets-XII*, 2013.

[202] Ming Liu, Zhi Xue, Xianghua Xu, Changmin Zhong, and Jinjun Chen. Host-based intrusion detection system with system calls: Review and future trends. *ACM Comput. Surv.*, 51, 2019.

[203] Peng Liu and Zhizhong Li. Task complexity: A review and conceptualization framework. *International Journal of Industrial Ergonomics*, 42, 2012.

[204] Yang Liu, Armin Sarabi, Jing Zhang, Parinaz Naghizadeh, Manish Karir, Michael Bailey, and Mingyan Liu. Cloudy with a chance of breach: Forecasting cyber security incidents. In *Proc. of USENIX-15*, 2015.

[205] Jonathan Lusthaus. Beneath the dark web: Excavating the layers of cybercrime's underground economy. In *Proc. of EuroS&PW-19*, 2019.

[206] Jason Luttgens, Matthew Pepe, and Kevin Mandia. *Incident Response & Computer Forensics*. Cambridge University Press, 2014.

[207] Arun S. Maiya and Tanya Y. Berger-Wolf. Online sampling of high centrality individuals in social networks. In *Proc. of PAKDD-10*, 2010.

[208] Alessandro Mantovani, Simone Aonzo, Yanick Fratantonio, and Davide Balzarotti. Re-mind: a first look inside the mind of a reverse engineer. In *Proc. of USENIX-22*, 2022.

[209] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109, 2016.

[210] Mirco Marchetti, Fabio Pierazzi, Alessandro Guido, and Michele Colajanni. Countering advanced persistent threats through security intelligence and big data analytics. In *Proc. of CyCon-16*, 2016.

[211] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ron Deibert, and Vern Paxson. An analysis of china's "great cannon". In *Proc. of FOCI-15*, 2015.

[212] Veronica Marotta, Vibhanshu Abhishek, and Alessandro Acquisti. Online tracking and publishers' revenues: An empirical analysis. In *Proc. of WEIS-19*, 2019.

[213] Fabio Massacci, Trent Jaeger, and Sean Peisert. Solarwinds and the challenges of patching: Can we ever stop dancing with the devil? *IEEE Secur. Priv.*, 19, 2021.

[214] Célestin Matte, Nataliia Bielova, and Cristiana Santos. Do cookie banners respect my choice? measuring legal compliance of banners from IAB europe's transparency and consent framework. In *Proc. of SSP-20*, 2020.

[215] MaxMind. IP Geolocation and Online Fraud Prevention. `http://dev.maxmind.com/`, 2017.

[216] Michael J. May, Carl A. Gunter, and Insup Lee. Privacy apis: Access control techniques to analyze and verify legal privacy policies. In *Proc. of CSFW-19*, 2006.

[217] MDN. Certificate transparency, 2021. `https://developer.mozilla.org/en-US/docs/Web/Security/Certificate_Transparency`. Accessed: 2021-07-23.

[218] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar R. Weippl. Block me if you can: A large-scale study of tracker-blocking tools. In *Proc. of EuroS&P-17*, 2017.

[219] Michael Meyners. Equivalence tests–a review. *Food quality and preference*, 26, 2012.

[220] Sadegh Momeni Milajerdi, Rigel Gjomemo, Birhanu Eshete, R. Sekar, and V. N. Venkatakrishnan. HOLMES: real-time APT detection through correlation of suspicious information flows. In *Proc. of SSP-19*, 2019.

[221] Dale Miller. Hereditary harrop formulas and logic programming. In *Proc. of CLMPST-87*, 1987.

[222] MITRE. Common vulnerabilities and exposures, 2022. `https://www.cve.org/`.

[223] Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen Carley. Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose. In *Proc. of ICWSM-13*, 2013.

[224] Marti Motoyama, Damon McCoy, Kirill Levchenko, Stefan Savage, and Geoffrey M Voelker. An analysis of underground forums. In *Proc. of IMC-11*, 2011.

[225] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in html5. In *Proc. of W2SP-12*, 2012.

[226] Antonio Nappa, Richard Johnson, Leyla Bilge, Juan Caballero, and Tudor Dumitras. The attack of the clones: A study of the impact of shared code on vulnerability patching. In *Proc. of SSP-15*, 2015.

[227] Kartik Nayak, Daniel Marino, Petros Efstathopoulos, and Tudor Dumitras. Some vulnerabilities are different than others - studying vulnerabilities and attack surfaces in the wild. In *Proc. of RAID-14*, 2014.

[228] George C Necula. Proof-carrying code. In *Proc. of POPL-97*, 1997.

[229] European Network and Information Security Agency. Study of the cost of dnssec deployment, 2009. `https://www.enisa.europa.eu/publications/archive/dnsseccosts`. Accessed: 2021-05-01.

[230] Viet Hung Nguyen, Stanislav Dashevskyi, and Fabio Massacci. An automatic method for assessing the versions affected by a vulnerability. *Empir. Softw. Eng.*, 21, 2016.

[231] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proc. of ACM CCS-12*, 2012.

[232] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proc. of SSP-13*, 2013.

[233] NIST. National vulnerability database, 2022. `https://nvd.nist.gov/`.

[234] NIST. National vulnerability database statistics, 2022. `https://nvd.nist.gov/vuln/search/statistics?form_type=Basic&results_type=statistics&search_type=all&isCpeNameSearch=false`. Accessed: 2022-11-07.

[235] Steven Noel, Matthew Elder, Sushil Jajodia, Pramod Kalapa, Scott O'Hare, and Kenneth Prole. Advances in topological vulnerability analysis. In *Cybersecurity Applications & Technology Conference for Homeland Security*, 2009.

[236] Novetta. Operation blockbuster - unraveling the long thread of the sony attack, 2016. `https://www.operationblockbuster.com/wp-content/uploads/2016/02/Operation-Blockbuster-Report.pdf`. Accessed: 2020-10-01.

[237] Jum C Nunnally. Psychometric theory. *New York*, 1978.

[238] Sean Oesch and Scott Ruoti. That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers. In *Proc. of USENIX-20*, 2020.

[239] Lukasz Olejnik, Claude Castelluccia, and Artur Janc. Why johnny can't browse in peace: On the uniqueness of web browsing history patterns. In *Proc. of HotPETs-12*, 2012.

[240] Lukasz Olejnik, Minh-Dung Tran, and Claude Castelluccia. Selling off user privacy at auction. In *Proc. of NDSS-14*, 2014.

[241] Daniela Seabra Oliveira, Tian Lin, Muhammad Sajidur Rahman, Rad Akefirad, Donovan Ellis, Eliany Perez, Rahul Bobhate, Lois DeLong, Justin Cappos, and Yuriy Brun. API blindspots: Why experienced developers write vulnerable code. In *Proc. of SOUPS-18*, 2018.

[242] Ricardo V. Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. The (in)completeness of the observed internet as-level structure. *IEEE/ACM Trans. Netw.*, 18, 2010.

[243] Judith S. Olson, Jonathan Grudin, and Eric Horvitz. A study of preferences for sharing and privacy. In *Proc. of CHI-05*, 2005.

[244] Jeremiah Onaolapo, Nektarios Leontiadis, Despoina Magka, and Gianluca Stringhini. Socialheisting: Understanding stolen facebook accounts. In *Proc. of USENIX-21*, 2021.

[245] Eric Osterweil, Burt Kaliski, Matt Larson, and Danny McPherson. Reducing the x. 509 attack surface with dnssec's dane. *SATIN*, 12, 2012.

[246] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *Proc. of ACM CCS-06*, 2006.

[247] Andy Ozment and Stuart E. Schechter. Milk or wine: Does software security improve with age? In *Proc. of USENIX-06*, 2006.

[248] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In *Proc. of WWW-19*, 2019.

[249] Ivan Pashchenko, Duc Ly Vu, and Fabio Massacci. A qualitative study of dependency management and its security implications. In *Proc. of ACM CCS-20*, 2020.

[250] Sergio Pastrana, Alice Hutchings, Andrew Caines, and Paula Buttery. Characterizing eve: Analysing cybercrime actors in a large underground forum. In *Proc. of RAID-18*, 2018.

[251] Sergio Pastrana, Alice Hutchings, Daniel R. Thomas, and Juan Tapiador. Measuring ewhoring. In *Proc. of IMC-19*, 2019.

[252] Sergio Pastrana, Daniel R. Thomas, Alice Hutchings, and Richard Clayton. Crimebb: Enabling cybercrime research on underground forums at scale. In *Proc. of WWW-18*, 2018.

[253] Judea Pearl. *Causality.* Cambridge University Press, 2009.

[254] Kexin Pei, Zhongshu Gu, Brendan Saltaformaggio, Shiqing Ma, Fei Wang, Zhiwei Zhang, Luo Si, Xiangyu Zhang, and Dongyan Xu. HERCULE: attack story reconstruction via community discovery on correlated log graph. In *Proc. of ACSAC-16*, 2016.

[255] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. TESSERACT: eliminating experimental bias in malware classification across space and time. In *Proc. of USENIX-19*, 2019.

[256] Ildiko Pete, Jack Hughes, Yi Ting Chua, and Maria Bada. A social network analysis and comparison of six dark web forums. In *Proc. of EuroS&PW*, 2020.

[257] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *New Security Paradigms Workshop*, 1998.

[258] Victor Le Pochat, Tom van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proc. of NDSS-19*, 2019.

[259] Rebecca S. Portnoff, Sadia Afroz, Greg Durrett, Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, Damon McCoy, Kirill Levchenko, and Vern Paxson. Tools for automated analysis of cybercriminal markets. In *Proc. of WWW-17*, 2017.

[260] ProjectZero. 0day "in the wild", 2021. `https://googleprojectzero.blogspot.com/p/0day.html`. Accessed: 2021-05-15.

[261] Python. tldextract 3.1.0, 2020. URL: `https://pypi.org/project/tldextract/`.

[262] Sivaramakrishnan Ramanathan, Anushah Hossain, Jelena Mirkovic, Minlan Yu, and Sadia Afroz. Quantifying the impact of blocklisting in the age of address reuse. In *Proc. of IMC-20*, 2020.

[263] D. Raumer, S. Gallenmüller, P. Emmerich, L. Märdian, and G. Carle. Efficient serving of vpn endpoints on cots server hardware. In *Cloud Networking*, 2016.

[264] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000.

[265] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang. BGP routing stability of popular destinations. In *Proc. of ACM IMW-02*, 2002.

[266] Irwin Reyes, Primal Wiesekera, Abbas Razaghpanah, Joel Reardon, Narseo Vallina-Rodriguez, Serge Egelman, and Christian Kreibich. 'is our children's apps learning?' automatically detecting coppa violations. In *Proc. of IEEE ConPro-17*, 2017.

[267] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. "won't somebody think of the children?" examining COPPA compliance at scale. In *Proc. of PETS-18*, 2018.

[268] RIPE Atlas. Internet data collection system. `https://atlas.ripe.net/`, 2018.

[269] RIPE Stat. Information about specific IP addresses and prefixes, 2017. `https://stat.ripe.net/`.

[270] Peter Robinson. Task complexity, task difficulty, and task production: Exploring interactions in a componential framework. *Applied linguistics*, 22, 2001.

[271] Marco Robol, Elda Paja, Mattia Salnitri, and Paolo Giorgini. Modeling and reasoning about privacy-consent requirements. In *Proc. of PoEM-18*, 2018.

[272] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *Proc. of NSDI-12*, 2012.

[273] Martin Rosso, Michele Campobasso, Ganduulga Gankhuyag, and Luca Allodi. SAIBERSOC: synthetic attack injection to benchmark and evaluate the performance of security operation centers. In *Proc. of ACSAC'20*, 2020.

[274] Christian Rossow. Amplification hell: Revisiting network protocols for ddos abuse. In *Proc. of NDSS-14*, 2014.

[275] Artur Rot and Boguslaw Olszewski. Advanced persistent threats attacks in cyberspace. threats, vulnerabilities, methods of protection. In *FedCSIS 2017*, 2017.

[276] Sebastian Roth, Timothy Barron, Stefano Calzavara, Nick Nikiforakis, and Ben Stock. Complex security policy? A longitudinal analysis of deployed content security policies. In *Proc. of NDSS-20*, 2020.

[277] Sebastian Roth, Stefano Calzavara, Moritz Wilhelm, Alvise Rabitti, and Ben Stock. The security lottery: Measuring client-side web security inconsistencies. In *Proc. of USENIX-22*, 2022.

[278] Ling Rothrock*, CM Harvey, and John Burns. A theoretical framework and quantitative architecture to assess team task complexity in dynamic environments. *Theoretical Issues in Ergonomics Science*, 6, 2005.

[279] Andrew Ruef, Michael W. Hicks, James Parker, Dave Levin, Michelle L. Mazurek, and Piotr Mardziel. Build it, break it, fix it: Contesting secure development. In *Proc. of ACM CCS-16*, 2016.

[280] Kimberly Ruth, Aurore Fass, Jonathan Azose, Mark Pearson, Emma Thomas, Caitlin Sadowski, and Zakir Durumeric. A world wide view of browsing the world wide web. In *Proc. of IMC-22*, 2022.

[281] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *Proc. of USENIX-15*, 2015.

[282] Mohammad Hammas Saeed, Shiza Ali, Jeremy Blackburn, Emiliano De Cristofaro, Savvas Zannettou, and Gianluca Stringhini. Trollmagnifier: Detecting state-sponsored troll accounts on reddit. In *Proc. of SSP-22*, 2022.

[283] Dinuka Sahabandu, Baicen Xiao, Andrew Clark, Sangho Lee, Wenke Lee, and Radha Poovendran. DIFT games: Dynamic information flow tracking games for advanced persistent threats. In *Proc. of IEEE CDC-18*, 2018.

[284] Iskander Sanchez-Rola, Matteo Dell'Amico, Davide Balzarotti, Pierre-Antoine Vervier, and Leyla Bilge. Journey to the center of the cookie ecosystem: Unraveling actors' roles and relationships. In *Proc. of SSP-21*, 2021.

[285] Iskander Sánchez-Rola, Igor Santos, and Davide Balzarotti. Clock around the clock: Time-based device fingerprinting. In *Proc. of ACM CCS-18*, 2018.

[286] SANS. Sans vulnerability management survey 2019, 2019. `https://www.sans.org/reading-room/whitepapers/analyst/membership/38900`. Accessed: 2020-12-01.

[287] SANS. Sans 2022 cyber threat intelligence survey, 2022. `https://www.sans.org/white-papers/sans-2022-cyber-threat-intelligence-survey/`. Accessed: 2022-08-30.

[288] Armin Sarabi, Ziyun Zhu, Chaowei Xiao, Mingyan Liu, and Tudor Dumitras. Patch me if you can: A study on the effects of individual user behavior on the end-host vulnerability state. In *Proc. of PAM-17*, 2017.

[289] Kiavash Satvat, Rigel Gjomemo, and V. N. Venkatakrishnan. EXTRACTOR: extracting attack behavior from threat reports. In *Proc. of EuroS&P-21*, 2021.

[290] Dietram A Scheufele and David Tewksbury. Framing, agenda setting, and priming: The evolution of three media effects models. *Journal of communication*, 57, 2007.

[291] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24, 1999.

[292] Joseph Sexton, Curtis B. Storlie, and Joshua Neil. Attack chain detection. *Statistical Analysis and Data Mining*, 2015.

[293] Ankit Shah, Rajesh Ganesan, and Sushil Jajodia. A methodology for ensuring fair allocation of CSOC effort for alert investigation. *Int. J. Inf. Sec.*, 18, 2019.

[294] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. A methodology to measure and monitor level of operational effectiveness of a CSOC. *Int. J. Inf. Sec.*, 17, 2018.

[295] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. Understanding tradeoffs between throughput, quality, and cost of alert analysis in a csoc. *IEEE IEEE Trans. Inf. Forensics Secur.*, 14, 2019.

[296] Yun Shen and Gianluca Stringhini. ATTACK2VEC: leveraging temporal word embeddings to understand the evolution of cyberattacks. In *Proc. of USENIX-19*, 2019.

[297] Xiaokui Shu, Frederico Araujo, Douglas Lee Schales, Marc Ph. Stoecklin, Jiyong Jang, Heqing Huang, and Josyula R. Rao. Threat intelligence computing. In *Proc. of ACM CCS-18*, 2018.

[298] David Silver, Suman Jana, Dan Boneh, Eric Chen, and Collin Jackson. Password managers: Attacks and defenses. In *Proc. of USENIX-14*, 2014.

[299] Milivoj Simeonovski, Giancarlo Pellegrino, Christian Rossow, and Michael Backes. Who controls the internet?: Analyzing global threats using property graph traversals. In *Proc. of WWW-17*, 2017.

[300] Milivoj Simeonovski, Giancarlo Pellegrino, Christian Rossow, and Michael Backes. Who controls the internet?: Analyzing global threats using property graph traversals. In *Proc. of WWW-17*, 2017.

[301] Gilberto Atondo Siu, Ben Collier, and Alice Hutchings. Follow the money: The relationship between currency exchange and illicit behaviour in an underground forum. In *Proc. of EuroS&PW*, 2021.

[302] Snyk. Open source vulnerability database, 2022. `https://security.snyk.io/`.

[303] Konstantinos Solomos, Panagiotis Ilia, Nick Nikiforakis, and Jason Polakis. Escaping the confines of time: Continuous browser extension fingerprinting through ephemeral modifications. In *Proc. of ACM CCS-22*, 2022.

[304] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash cookies and privacy. In *Proc. of AAAI-10*, 2010.

[305] Kyle Soska and Nicolas Christin. Measuring the longitudinal evolution of the online anonymous marketplace ecosystem. In *Proc. of USENIX-15*, 2015.

[306] Patrick Speicher, Marcel Steinmetz, Michael Backes, Jörg Hoffmann, and Robert Künnemann. Stackelberg planning: Towards effective leader-follower state space search. In *Proc. of AAAI-18*, 2018.

[307] Patrick Speicher, Marcel Steinmetz, Robert Künnemann, Milivoj Simeonovski, Giancarlo Pellegrino, Jörg Hoffmann, and Michael Backes. Formally reasoning about the cost and efficacy of securing the email infrastructure. In *Proc. of EuroS&P-18*, 2018.

[308] Patrick Speicher, Marcel Steinmetz, Robert Künnemann, Milivoj Simeonovski, Giancarlo Pellegrino, Jörg Hoffmann, and Michael Backes. Formally reasoning about the cost and efficacy of securing the email infrastructure. In *Proc. of EuroS&P-18*, 2018.

[309] Sid Stamm, Brandon Sterne, and Gervase Markham. Reining in the web with content security policy. In *Proc. of WWW-10*, 2010.

[310] Statcounter. Desktop browser market share worldwide. `http://gs.statcounter.com/browser-market-share/desktop/worldwide`. Accessed: 2021-07-23.

[311] Richard Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9, 1979.

[312] Marius Steffens, Marius Musch, Martin Johns, and Ben Stock. Who's hosting the block party? studying third-party blockage of csp and sri. In *Proc. of NDSS-21*, 2021.

[313] Marius Steffens, Christian Rossow, Martin Johns, and Ben Stock. Don't trust the locals: Investigating the prevalence of persistent client-side cross-site scripting in the wild. In *Proc. of NDSS-19*, 2019.

[314] Timo Steffens. *Attribution of Advanced Persistent Threats - How to Identify the Actors Behind Cyber-Espionage.* Springer, 2020.

[315] Ivan Stevanovic. Operating system market share – bill gates is still alone at the top!, 2020. `https://kommandotech.com/statistics/operating-system-market-share/`. Accessed: 2021-01-04.

[316] Ben Stock and Martin Johns. Protecting users against xss-based password manager abuse. In *Proc. of AsiaCCS-14*, 2014.

[317] Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitras. Expected exploitability: Predicting the development of functional vulnerability exploits. In *Proc. of USENIX-22*, 2022.

[318] Zhibo Sun, Adam Oest, Penghui Zhang, Carlos E. Rubio-Medrano, Tiffany Bao, Ruoyu Wang, Ziming Zhao, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. Having your cake and eating it: An analysis of concession-abuse-as-a-service. In *Proc. of USENIX-21*, 2021.

[319] Zhibo Sun, Carlos E. Rubio-Medrano, Ziming Zhao, Tiffany Bao, Adam Doupé, and Gail-Joon Ahn. Understanding and predicting private interactions in underground forums. In *Proc. of ACM CODASPY-19*, 2019.

[320] Sathya Chandran Sundaramurthy, Alexandru G. Bardas, Jacob Case, Xinming Ou, Michael Wesch, John McHugh, and S. Raj Rajagopalan. A human capital model for mitigating security analyst burnout. In *Proc. of SOUP-15*, 2015.

[321] Sathya Chandran Sundaramurthy, John McHugh, Xinming Ou, Michael Wesch, Alexandru G. Bardas, and S. Raj Rajagopalan. Turning contradictions into innovations or: How we learned to stop whining and improve security operations. In *Proc. of SOUP-16*, 2016.

[322] Sathya Chandran Sundaramurthy, John McHugh, Xinming Simon Ou, S. Raj Rajagopalan, and Michael Wesch. An anthropological approach to studying CSIRTs. *IEEE Secur. Priv.*, 12, 2014.

[323] Erik Sy, Christian Burkert, Hannes Federrath, and Mathias Fischer. A QUIC look at web tracking. *Proc. Priv. Enhancing Technol.*, 2019, 2019.

[324] Samaneh Tajalizadehkhoob, Tom van Goethem, Maciej Korczynski, Arman Noroozian, Rainer Böhme, Tyler Moore, Wouter Joosen, and Michel van Eeten. Herding vulnerable cats: A statistical approach to disentangle joint responsibility for web security in shared hosting. In *Proc. of ACM CCS-17*, 2017.

[325] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned.* Cambridge University Press, 2011.

[326] MyEtherWallet #teamMEW. A message to our community - a response to the DNS HACK of april 24th 2018, 2018. https://medium.com/@myetherwallet/a-message-to-our-community-a-response-to- the-dns-hack-of-april-24th-2018-26cfe491d31c. Accessed: 2022-11-01.

[327] Steven J. Templeton and Karl E. Levitt. A requires/provides model for computer attacks. In *New Security Paradigms Workshop*, 2000.

[328] ThaiCERT, 2019. `https://www.thaicert.or.th/downloads/files/A_Threat_Actor_Encyclopedia.pdf`.

[329] Giorgio Di Tizio. Drive-by download attacks as a stackelberg planning problem. Master's thesis, University of Trento, 2018.

[330] Giorgio Di Tizio, Michele Armellini, and Fabio Massacci. Advanced Persistent Threats (APTs) campaigns database, 2022. `https://doi.org/10.5281/zenodo.6514817`.

[331] Giorgio Di Tizio and Fabio Massacci. A calculus of tracking: Theory and practice. *Proc. Priv. Enhancing Technol.*, 2021, 2021.

[332] Giorgio Di Tizio, Fabio Massacci, Luca Allodi, Stanislav Dashevskyi, and Jelena Mirkovic. An experimental approach for estimating cyber risk: a proposal building upon cyber ranges and capture the flags. In *Proc. of EuroS&PW-20*, 2020.

[333] Ben Toews. Subresource integrity, 2015. `https://github.blog/2015-09-19-subresource-integrity/`. Accessed: 2022-11-01.

[334] Álvaro Torralba, Patrick Speicher, Robert Künnemann, Marcel Steinmetz, and Jörg Hoffmann. Faster Stackelberg Planning via Symbolic Search and Information Sharing. *Proc. of AAAI-21*, 35, 2021.

[335] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. Beyond the front page: Measuring third party dynamics in the field. In *Proc. of WWW-20*, 2020.

[336] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. Beyond the front page: Measuring third party dynamics in the field. In *Proc. of WWW-20*, 2020.

[337] Tobias Urban, Matteo Große-Kampmann, Dennis Tatang, Thorsten Holz, and Norbert Pohlmann. Plenty of phish in the sea: Analyzing potential pre-attack surfaces. In *Proc. of ESORICS-20*, 2020.

[338] Martin Ussath, David Jaeger, Feng Cheng, and Christoph Meinel. Advanced persistent threats: Behind the scenes. In *CISS 2016*, 2016.

[339] Olivier van der Toorn, Johannes Krupp, Mattijs Jonker, Roland van Rijswijk-Deij, Christian Rossow, and Anna Sperotto. Anyway: Measuring the amplification ddos potential of domains. In *Proc. of CNSM-21*, 2021.

[340] Thijs van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. Deepcase: Semi-supervised contextual analysis of security events. In *Proc. of SSP-22*, 2022.

[341] Rolf Van Wegberg, Fieke Miedema, Ugur Akyazi, Arman Noroozian, Bram Klievink, and Michel van Eeten. Go see a specialist? predicting cybercrime sales on online anonymous markets from vendor and product characteristics. In *Proc. of WWW-20*, 2020.

[342] Rolf van Wegberg, Samaneh Tajalizadehkhoob, Kyle Soska, Ugur Akyazi, Carlos Hernandez Gañán, Bram Klievink, Nicolas Christin, and Michel van Eeten. Plug and prey? measuring the commoditization of cybercrime via online anonymous markets. In *Proc. of USENIX-18*, 2018.

[343] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. Fp-scanner: The privacy implications of browser fingerprint inconsistencies. In *Proc. of USENIX-18*, 2018.

[344] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. FP-STALKER: tracking browser fingerprint evolutions. In *Proc. of SSP-18*, 2018.

[345] Verizon. Data breach investigations report, 2022. `https://www.verizon.com/business/en-gb/resources/2022-data-breach-investigations-report-dbir.pdf`. Accessed: 2022-11-01.

[346] Mathew Vermeer, Michel van Eeten, and Carlos Gañán. Ruling the rules: Quantifying the evolution of rulesets, alerts and incidents in network intrusion detection. In *Proc. of AsiaCCS-22*, 2022.

[347] Nikos Virvilis and Dimitris Gritzalis. The big four - what we did wrong in advanced persistent threat detection? In *Proc. of ARES-13*, 2013.

[348] Daniel Votipka, Kelsey R. Fulton, James Parker, Matthew Hou, Michelle L. Mazurek, and Michael Hicks. Understanding security mistakes developers make: Qualitative analysis from build it, break it, fix it. In *Proc. of USENIX-20*, 2020.

[349] Daniel Votipka, Seth M. Rabin, Kristopher K. Micinski, Jeffrey S. Foster, and Michelle L. Mazurek. An observational investigation of reverse engineers' processes. In *Proc. of USENIX-20*, 2020.

[350] Daniel Votipka, Rock Stevens, Elissa M. Redmiles, Jeremy Hu, and Michelle L. Mazurek. Hackers vs. testers: A comparison of software vulnerability discovery processes. In *Proc. of SSP-18*, 2018.

[351] Anh V. Vu, Jack Hughes, Ildiko Pete, Ben Collier, Yi Ting Chua, Ilia Shumailov, and Alice Hutchings. Turning up the dial: the evolution of a cybercrime market through set-up, stable, and covid-19 eras. In *Proc. of IMC-20*, 2020.

[352] Claudia Wagner, Philipp Singer, Fariba Karimi, Jürgen Pfeffer, and Markus Strohmaier. Sampling from social networks with attributes. In *Proc. of WWW-17*, 2017.

[353] Gerry Wan, Liz Izhikevich, David Adrian, Katsunari Yoshioka, Ralph Holz, Christian Rossow, and Zakir Durumeric. On the origin of scanning: The impact of location on internet-wide scans. In *Proc. of IMC-20*, 2020.

[354] Chuhan Wang, Kaiwen Shen, Minglei Guo, Yuxuan Zhao, Mingming Zhang, Jianjun Chen, Baojun Liu, Xiaofeng Zheng, Haixin Duan, Yanzhong Lin, et al. A large-scale and longitudinal measurement study of {DKIM} deployment. In *Proc. of USENIX-22*, 2022.

[355] Lukas Weichselbaum, Michele Spagnuolo, Sebastian Lekies, and Artur Janc. CSP is dead, long live csp! on the insecurity of whitelists and the future of content security policy. In *Proc. of ACM CCS-16*, 2016.

[356] Joel Weinberger, Frederik Braun, Devdatta Akhawe, and Francois Marier. Subresource integrity. Technical report, W3C, 2016. `http://www.w3.org/TR/2016/REC-SRI-20160623/`.

[357] Michael Weissbacher, Tobias Lauinger, and William K. Robertson. Why is CSP failing? trends and challenges in CSP adoption. In Angelos Stavrou, Herbert Bos, and Georgios Portokalidis, editors, *Proc. of RAID-14*, 2014.

[358] Miuyin Yong Wong, Matthew Landen, Manos Antonakakis, Douglas M. Blough, Elissa M. Redmiles, and Mustaque Ahamad. An inside look into the practice of malware analysis. In *Proc. of ACM CCS-21*, 2021.

[359] Robert E Wood. Task complexity: Definition of the construct. *Organizational behavior and human decision processes*, 37, 1986.

[360] Daniel W Woods and Rainer Böhme. Systematization of knowledge: Quantifying cyber risk. In *Proc. of S&P-21*, 2021.

[361] Qiushi Wu, Yue Xiao, Xiaojing Liao, and Kangjie Lu. Os-aware vulnerability prioritization via differential severity analysis. In *Proc. of USENIX-22*, 2022.

[362] Chaowei Xiao, Armin Sarabi, Yang Liu, Bo Li, Mingyan Liu, and Tudor Dumitras. From patching delays to infection symptoms: Using risk profiles for an early discovery of vulnerabilities exploited in the wild. In *Proc. of USENIX-18*, 2018.

[363] L. Yang, P. Li, X. Yang, and Y. Y. Tang. A risk management approach to defending against the advanced persistent threat. *IEEE Trans. Dependable Secur. Comput.*, 17, 2018.

[364] Seok-Ho Yoon, Ki-Nam Kim, Jiwon Hong, Sang-Wook Kim, and Sunju Park. A community-based sampling method using DPL for online social networks. *Inf. Sci.*, 306, 2015.

[365] Kan Yuan, Haoran Lu, Xiaojing Liao, and XiaoFeng Wang. Reading thieves' cant: Automatically identifying and understanding dark jargons from cybercrime marketplaces. In *Proc. of USENIX-18*, 2018.

[366] Jing Zhang, Zakir Durumeric, Michael Bailey, Mingyan Liu, and Manish Karir. On the mismanagement and maliciousness of networks. In *Proc. of NDSS-14*, 2014.

[367] Yiming Zhang, Yujie Fan, Yanfang Ye, Liang Zhao, Jiabin Wang, Qi Xiong, and Fudong Shao. Kadetector: Automatic identification of key actors in online hack forums based on structured heterogeneous information network. In *ICBK-18*, 2018.

[368] Junzhou Zhao, Pinghui Wang, John C. S. Lui, Don Towsley, and Xiaohong Guan. Sampling online social networks by random walk with indirect jumps. *Data Min. Knowl. Discov.*, 33, 2019.

[369] Wentao Zhao, Pengfei Wang, and Fan Zhang. Extended petri net-based advanced persistent threat analysis model. In *Computer Engineering and Networking*. 2014.

[370] Ziming Zhao, Gail-Joon Ahn, Hongxin Hu, and Deepinder Mahi. Socialimpact: Systematic analysis of underground social dynamics. In *Proc. of ESORICS-12*, 2012.

[371] Chen Zhong, John Yen, Peng Liu, and Robert F. Erbacher. Automate cybersecurity data triage by leveraging human analysts' cognitive process. In *Proc. of IDS-16*, 2016.

[372] Chen Zhong, John Yen, Peng Liu, Robert F. Erbacher, Christopher Garneau, and Bo Chen. Studying analysts' data triage operations in cyber defense situational analysis. In *Theory and Models for Cyber Situation Awareness*, volume 10030. 2017.

[373] Carson Zimmerman. *Ten Strategies of a World-Class Cybersecurity Operations Center*. MITRE, 2014.

Appendix A

IFOL Rules for Web Tracking

We have the following rules for *intuitionistic* logic:

$$\frac{}{A \vdash A}\ (Ax) \qquad \frac{\mathcal{N} \vdash A \qquad A, \mathcal{N} \vdash B}{\mathcal{N} \vdash B}\ (Cut) \qquad \frac{\mathcal{N}, A, A \vdash B}{\mathcal{N}, A \vdash B}\ (CL)$$

$$\frac{\mathcal{N} \vdash B}{\mathcal{N}, A \vdash B}\ (WL) \qquad \frac{\mathcal{N} \vdash}{\mathcal{N} \vdash B}\ (WR) \qquad \frac{\mathcal{N} \vdash A}{\mathcal{N}, \neg A \vdash}\ (\neg L) \qquad \frac{\mathcal{N}, A \vdash}{\mathcal{N} \vdash \neg A}\ (\neg R)$$

$$\frac{\mathcal{N}, A, B \vdash C}{\mathcal{N}, A \wedge B \vdash C}\ (\wedge L) \qquad \frac{\mathcal{N} \vdash A \qquad \mathcal{N} \vdash B}{\mathcal{N} \vdash A \wedge B}\ (\wedge R)$$

$$\frac{\mathcal{N}, A \vdash C \qquad \mathcal{N}, B \vdash C}{\mathcal{N}, A \vee B \vdash C}\ (\vee L) \qquad \frac{\mathcal{N} \vdash A}{\mathcal{N} \vdash A \vee B}\ (\vee R_1) \qquad \frac{\mathcal{N} \vdash B}{\mathcal{N} \vdash A \vee B}\ (\vee R_1)$$

$$\frac{\mathcal{N} \vdash A \qquad \mathcal{N}, B \vdash C}{\mathcal{N}, A \to B \vdash C}\ (\to L) \qquad \frac{\mathcal{N}, A \vdash B}{\mathcal{N} \vdash A \to B}\ (\to R)$$

In our derivations, we do use neither $\vee R_i$ nor $\vee L$ rules as we are only interested in deriving knowledge predicates.

Construct Proof of Tracking

We show that from a derivation we can reconstruct the connections responsible for the tracking.

Theorem 2 (**Map proofs to configurations**). *Given a derivation of* $Knows(w^*, w)$ *from an internet snapshot* $\mathcal{N}$ *(*$\mathcal{N} \vdash Knows(w^*, w)$*), one can extract an* essential *subset of the configuration* $\mathcal{N}_\omega \subseteq \mathcal{N}$ *such that* $\mathcal{N} \setminus \mathcal{N}_\omega \nvdash Knows(w^*, w)$. ■

Proof. This result follows from the existence of uniform proofs[1] for the fragment of interest [69] and the existence of a feasible interpolation for intuitionistic logic [68, 140]. Given a derivation of $\mathcal{N} \vdash \mathit{Knows}(w^*, w)$ one can construct a uniform proof and the existence of the interpolant guarantees that we have a set of formulae that only includes constants shared from the antecedent (the internet configuration) and the succedent (the knowledge predicate). Hence we can use the proof to reconstruct the tracking steps and data exchanges responsible for $\mathit{Knows}(w^*, w)$ in a subset $\mathcal{N}_1$, as the predicates present in the proof and the interpolant. We can then eliminate $\mathcal{N}_1$ from $\mathcal{N}$ and try to derive $\mathcal{N} \backslash \mathcal{N}_1 \vdash \mathit{Knows}(w^*, w)$. If we succeed, it means there is another way to exchange data, so we extract a new subset $\mathcal{N}_2$ and continue the process until for $\mathcal{N}_i$, $i = 1 \ldots$ no derivation is possible. As deciding a single query is decidable in polynomial time (See Theorem 1) the process terminates after a polynomial number of interactions. The union of all sets $\mathcal{N}_i$ is the desired set $\mathcal{N}_\omega$. ■

As immediate from the proof above, one could also stop the search as soon as the first subset of the internet snapshot, $\mathcal{N}_1$, responsible for the tracking is identified. This is what we do with a theorem prover.

A.0.1 Examples of Complex Tracking Interactions

Figures A.1a, A.1b and A.2 show different cases of tracking. Figure A.1a describes the use of first-party cookies on a website. Figures A.1b and A.2 describe the practice of tracking carried out by third-party websites. In the first case, the tracker is directly present on the website, while in the latter it is included by a third-party website. Figure A.3a describes the leaf of recursion where website w accesses itself because either it uses content from itself or it redirects to its content. Figure A.3b describes how to propagate user's information through websites. The browser is forced to load the resources from w, the resources used by w from w', and the resources used by w' from w''. We can also model particular cases where w shares cookies with w' ($\mathit{Access}_{\mathit{cookie}}(w, w')$), but w' does not propagate its cookies to w'' ($\mathit{Link}(w', w'')$). Thus, we obtain ($\mathit{Access}(w, w'')$).

Figure A.4 illustrates how cookie syncing allows an attacker to track users on websites where it is not explicitly present.

[1]A finite constructive process applies uniformly to every formula, either producing an intuitionistic proof of the formula or demonstrating that no such proof can exist.

$$\dfrac{\mathit{Visits}(w) \qquad \dfrac{\mathit{Link}(w, w) \quad \neg \mathit{Block\_request}(w)}{\mathit{Access}(w, w)} \qquad \neg \mathit{Block\_tp\_cookie}(w)}{\mathit{Knows}(w, w)}$$

If a user visits a website w that is allowed to store cookies, then w can know that the user visited it. This is a special case of `3rdpartyTracking` in Figure 3.1a. In this case $\neg \mathit{Block\_tp\_cookie}(w)$ is always true because there are not 3rd-party cookies.

$$\dfrac{\dfrac{\mathit{Link}(w, w') \quad \neg \mathit{Block\_request}(w')}{\mathit{Access}(w, w')} \qquad \mathit{Visits}(w) \qquad \neg \mathit{Block\_tp\_cookie}(w')}{\mathit{Knows}(w', w)}$$

If a user visits a website w that forces to access resources from another website w', then if the website w' is not blocked by any mitigation, it can know that the user visited w.

Figure A.1: Knows Visits Derivations

$$\dfrac{\mathit{Visits}(w) \qquad \dfrac{\dfrac{\mathit{Link}(w, w') \quad \neg \mathit{Block\_request}(w')}{\mathit{Access}(w, w')} \quad \dfrac{\mathit{Link}(w', w'') \quad \neg \mathit{Block\_request}(w'')}{\mathit{Access}(w', w'')}}{\mathit{Access}(w, w'')} \qquad \neg \mathit{Block\_tp\_cookie}(w'')}{\mathit{Knows}(w'', w)}$$

If a user visits a website w that accesses resources from a 3rd-party website w', the website may not only track the user but it can also redirect (include) another website w'' that can set its cookie if no mitigation blocks it. This situation describes both *Third parties that include trackers* and *Basic tracking initiated by a tracker* [124], where w' tracks/does not track w (it can be verified with the rule `3rdpartyTracking`).

Figure A.2: Knows by external trackers

$$\dfrac{\mathit{Link}(w, w) \quad \neg \mathit{Block\_request}(w)}{\mathit{Access}(w, w)} \quad \dfrac{\mathit{Link}_{\mathit{cookie}}(w, w) \quad \neg \mathit{Block\_request}(w)}{\mathit{Access}_{\mathit{cookie}}(w, w)}$$

It is the leaf of the derivation corresponding to the access of a sequence of resources.

$$\dfrac{\dfrac{\mathit{Link}(w, w') \quad \neg \mathit{Block\_request}(w')}{\mathit{Access}(w, w')} \qquad \mathit{Link}(w', w'') \quad \neg \mathit{Block\_request}(w'')}{\mathit{Access}(w, w'')}$$

If a website w accesses resources of website w', and w' has a link to content from w'', then there is an access between w and w'' only if w'' is not blocked by any extension. We can also use $\mathit{Access}_{\mathit{cookie}}(w, w')$ and $\mathit{Link}_{\mathit{cookie}}(w', w'')$ to describe a link with exchange of cookies between w and w'' ($\mathit{Link}_{\mathit{cookie}}(w, w'')$).

Figure A.3: Network Interactions Derivations

$$\dfrac{\mathit{Knows}(w', w) \qquad \dfrac{\dfrac{\mathit{Link}_{\mathit{cookie}}(w', w'') \quad \neg \mathit{Block\_request}(w'')}{\mathit{Access}_{\mathit{cookie}}(w', w'')} \qquad \neg \mathit{Block\_tp\_cookie}(w'')}{\mathit{Cookie\_sync}(w', w'')}}{\mathit{Knows}(w'', w)}$$

w' can track a user on w and redirects the user to another 3rd-party website w'' inserting the cookie information. The two 3rd-party websites can share their cookies and w'' can track users on w even if it is not directly embedded. This situation can be mitigated if either w' or w'' are either blocked by an extension or cannot set cookies. This is called *3rd-2-3rd party cookie syncing*.

Figure A.4: Known by external trackers via Cookie Syncing

A.0.2 Example of Proof

We employ the *getLKProof* method in `Slakje` to generate a proof as a sequence of sequents. The proof can be visualized using the *prooftool* of GAPT, however, it is extremely verbose. Figure A.5 shows the proof for $\mathit{req\_COPPA}(\mathit{flashtalking.com})$, where PII can be potentially collected from *thesaurus.com*.

$$\dfrac{}{\mathit{includeContent}(\mathit{thes},\mathit{flt}) \vdash \mathit{includeContent}(\mathit{thes},\mathit{flt})}\ \mathit{ax} \qquad \dfrac{}{\mathit{link}(\mathit{thes},\mathit{flt}) \vdash \mathit{link}(\mathit{thes},\mathit{flt})}\ \mathit{ax}$$

$$\dfrac{\mathit{includeContent}(\mathit{thes},\mathit{flt}) \vdash \mathit{includeContent}(\mathit{thes},\mathit{flt}) \qquad \mathit{link}(\mathit{thes},\mathit{flt}) \vdash \mathit{link}(\mathit{thes},\mathit{flt})}{\mathit{includeContent}(\mathit{thes},\mathit{flt}) \vdash \mathit{link}(\mathit{thes},\mathit{flt})}\ \rightarrow: l, \forall l, \forall l, \mathit{IncludeW}$$

$$\dfrac{}{\neg \mathit{block\_requests}(\mathit{flt}) \vdash \neg \mathit{block\_requests}(\mathit{flt})}\ \mathit{ax}$$

$$\dfrac{\mathit{includeContent}(\mathit{thes},\mathit{flt}) \vdash \mathit{link}(\mathit{thes},\mathit{flt}) \qquad \neg \mathit{block\_requests}(\mathit{flt}) \vdash \neg \mathit{block\_requests}(\mathit{flt})}{\mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}) \vdash \mathit{link}(\mathit{thes},\mathit{flt}) \land \neg \mathit{block\_requests}(\mathit{flt})}\ \land : R$$

$$\dfrac{}{\mathit{access}(\mathit{thes},\mathit{flt}) \vdash \mathit{access}(\mathit{thes},\mathit{flt})}\ \mathit{ax}$$

$$\dfrac{\mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}) \vdash \mathit{link}(\mathit{thes},\mathit{flt}) \land \neg \mathit{block\_requests}(\mathit{flt}) \qquad \mathit{access}(\mathit{thes},\mathit{flt}) \vdash \mathit{access}(\mathit{thes},\mathit{flt})}{\mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}) \vdash \mathit{access}(\mathit{thes},\mathit{flt})}\ \rightarrow: l, \forall l, \forall l, \mathit{AccessToW}$$

$$\dfrac{}{\mathit{visit}(\mathit{thes}) \vdash \mathit{visit}(\mathit{thes})}\ \mathit{ax}$$

$$\dfrac{\mathit{visit}(\mathit{thes}) \vdash \mathit{visit}(\mathit{thes}) \qquad \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}) \vdash \mathit{access}(\mathit{thes},\mathit{flt})}{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}) \vdash \mathit{visit}(\mathit{thes}) \land \mathit{access}(\mathit{thes},\mathit{flt})}\ \land : R$$

$$\dfrac{}{\neg \mathit{block\_tp\_cookie}(\mathit{flt}) \vdash \neg \mathit{block\_tp\_cookie}(\mathit{flt})}\ \mathit{ax}$$

$$\dfrac{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}) \vdash \mathit{visit}(\mathit{thes}) \land \mathit{access}(\mathit{thes},\mathit{flt}) \qquad \neg \mathit{block\_tp\_cookie}(\mathit{flt}) \vdash \neg \mathit{block\_tp\_cookie}(\mathit{flt})}{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}), \neg \mathit{block\_tp\_cookie}(\mathit{flt}) \vdash (\mathit{visit}(\mathit{thes}) \land \mathit{access}(\mathit{thes},\mathit{flt})) \land \neg \mathit{block\_tp\_cookie}(\mathit{flt})}\ \land : R$$

$$\dfrac{}{\mathit{knows}(\mathit{flt},\mathit{thes}) \vdash \mathit{knows}(\mathit{flt},\mathit{thes})}\ \mathit{ax}$$

$$\dfrac{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}), \neg \mathit{block\_tp\_cookie}(\mathit{flt}) \vdash (\mathit{visit}(\mathit{thes}) \land \mathit{access}(\mathit{thes},\mathit{flt})) \land \neg \mathit{block\_tp\_cookie}(\mathit{flt}) \qquad \mathit{knows}(\mathit{flt},\mathit{thes}) \vdash \mathit{knows}(\mathit{flt},\mathit{thes})}{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}), \neg \mathit{block\_tp\_cookie}(\mathit{flt}) \vdash \mathit{knows}(\mathit{flt},\mathit{thes})}\ \rightarrow: l, \forall : l, \forall : l, \mathit{3rdpartyTracking}$$

$$\dfrac{}{\mathit{kids}(\mathit{thes}) \vdash \mathit{kids}(\mathit{thes})}\ \mathit{ax}$$

$$\dfrac{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}), \neg \mathit{block\_tp\_cookie}(\mathit{flt}) \vdash \mathit{knows}(\mathit{flt},\mathit{thes}) \qquad \mathit{kids}(\mathit{thes}) \vdash \mathit{kids}(\mathit{thes})}{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}), \neg \mathit{block\_tp\_cookie}(\mathit{flt}) \vdash \mathit{knows}(\mathit{flt},\mathit{thes}) \land \mathit{kids}(\mathit{thes})}\ \land : R$$

$$\dfrac{}{\mathit{req\_coppa}(\mathit{flt}) \vdash \mathit{req\_coppa}(\mathit{flt})}\ \mathit{ax}$$

$$\dfrac{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}), \neg \mathit{block\_tp\_cookie}(\mathit{flt}) \vdash \mathit{knows}(\mathit{flt},\mathit{thes}) \land \mathit{kids}(\mathit{thes}) \qquad \mathit{req\_coppa}(\mathit{flt}) \vdash \mathit{req\_coppa}(\mathit{flt})}{\mathit{visit}(\mathit{thes}), \mathit{includeContent}(\mathit{thes},\mathit{flt}), \neg \mathit{block\_requests}(\mathit{flt}), \neg \mathit{block\_tp\_cookie}(\mathit{flt}), \mathit{kids}(\mathit{thes}) \vdash \mathit{req\_coppa}(\mathit{flt})}\ \rightarrow: l, \forall : l, \forall : l, \mathit{COPPAcomplColl}$$

Figure A.5: Proof of *req_COPPA*(*flt* = *flashtalking.com*), where *thes* is the children-related website *thesaurus.com*

Appendix B

Rules for Web Infrastructure Model

This appendix contains the complete threat model used to describe Web-based attacks. Table B.1 contains the entire list of predicates used in the model. Each predicate in the *precondition* of an action is in conjunction with the other predicates; the presence of disjunctions in a rule is used as shorthand to represent different rules for the same action with a shared part in the *precondition*.

Table B.1: Threat Model Predicates

Predicate	Description
$x \xrightarrow{\text{loc}} cn$	$x \in AS \cup IP \cup D \cup NS$ is located in $cn \in Country$
$d \xrightarrow{\text{A}} i$	$d \in D \cup NS$ has address $i \in IP$
$i \xrightarrow{\text{orig}} a$	$i \in IP$ belongs to $a \in AS$
$c \xrightarrow{\text{JS}} d$	$d \in D$ contains JS scripts hosted in the element $c \in D$
$avail\_over\_HTTPS(c, d)$	The JS resources retrieved from c by d are available over HTTPS
$e \xrightarrow{\text{DNS}} d$	$e \in NS$ is one of the authoritative name servers of d
$p \xrightarrow{parent\_zone} e$	$p \in NS$ manages the parent zone of the element $e \in NS$
$a \xrightarrow{\text{RTE(b)}} c$	Given $a, b, c \in AS$, the route from a to c passes through b
$C(x)$	$x \in AS \cup IP \cup D \cup Country \cup NS \cup CA$ is compromised. In case $x \in D \cup NS$, x can be used to directly (indirectly) affect user's visits (*Globally compromised*)
$C^{\text{web}}(d)$	The website hosted on $d \in D$ is compromised. JS included from d is not necessarily compromised as well (*Website access compromised*).
$C^{\text{web}}(c, d)$	The website on $d \in D$ is considered compromised for all the visitors from $c \in Country$. (*Website access compromised from* c)
$XSS(d)$	$d \in D$ is vulnerable to XSS
$Upgrade\ Requests(d)$	$d \in D$ employs the field `upgrade-insecure-requests` in the CSP to force HTTPS for all the resource requests.

$SRI(d, c)$	$d \in D$ implements the Sub-Resource Integrity mitigation for **all** the resources, used by d, stored in $c \in D$. It is assumed $d \neq c$
$IPsec(a, b)$	The packets routed between $a \in AS$ and $b \in AS$ are protected via IPsec
$DNSSEC(f)$	The element $f \in NS$ implements DNSSEC
$HTTPS(d)$	The element $d \in D$ implements HTTPS
$l\_HTTPS(d, e)$	**All** the JS resources, used by $d \in D$ and hosted in $e \in D$, are explicitly using HTTPS in the source code
$l\_HTTPS\_\ compat(d, e)$	**All** the JS resources, used by $d \in D$ and hosted in $e \in D$, are either explicitly using HTTPS in the source code or a protocol-relative URL
$HSTS(d)$	$d \in D$ implements the header `strict-transport-security`
$Redirect(d)$	$d \in D$ redirects HTTP connections to HTTPS. The redirection is either temporary or permanent
$CT(d)$	The digital certificates, for $d \in D$, are signed by CAs that are compliant with the CT
$DANE(d)$	$d \in NS$ implements DANE
$I^{\mathrm{DNS}}(d)$	The DNS resolution of $d \in D$ is compromised (*Globally compromised DNS*).
$I^{\mathrm{DNS}}(d, e)$	The DNS resolution of $d \in D$ is compromised for the visitors from $c \in Country$ (*Country compromised DNS*)
$I^{\mathrm{R}}(i, j)$	The route between $i, j \in IP$ is compromised
$I^{\mathrm{CA}}(d)$	$d \in D$ is vulnerable to certificate authority attacks
$TLSA\_0(a), TLSA\_2(a)$	$a \in CA$ is present in the certificate chain of the TLSA record with certificate usage field 0 or 2

Using the notation of the Boolean Logic, the symbol $\neg$ in front of a predicate negates the predicate itself.

B.0.1 Attacker rules

This section describes the propagation rules for the attacker used in the threat model. We provide each rule, followed by the intuition of what kind of attack it represents.

Initially Compromised Nodes

$$\frac{\begin{array}{c} x \in AS \cup IP \cup NS \cup CA \\ cn \in Country \quad x \xrightarrow{\mathrm{loc}} cn \quad C(cn) \end{array}}{C(x)} \tag{B.1}$$

`Intuition:` All the autonomous systems, IPs, name servers and certificate authorities associated to a malicious country are under the control of the attacker.

$$\frac{i \in IP \quad a \in AS \quad i \xrightarrow{\text{orig}} a \quad C(a)}{C(i)} \tag{B.2}$$

`Intuition:` All the IPs, that belong to an autonomous system compromised by the attacker, are considered under the control of the attacker.

$$\frac{i \in IP \quad d \in D \cup NS \quad d \xrightarrow{\text{A}} i \quad C(i)}{C(d)} \tag{B.3}$$

`Intuition:` If a domain (name server) resolves to an IP address under the control of the attacker, then also the domain (name server) is considered compromised.

$$\frac{i \in IP \quad d \in D \cup NS \quad d \xrightarrow{\text{A}} i \quad C(d)}{C(i)} \tag{B.4}$$

`Intuition:` The same applies in the opposite direction. If a domain or NS is compromised, the corresponding IP is also considered compromised.

Content Compromise

$$\frac{d \in D \quad XSS(d)}{C^{\text{web}}(d)} \tag{B.5}$$

`Intuition:` If a web server is vulnerable to XSS attacks then the attacker can gain control of the content of the website. We did not consider using CSP because its impact on the functionality of a website is currently not measurable. For example, CDNs often inject scripts in websites and thus the cost of deploying a CSP can hardly be measured [312].

DNS Compromise

$$\frac{d \in D \quad e \in NS \quad e \xrightarrow{DNS} d \quad C(e)}{I^{\text{DNS}}(d)} \tag{B.6}$$

`Intuition:` If one of the authoritative name servers of a domain is under the control of the attacker, then the DNS resolution for this domain is considered compromised[1]. An attacker can modify the DNS resolution and map the domain name to a different IP.

[1] Due to the fact that there is no information about which authoritative NS is queried by a client, this is a simplification implemented in the model. Furthermore, if the attacker is able to compromise one of the authoritative NS for a domain, it is possible that it is also able to compromise the other NSs.

Route Compromise

$$\frac{\begin{array}{c} a,b,c \in AS \quad a \neq b \neq c \quad C(b) \\ a \xrightarrow{RTE(b)} c \quad \neg IPsec(a,c) \quad i \xrightarrow{orig} a \quad j \xrightarrow{orig} c \end{array}}{I^{\mathrm{R}}(i,j)} \tag{B.7}$$

`Intuition:` If a route from one AS to another is IPsec protected and passes through a third AS under the control of the attacker, then the route is insecure and the two endpoints of the communication could be targeted by an attack.

This rule does not consider the case in which the sender or the destination is compromised, because IPsec cannot protect against this scenario.

$$\frac{a \in AS \quad C(a) \quad i \in IP \quad j \in IP \quad i \xrightarrow{orig} a}{I^{\mathrm{R}}(i,j)} \tag{B.8}$$

`Intuition:` If the sender AS is under the control of the attacker, then **all** the routes originating from this AS are deemed to be insecure. This scenario describes the situation where a country or a provider implements surveillance over its population. Note that the case in which an endpoint is compromised is captured by (B.2), which would mark the respective IP and thus domain or name server compromised.

Route to Web Server Compromise

$$\frac{\begin{array}{c} e \in Country \quad d \in D \quad a \in AS \quad i,j \in IP \\ d \xrightarrow{\mathrm{A}} j \quad i \xrightarrow{orig} a \quad a \xrightarrow{loc} e \quad I^{\mathrm{R}}(i,j) \\ \neg(HTTPS(d) \wedge \neg I^{\mathrm{CA}}(d) \wedge Redirect(d) \wedge HSTS(d)) \end{array}}{C^{\mathrm{web}}(e,d)} \tag{B.9}$$

`Intuition:` If a route between a client and a web server is insecure, assuming the worst scenario in which a *non-tech-savvy* user accesses the web server via HTTP (at the time of writing HTTP is the default protocol used by browsers if a protocol is not explicitly defined), then the attacker can implement a MITM attack in the following cases:

- *Case 1:* If the web server does not implement HTTPS, then the attacker can eavesdrop and replace the content retrieved from the web server;
- *Case 2:* If the web server implements HTTPS but it does not redirect to HTTPS, then, for the hypothesis previously presented, the attacker can eavesdrop and replace the content retrieved from the web server. The HSTS header does not provide any protection if Redirect is not implemented; indeed the header is ignored in an HTTP connection [154];

- *Case 3:* If the web server implements HTTPS and redirects HTTP traffic to HTTPS but it does not implement HSTS, then the attacker can compromise the connection before the redirection phase;
- *Case 4:* If the web server implements HTTPS but it is vulnerable to certificate authority attacks[2], then a malicious CA can forge digital certificates for the domain and use them to authenticate connections to malicious web servers.

In *Case 3* we ignore the use of a permanent redirection and we require, in addition, the presence of the `strict-transport-security` header. This choice is because the redirection is not a secure mitigation and the HSTS provides a better security compared to the Permanent redirection:

- HSTS covers the entire domain;
- HSTS implements a preloaded list[3];

For those domains that are not in the preloaded HSTS list, the first access to a web server is still insecure even if all the previous requirements are met.[4] For a first approximation, we assumed the attacker to not be allowed to exploit this vulnerable window.

The *postcondition* declares that all the connections originating from the country where the sender AS is located, are compromised. This is an upper-bound assumption because there could exist ASes in the country that do not present an insecure route. This simplification is due to the fact that there is no information about the location within the country of the client contacting the web server.

Route to Name Server Compromise

$$\frac{\begin{array}{c} a{\in}AS \quad i,j{\in}IP \quad e{\in}Country \quad a\xrightarrow{loc}e \quad i\xrightarrow{orig}a \\ f\xrightarrow{DNS}d \quad f\xrightarrow{A}j \quad I^{\mathrm{R}}(i,j) \quad \neg DNSSEC(f) \end{array}}{I^{\mathrm{DNS}}(d,e)} \tag{B.10}$$

`Intuition:` If a route between a client and a name server is insecure and the NS does not implement the DNSSEC protocol, then the attacker can redirect the client to a malicious NS or can implement a DNS cache poisoning attack. Thus, the DNS resolution of the domain is compromised for all connections originating in the country where the client AS

[2]See rules: B.16, B.17, B.18

[3]It is a list of domains that are automatically configured with HSTS. This list is integrated into the browser.

[4]A possible mitigation for this scenario is to increase the number of domains contained in the preloaded HSTS list.

is located[5].

From DNS to Domain Compromise

$$\frac{I^{\text{DNS}}(d) \qquad \neg(HTTPS(d) \wedge \neg I^{\text{CA}}(d) \wedge Redirect(d) \wedge HSTS(d))}{C^{\text{web}}(d)} \tag{B.11}$$

`Intuition:` If a web server has a *Globally compromised DNS*[6] and either it does not fulfill all the conditions to establish a secure connection or the attacker is able to forge a malicious certificate for the website, then the attacker can redirect **all** the clients to a malicious web server that can claim to be the legitimate one.

$$\frac{\begin{array}{c} I^{\text{DNS}}(d,e) \qquad e \in Country \\ \neg(HTTPS(d) \wedge \neg I^{\text{CA}}(d) \wedge Redirect(d) \wedge HSTS(d)) \end{array}}{C^{\text{web}}(e,d)} \tag{B.12}$$

`Intuition:` The same situation applies in case the web server has a *Country compromised DNS*; the only difference lies in the *post condition*, where the attacker can only redirect the clients from the particular country to a malicious web server.

$$\frac{\begin{array}{c} I^{\text{DNS}}(c) \qquad c \xrightarrow{JS} d \qquad \neg SRI(d,c) \\ \neg(HTTPS(d) \wedge Redirect(d)) \vee I^{\text{CA}}(c) \\ \neg l\_HTTPS(d,c) \vee I^{\text{CA}}(c) \qquad \neg UpgradeRequests(d) \vee I^{\text{CA}}(c) \end{array}}{C^{\text{web}}(d)} \tag{B.13}$$

`Intuition:` If a CDN, that provides JS resources for a certain web server, has a *Globally compromised DNS*, then the attacker can redirect the client to a CDN that provides malicious JS resources. This scenario is possible if **all** these conditions are met:

- *The web server does not implement SRI*, thus the JS resource can be replaced with a malicious one.
- *The protocol to retrieve the resource from c is not HTTPS or the attacker is able to forge a certificate for the CDN.*
- *The web server does not implement the `upgrade-insecure-requests` field in the CSP or the attacker is able to forge a certificate for the CDN.*
- *The website is not accessible via HTTPS or does not redirect automatically to the secure protocol or the attacker is able to forge a certificate for the CDN*

[5]This is the same simplification presented in rule B.9

[6]This means that the attacker has control over the content provided by one of the authoritative NSs for this domain.

Note that the website on d is required to implement a redirect to HTTPS only if it uses protocol-relative URLs. In case d does deliver its content via HTTP and includes resources explicitly via HTTPS it is able to protect against this attack on the resolution of c.

$$\frac{\begin{array}{c} I^{\text{DNS}}(c,e) \quad e\in Country \quad c\xrightarrow{JS}d \quad \neg SRI(d,c) \\ \neg(HTTPS(d)\wedge Redirect(d))\vee I^{\text{CA}}(c) \\ \neg l\_HTTPS(d,c)\vee I^{\text{CA}}(c) \quad \neg UpgradeRequests(d)\vee I^{\text{CA}}(c) \end{array}}{C^{\text{web}}(e,d)} \tag{B.14}$$

`Intuition:` The same situation applies in case the CDN has a *Country compromised DNS*; the *post condition* presents the same structure of rule B.12.

Inline JS Injection

$$\frac{\begin{array}{c} c\in Country \quad i,j\in IP \quad d_1,d_2\in D \quad d_2\xrightarrow{JS}d_1 \quad a\in AS \\ i\xrightarrow{orig}a \quad d_2\xrightarrow{A}j \quad I^{\text{R}}(i,j) \quad a\xrightarrow{loc}c \quad \neg SRI(d_1,d_2) \\ \neg(HTTPS(d_1)\wedge Redirect(d_1))\vee I^{\text{CA}}(d_2) \\ \neg l\_HTTPS(d_1,d_2)\vee I^{\text{CA}}(d_2) \quad \neg UpgradeRequests(d_1)\vee I^{\text{CA}}(d_2) \end{array}}{C^{\text{web}}(c,d_1)} \tag{B.15}$$

`Intuition:` If the route from a client to a CDN, that provides JS resources to a web server, is insecure[7] and **all** these conditions are met:

- *The web server does not implement SRI:* in this case a MITM attacker can drop the legitimate JS resource and can replace the content with malicious code.
- *The protocol used to retrieve the resources of the CDN in the web server HTML code of d_1 is not HTTPS or the attacker can forge a malicious certificate for the CDN.*
- *The web server does not implement the* **`upgrade-insecure-requests`** *field in the CSP or the attacker can forge certificates for the CDN.*
- *The website is not accessible via HTTPS or does not redirect automatically to the secure protocol or the attacker can forge a certificate for the CDN*

Then, the attacker can intercept the JS requests and inject malicious JS code.

Note that we required that any mitigation (*UpgradeRequests* or *l_HTTPS*) does not break the functionality of the website by requiring that the resource is available over HTTPS (see Appendix B.0.2).

[7]This model assumes that the web server does not implement a proxy to retrieve the resources from the CDN on behalf of clients.

Certificate Compromise

$$\frac{\begin{array}{c} a{\in}CA \quad d{\in}D \quad e{\in}NS \quad C(a) \\ e\xrightarrow{DNS}d \quad \neg CT(d) \quad \neg DANE(e) \end{array}}{I^{\text{CA}}(d)} \tag{B.16}$$

`Intuition:` If a certificate authority is under the control of the attacker and these conditions are met:

- *The web server's digital certificates are signed by CAs that are not compliant with the Certificate Transparency project.*
- *The authoritative NSs of the domain do not implement the DANE protocol.*

Then, the attacker can forge malicious digital certificates for the domain and use them to generate authenticated connections to malicious web servers.

$$\frac{\begin{array}{c} a{\in}CA \quad d{\in}D \quad e{\in}NS \quad C(a) \\ e\xrightarrow{DNS}d \quad \neg CT(d) \quad C(e) \end{array}}{I^{\text{CA}}(d)} \tag{B.17}$$

`Intuition:` If, in the same scenario of rule B.16, one of the NS is under the control of the attacker, the DANE protocol cannot be trusted. For example, the attacker can modify the TLSA records and insert a new hash of a digital certificate signed by the compromised CA.

$$\frac{\begin{array}{c} a{\in}CA \quad d{\in}D \quad e{\in}NS \quad C(a) \quad e\xrightarrow{DNS}d \\ \neg C(e) \quad (TLSA\_0(d,a) \vee TLSA\_2(d,a)) \end{array}}{I^{\text{CA}}(d)} \tag{B.18}$$

`Intuition:` If, in the same scenario of rule B.16, the authoritative NS is not compromised and implements the DANE protocol, the attacker can forge new digital certificates if one of these two conditions is met:

- *The TLSA certificate usage field is 0 and the compromised CA is in the Certificate Chain*[8]
- *The TLSA certificate usage field is 2 and the compromised CA is in the Certificate Chain from the Server certificate to the Trust anchor.*

Third-party JS Injection

$$\frac{d, e \in D \quad e \xrightarrow{\text{JS}} d \quad \neg SRI(d, e) \quad C(e)}{C^{\text{web}}(d)} \tag{B.19}$$

[8]The model assumes that the TLSA record defines the entire chain; this is the most secure approach.

`Intuition:` If a web server contains a JS resource that is not protected via Subresource Integrity and is hosted in a domain under the control of the attacker, then the attacker can modify the content of the JS script with malicious code.

Access compromised to website access compromised

$$\frac{d \in D \quad C(d)}{C^{\text{web}}(d)} \tag{B.20}$$

`Intuition:` If a domain d is globally compromised, the website on d is compromised as well.

B.0.2 Defender rules

This section describes the propagation rules for the defender used in the threat model. We describe only those rules that require some preconditions to be implemented. The remaining mitigations have no preconditions.

Secure Inclusions

$$\frac{d, c \in D \quad c \xrightarrow{JS} d \quad avail\_over\_HTTPS(c, d)}{l\_HTTPS(d, c)} \tag{B.21}$$

`Intuition:` If a web server contains JS resources from a different domain which are **all** available over HTTPS, then the defender can explicitly enforce HTTPS for retrieving the JS resource in the source code. We do not allow new domains to use protocol-relative URLs (l_HTTPS_compat) because it is an anti-pattern. and if resources are available over HTTPS they can always be retrieved explicitly over HTTPS even if the domain d is using HTTP.

$$\frac{d, c \in D \quad \bigwedge_{c.c \xrightarrow{JS} d} avail\_over\_HTTPS(c, d)}{UpgradeRequests(d)} \tag{B.22}$$

`Intuition:` If the entry *UpgradeRequests* of the CSP is utilized, we need to check that all the JS resources retrieved from all the different domains c are retrievable over HTTPS.

Redirection to HTTPS and HSTS

$$\frac{\begin{array}{c} d,c \in D \quad HTTPS(d) \\ (\bigwedge_{c.c \xrightarrow{JS} d} (l\_HTTPS(d,c) \vee \\ (l\_HTTPS\_compat(d,c) \wedge avail\_over\_HTTPS(c,d)))) \\ \vee UpgradeRequests(d) \end{array}}{Redirect(d)} \tag{B.23}$$

`Intuition:` To implement a redirection over HTTPS to the domain d, it must implement HTTPS and all the included JS resources from external domains must be retrieved over HTTPS (either explicitly or using protocol-relative URLs). This is required to not break a functionality of the domain d (due to mixed-content).

Indeed, if a redirection over HTTPS is established, but the JS resources are not retrievable over HTTPS, it will trigger a mixed-content warning in all the major browsers. In case the domain employs protocol-relative URLs (l_HTTPS_compat), it is also required to have the resource available over HTTPS. The predicates obtained from the rules B.21 and B.22 already required the availability of the resources over HTTPS.

$$\frac{d \in D \quad HTTPS(d)}{HSTS(d)} \tag{B.24}$$

`Intuition:` The precondition to implement the security header *HSTS* is the presence of HTTPS on the domain.

DNSSEC

$$\frac{e, p \in NS \quad \bigwedge_{p.p \xrightarrow{parent\_zone} e} DNSSEC(p)}{DNSSEC(e)} \tag{B.25}$$

`Intuition:` The precondition to deploy *DNSSEC* on a name server is the implementation of DNSSEC in all the parent zones.

DANE

$$\frac{e \in NS \quad DNSSEC(e)}{DANE(e)} \tag{B.26}$$

`Intuition:` The precondition to deploy *DANE* on a name server is the implementation of DNSSEC.

Certificate Transparency

$$\frac{d \in D \quad HTTPS(d)}{CT(d)} \tag{B.27}$$

The precondition to employ Certificate Transparency logs is that the domain implements HTTPS, i.e., it has a digital certificate.

Appendix C

Replication Guide for SOC Experiment

C.0.1 Mapping Attack Scenarios to Task Complexity

We present how to instantiate the two attack scenarios using the theory of SOC task complexity and Table C.1 provides additional information about the causal relations of the sub-tasks and their output that concur at the task complexity.

MIRAI Attack Scenario

Identification Victim An alert referring to a connection to a MIRAI server. The *alert body*$_1$ analysis is composed of a single act with as external input the malware domain name.

Identification Attacker Alerts refer to scan attempts against the network. The *alert body*$_2$ analysis is composed of a single act with as external input the scan attempt.

Identification Reconnaissance Starting from the victim IP, alerts refer to multiple port scan probes sent to the victim IP. The *alert body*$_3$ analysis is composed of a single act with as external inputs the scan technique and the multiple attempts.

Identification Vulnerability Starting from the victim/attacker, logs in Kibana show attempts of connection to the SSH port that end in a successful connection. The *alert body*$_4$ analysis is composed by three acts: first *investigate* the connections from/to the attacker/victim IP via Kibana, then from this connection *identify* the port contacted, and finally *analyze* the connections status. The external inputs are the use of Kibana and the number of attempts. Additional inputs are the port number and the connection status produced by the previous acts.

Identification Malware Delivery Starting from the victim IP, an alert referring to the request of content from an unusual external IP. The *alert body*$_5$ analysis is composed by two acts: *find* the resource requested ((*/apex*)) and then *inspect* the resource content (a shellcode). The resource requested is an external input, while the resource content is produced by the previous act.

Identification Exfiltration Starting from the victim IP, an alert referring to the connection to a C&C. The *alert body*$_6$ analysis is composed of a single act that inspects the content of the transmitted data. The external input in the alert is the credentials submitted to the external IP.

EXIM Attack Scenario

Identification Victim An alert referring to a RCE. The *alert body*$_1$ analysis is composed of a single act with as external input the RCE info.

Identification Attacker An alert referring to a RCE. The *alert body*$_2$ analysis is composed of a single act with as external input the RCE info.

Identification Reconnaissance Starting from the victim IP, logs show scans to different victim's ports where the victim IP replied with a `RST/ACK` (port closed). Finally, the attacker finds the SMTP port open. The *alert body*$_3$ analysis is composed by three acts: first *investigate* the connections towards the victim IP via Kibana, then *identify* that multiple ports are contacted, and *analyze* the reply (`RST/ACK`). The external input is the use of Kibana, while the attempted connections to multiple ports and the `RST/ACK` packets sent as a response are input produced by the previous acts.

Identification Vulnerability Starting from the victim or attacker IP, an alert referring to the exploitation of a RCE. The *alert body*$_4$ analysis is composed of a single act with as external input the RCE info.

Identification Malware Delivery Starting from the victim IP, logs refer to the request for content from an external IP. The *alert body*$_5$ analysis is composed by three acts: first *investigate* the connections established by victim IP via Kibana, then *identify* a connection to an external IP, and *resolve* the IP to the suspicious domain *exploit-db.com.* The external input is the use of Kibana, while the establishment of a SSL connection and the domain name contacted are inputs produced by the previous acts.

Identification Exfiltration Starting from the victim IP, logs refer to the connection to an external IP. The *alert body*$_5$ analysis is composed by three acts: first *investigate* the external connections generated from the victim IP via Kibana, then *identify* a connection to an external IP, and *resolve* the IP to the suspicious domain *l6asd8cs-google-support.abc.xn-p2a.jetzt*. The external input is the use of Kibana, while the establishment of a SSL connection and the domain name contacted are inputs produced by the previous acts.

Table C.1: Causal relations and output sub-tasks

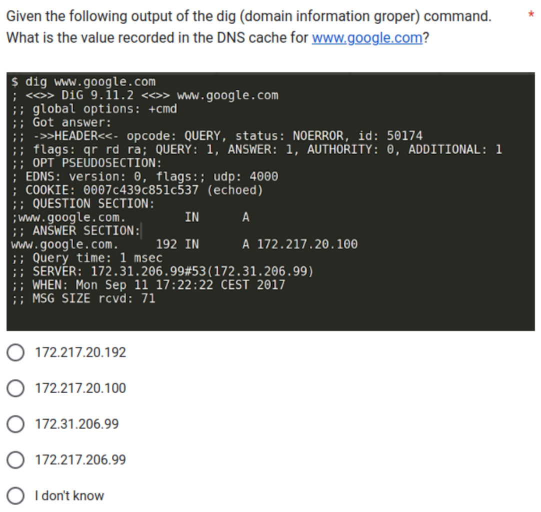
Sub-task	Causal Relation	Rationale	Output
`Identif. Victim`	/	To determine the victim the position is crucial because, for example, the attacker does not exfiltrate its credential to the victim but only vice versa	IP address of the victim, the type of system infected, its department, etc.
`Identif. Attacker`	/	To determine the attacker the position is crucial because, for example, the victim does not send an exploit to the attacker but only vice versa	IP address of the attacker, its geolocation, etc.
`Identif. Recon`	`Identif. Victim` OR `Identif. Attacker`	The analyst needs first to identify the victim/attacker IP and then can proceed to identify the alert as part of the reconnaissance phase of the attack. In the case of multiple probes, to identify a recon phase the order of probed systems or ports does not matter thus there is no causal relation. In this case, information cue related to the attacker/victim IP and its position in the n alerts do not change	Scan technique, the timing, etc.
`Identif. Vuln.`	`Identif. Victim` OR `Identif. Attacker`	The analyst first needs to identify the victim or the attacker to determine the vulnerability exploited. In case the type of vulnerability requires multiple payloads the victim and attacker IPs, as well as their position, do not change and do not have a causal relation for the analysis. The information remains the same as in the one-shot payload	Type of vulnerability exploited (SQLi, RCE, etc.), the software affected, etc.
`Identif. Malware Delivery`	`Identif. Victim` OR `Identif. Attacker`	To determine the infrastructure delivering malware additional inputs are required: the external IP contacted, the IP of the victim/attacker, and the position as source and target in the alert. Indeed, the position is a critical information as the victim receives malware from external IP and not vice versa. Instead of the victim IP, an analyst can also employ a previously identified attacker IP as an additional source, for example by determining that both the attacker IP and the malware server IP are from the same subnet	IP address of the server contacted, the malware family downloaded, etc.
`Identif. Exfiltr.`	`Identif. Victim` OR `Identif. Attacker`	To determine the victim the position is crucial because, for example, the attacker does not exfiltrate its credential to the victim but only vice versa	IP of the C&C, exfiltrated information, geolocation of the C&C, etc.

C.0.2 Background and Skills questionnaire

Table C.2 shows an example of the instantiation of the background level with a concrete example and the associated numerical value used to present the results. Table C.3 shows the entire list of skills questions by category while Figure C.1a and Figure C.1b show two examples of the technical questions to assess students' skill levels related to networking and vulnerability respectively.

Table C.2: Background question on Hacking

Options	Value
"I have attended a couple of lecture on Hacking or security in a course or on some website"	1
"I have attended a course/tutorial on buffer overflow or software testing"	2
"I have used a fuzzer or vulnerability scanner in hands-on labs or during short internships"	3
"I have used vulnerability & network scanners during my professional experiences (>3 months)"	4
"I am a professional pen tester (e.g. SANS certified) in my work (>3 years work) or have an equivalent professional certification"	5

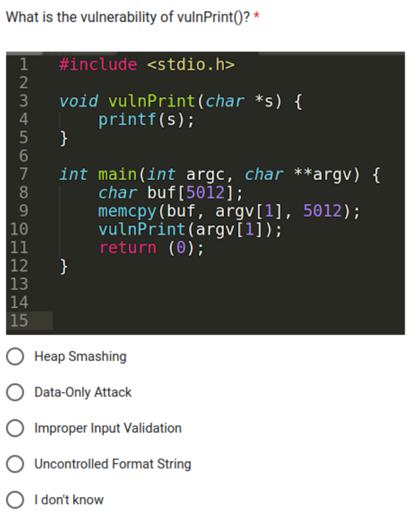


Given the following output of the dig (domain information groper) command. *
What is the value recorded in the DNS cache for www.google.com?

```
$ dig www.google.com
; <<>> DiG 9.11.2 <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50174
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
; COOKIE: 0007c439c851c537 (echoed)
;; QUESTION SECTION:
;www.google.com.         IN      A
;; ANSWER SECTION:
www.google.com.     192 IN      A 172.217.20.100
;; Query time: 1 msec
;; SERVER: 172.31.206.99#53(172.31.206.99)
;; WHEN: Mon Sep 11 17:22:22 CEST 2017
;; MSG SIZE rcvd: 71
```

- ○ 172.217.20.192
- ○ 172.217.20.100
- ○ 172.31.206.99
- ○ 172.217.206.99
- ○ I don't know

(a) Networking question in the technical assessment related to gathering DNS information



What is the vulnerability of vulnPrint()? *

```c
#include <stdio.h>

void vulnPrint(char *s) {
    printf(s);
}

int main(int argc, char **argv) {
    char buf[5012];
    memcpy(buf, argv[1], 5012);
    vulnPrint(argv[1]);
    return (0);
}
```

- ○ Heap Smashing
- ○ Data-Only Attack
- ○ Improper Input Validation
- ○ Uncontrolled Format String
- ○ I don't know

(b) Vulnerability question in the technical assessment related to software vulnerability identification

Figure C.1: Example of technical questions to assess student skill levels

Table C.3: Skills questions by category

Category	Question
Networking	*"Background: Hosts on the Internet keep track of the common names (like website.org) and the IP addresses (like 127.88.50.109) using a database called the Domain Name System. Hosts typically check a local hosts file first (files like /etc/hosts or %systemroot%system32driversetchosts). If there is no entry in that file, then a host will request the address of a server that it wishes to contact, say www.google.com, and the DNS server the host is configured to use will return the IP Address so the connection can begin. Host A on your network performs a DNS lookup of website.org and gets the result 127.98.101.45 Host B on your network performs a DNS lookup of website.org and gets the result 127.129.58.111 Which of the following could explain why?"*
Networking	*"The client on either side of a TCP session maintains a 32-bit sequence number it uses to keep track of how much data it has sent. This sequence number is included on each transmitted packet, and acknowledged by the opposite host as an acknowledgement number to inform the sending host that the transmitted data was received successfully. Consider the following lists of sequence and acknowledgement numbers, where packet 2 is delayed and arrives out of sequence. Complete the sequence with the missing values"*
Networking	*"Given the following output of the dig (domain information groper) command. What is the value recorded in the DNS cache for www.google.com?"*
Networking	*"What is the port through which the server replied to the following wget request?"*
Networking	*"What is the problem shown by the following output of tcpdump?"*
Vulnerability	*"What is the vulnerability of vulnPrint()?"*
Web Progr.	*"Where is this code located to handle HTTP request?"*
Web Progr.	*"When programming sockets for a server in C, what is the correct order of the needed functions?"*
Net. Analysis	*"Write the source and the destination ports, basing on the following output of tcpdump"*
Net. Analysis	*"What is the option to obtain the epoch time in the output of tcpdump?"*

C.0.3 Manual Scoring procedure

Table C.4 and Table C.5 shows the instantiation of the guidelines for the scoring of the tickets for the two scenarios depicted in Figure 5.2.

Table C.4: Guidelines scoring tickets phases MIRAI

Score	`Identification Reconnaissance`	`Identification Vulnerability`	`Identification Malware Delivery`	`Identification Exfiltration`
OK	(Ticket reported **only** (199.19.215.23 (and possibly 199.19.215.29, 91.198.120.42) in "Attacker IP" field or in the explanation) ∨ (121.145.34.116 in the "Victim IP" field or in the explanation)) ∧ (*Port Scan/SSH Scan/O.S. Scan* in "reconnaissance" action or in the explanation)	(Ticket reported **only** (199.19.215.23 (and possibly 199.19.215.29, 91.198.120.42) in "Attacker IP" field or in the explanation) ∨ (121.145.34.116 in "Victim IP" or in the explanation)) ∧ (*Weak Password* in "vulnerability" action or in the explanation)	Ticket reported **only** 91.198.120.42 or p.pi.fi (and possibly 199.19.215.29, 199.19.215.23) in "http requests" action or in the explanation	Ticket reported **only** 199.19.215.29 (and possibly 199.19.215.23, 91.198.120.42) in the "data exfiltration" action or in the explanation
NO	(Ticket did not report (199.19.215.23 neither in "Attacker IP" field nor in the explanation) ∧ (121.145.34.116 neither in "Victim IP" field nor in the explanation)) ∨ (*Port Scan/SSH Scan/O.S. Scan* not in "reconnaissance" action nor in the explanation)	(Ticket did not report (199.19.215.23 neither in "Attacker IP" field nor in the explanation) ∧ (121.145.34.116 neither in "Victim IP" field nor in the explanation)) ∨ (*Weak Password* not in "vulnerability action" nor in the explanation)	Ticket did not report (91.198.120.42 or p.pi.fi neither in http requests action nor in "Attacker IP" field nor in the explanation) ∧ (121.145.34.116 not in "http requests" action)	Ticket did not report (199.19.215.29 neither in data exfiltration action nor in "Attacker IP" field nor in the explanation) ∧ (121.145.34.116 not in "data exfiltration" action)
MAY	Everything else	Everything else	Everything else	Everything else

Table C.5: Guidelines scoring tickets phases EXIM

Score	`Identification Reconnaissance`	`Identification Vulnerability`	`Identification Malware Delivery`	`Identification Exfiltration`
OK	(Ticket reported **only** (54.37.60.203 (and possibly 31.220.56.38, 192.124.249.8, and 46.38.239.190) in "Attacker IP" field or in the explanation) $\vee$ (121.145.34.27 in the "Victim IP" field or in the explanation)) $\wedge$ (*Port Scan/SSH Scan/O.S. Scan* in "reconnaissance" action or in the explanation)	(Ticket reported **only** (31.220.56.38 (and possibly 54.37.60.203, 192.124.249.8, and 46.38.239.190) in "Attacker IP" field or in the explanation) $\vee$ (121.145.34.27 in "Victim IP" or in the explanation)) $\wedge$ (*Remote Code Execution* in "vulnerability" action or in the explanation)	Ticket reported **only** 192.124.249.8 or exploit-db (and possibly 31.220.56.38, 54.37.60.203, and 46.38.239.190) in "http requests action" or in the explanation	Ticket reported **only** 46.38.239.190 or l6asd8cs-google-support.abc.xn-p2a.jetzt (and possibly 31.220.56.38, 54.37.60.203, and 192.124.249.8) in the "data exfiltration" action or in the explanation
NO	(Ticket did not report (54.37.60.203 neither in "Attacker IP" field nor in the explanation) $\wedge$ (121.145.34.27 neither in "Victim IP" field nor in the explanation)) $\vee$ (*Port Scan/SSH Scan/O.S. Scan* not in "reconnaissance action" nor in the explanation)	(Ticket did not report (31.220.56.38 neither in "Attacker IP" field nor in the explanation) $\wedge$ (121.145.34.27 neither in "Victim IP" field nor in the explanation)) $\vee$ (*Remote Code Execution* not in "vulnerability action" nor in the explanation)	Ticket did not report (192.124.249.8 or exploit-db neither in "http requests" action nor in "Attacker IP" field nor in the explanation) $\wedge$ (121.145.34.27 not in "http requests" action)	Ticket did not report (46.38.239.190 or l6asd8cs-google-support.abc.xn-p2a.jetzt neither in "data exfiltration" action nor in "Attacker IP" field nor in the explanation) $\wedge$ (121.145.34.27 not in "data exfiltration" action)
MAY	Everything else	Everything else	Everything else	Everything else