*Risk Analysis, Vol. 0, No. 0, 2021* DOI: 10.1111/risa.13732

# The Work-Averse Cyberattacker Model: Theory and Evidence from Two Million Attack Signatures

**Luca Allodi**,[1,*] **Fabio Massacci**,[2,3] **and Julian Williams**[4]

The assumption that a cyberattacker will potentially exploit all present vulnerabilities drives most modern cyber risk management practices and the corresponding security investments. We propose a new attacker model, based on dynamic optimization, where we demonstrate that large, initial, fixed costs of exploit development induce attackers to delay implementation and deployment of exploits of vulnerabilities. The theoretical model predicts that mass attackers will preferably (i) exploit only one vulnerability per software version, (ii) largely include only vulnerabilities requiring low attack complexity, and (iii) be slow at trying to weaponize new vulnerabilities . These predictions are empirically validated on a large data set of observed massed attacks launched against a large collection of information systems. Findings in this article allow cyber risk managers to better concentrate their efforts for vulnerability management, and set a new theoretical and empirical basis for further research defining attacker (offensive) processes.

**KEY WORDS:** Cyber security; hackers model; risk management; update costs

## 1. INTRODUCTION

A natural starting point for an evidence, big-data based cyber-risk model is to look at "attacks in the wild": Each attempt to attack a system using a vulnerability and an exploit mechanism generates a specific *attack signature*, which may be recorded by software security vendors and can be identified by security researchers (Bilge & Dumitras, 2012) and linked to vulnerabilities that attackers seek to exploit (Allodi & Massacci, 2014).

[1]Technical University of Eindhoven, Groene Loper 5, Eindhoven, The Netherlands.
[2]University of Trento, Via Sommarive 9, Povo (Trento), Italy.
[3]Vrije Universiteit Amsterdam, De Boelelaan 1111, Amsterdam, The Netherlands.
[4]Durham University Business School, Mill Hill Lane, Durham, UK.
*Address correspondence to Luca Allodi, Technical University of Eindhoven, TU/e Science Park, Building "MetaForum", Groene Loper 5, Office MF6.122. NL-5612 AE, Eindhoven, The Netherlands; l.allodi@tue.nl.

For example, attackers focusing on chip and pin credit cards, which require physical access, are proactive and rapidly update their small menu of exploits (Murdoch, Drimer, Anderson, & Bond, 2010). In contrast, attackers on the web seem to be wary of exploiting the full range of vulnerabilities available to them: The actual risk of attacks in the wild is limited to 100 vulnerabilities out of the 50,000 reported in vulnerability databases (Allodi & Massacci, 2014; Nayak, Marino, Efstathopoulos, & Dumitraş, 2014). Even untimely disclosures do not seem to increase attack volumes (Mitra & Ransbotham, 2015).

This empirical evidence of web attacker behavior is at odds with the attacker models that underpin most cyber-risk models: A system should be secured "against arbitrary behavior of the saboteur" (Dolev & Yao, 1983). Variants of the all-powerful attacker model exist (e.g., honest-but-curious, game-based models) but they only changed the power and speed of attacks not the will: If there is a vulnerability that the attacker can exploit, she will eventually do it.

As a result, current cyber-risk standards (e.g., US NIST-800, UK IAS) provide advice based on vulnerability severity: All severe vulnerabilities present in a system should be addressed ((e.g., the comments by Schneier, 2008, that cover similar ground). Indeed, papers on web security report the persistence of vulnerabilities on internet sites as evidence for risk underestimation by website owners (Nikiforakis et al., 2014; Stock, Lekies, & Johns, 2013). While sound in the presence of limited information, this advice often yields disproportionate mitigations, which only address threat inflation by security vendors (Brito & Watkins, 2011). Big data on attacks may allow us to use more accurate models of attackers and empirically validate them. Such models will then provide a better cyber-risk assessment strategy for defenders.

The key contribution of this article is a novel theoretical model of the dynamic decisions of the attacker based on Stokey's logic of inaction (Stokey, 2008). We call attackers "*work averse*" to capture the natural assumption that attackers will not engineer and adopt new, complex exploits if they can obtain a satisfactory result with what they already have. If this assumption empirically holds, attackers will flock to the set of low-complexity vulnerabilities with high impact, and postpone the adoption of new exploits until the previous ones become ineffective (e.g., as most vulnerable systems get patched). The proposed model steers away from classical, and empirically disproved (e.g., Allodi, 2015; Nayak et al., 2014) assumptions on the production function of new cyber-attacks, and reconciles these empirical observations with a novel model describing the arrival process of new attacks at scale. Our model has profound implications for practical cyber-risk management; for example, our model indicates that once an attack for a vulnerability in a software is deployed at scale, the remaining vulnerabilities for the same software will—at large—be left untouched by mass attackers. Defenders can then concentrate efforts in different parts of the system (e.g., a different software component). We enucleated several empirical hypotheses of attacker behavior that are direct consequences of this model, with a direct impact on the corresponding risk management process by defenders.

Mathematically, we model the timing of effort by the attacker as a dynamic programming problem and then, for the purpose of empirical analysis, restrict it to an attacker focusing on the "next" update of their exploit portfolio (Section 3). To evaluate empirically the time delays in between these exploit updates we derive, directly from the theoretical model, a regression model of equilibrium update times (Section 8) regressing over vulnerability and attacked system characteristics. We then use results from the regression model to test several empirical hypotheses for the regression variables emerging naturally from the theoretical model (summary in Table V). To empirically validate our model, we leverage on big data analysis and the Worldwide Intelligence Network Environment (WINE) data set (Dumitras & Shou, 2011) spanning two million attack signatures recorded in the wild by Symantec, a large security firm (Section 6). In the empirical study, we control for several factors related to the characteristics of the user and their system (e.g., user geographical locations). We discuss the results of the empirical analysis (Section 7) and conclude the article by outlining implications for theory and practice (Section 10).

## 2. BACKGROUND

The risk analysis literature has considered the need for data-driven cyber-risk models numerous times (Allodi & Massacci, 2017; Rao et al., 2016). A significant obstacle of cyber-risk models is the lack of an empirically tested *attack production model* that describes the attacker decision process (Cox Jr, 2008). A major difficulty is that attackers are very diverse and do not have any sort of centralized decision-making process. When characterizing this process, a crucial differentiation must however be made between "Mass-Attackers" (who focus on high-volume of possibly low-value targets) and "Advanced Persistent Threats" (or APTs for short) generated by highly specialized groups that target specifically few high-value targets. Targeted cyber-attacks are characterized by a strong "adversarial" connotation (Rios Insua, Rios, & Banks, 2009) where the attacker can (and has the resources to) perform sophisticated reconnaissance of the target(s), identify suitable *0-day* attacks, and tailor the whole attack process against the specific target (Paté-Cornell, 2012). Yet, these attackers only make up for a small fraction of the attack space (Bilge & Dumitras, 2012; Research, 2018).

By contrast, mass-scale attacks focus on (known) vulnerabilities that remain long unpatched (Nappa, Johnson, Bilge, Caballero, & Dumitras, 2015; Research, 2018) and are concentrated among a few only of the several thousand vulnerabilities available to

attackers (Allodi & Massacci, 2014; Research, 2018). Unfortunately, timely patching of all vulnerabilities is infeasible (Kotzias, Bilge, Vervier, & Caballero, 2019) due to the high costs associated with the patching decision (Verizon, 2011). This effectively makes the problem of identifying which vulnerabilities are (or are going to be) high risk a highly practical problem to solve.

To help defenders make more informed decisions, we need a *characteristic model* of the mass attackers that does not depend on idiosyncratic characteristics of the attacker (e.g., their origin or motive). Since exploit engineering is an expertise-intensive, time-consuming process, and the mass attackers do not generate APTs, a realistic model of the mass attacker is incompatible with models whereby the attacker can exploit any vulnerability at will. Without reconstructing this missing block, it is impossible to build predictive models to infer which vulnerabilities must be fixed immediately, and which can wait.

Defender strategies in terms of patching times have been investigated both empirically (Kotzias et al., 2019; Okhravi & Nicol, 2008) and theoretically (Serra, Jajodia, Pugliese, Rullo, & Subrahmanian, 2015b), assuming specific threat models and attack production functions. For example, software vendors may maximize profit by exploiting attacker behavior (Kannan, Rahman, & Tawarmalani, 2016); system diversification, as opposed to patching, may yield lower costs when compared to ineffective "single-metric" patching policies (Dey, Lahiri, & Zhang, 2015). Similarly, security best practices do not necessarily lead to more robust firm security, and the relation between security and liability may also be affected by different managerial settings (Lee, Geng, & Raghunathan, 2016).

The importance of well-grounded observations for realistic and operational models capable of supporting strategic decision making at the level of a firm or organization is of relevance across several domains, including system resilience (Guikema, McLay, & Lambert, 2015). For example, the balance between recommendation and implementation of rules and regulations aimed at reducing the attack surface of a system is delicate, and must be modulated against the threat: overregulation risks (e.g., by alienating users that are supposed to implement those policies) opening up additional attack paths exploitable by attackers. On the other hand, underregulation is also undesirable to avoid leaving important vulnerabilities open (Gisladottir, Ganin, Keisler, Kepner, & Linkov, 2017). Game theory is a popular tool to investigate these trade-offs, but assumptions behind those model must remain realistic to derive effective "operational" recommendations (Guikema et al., 2015). Most game-theoretic models generally consider the attacker to be potentially capable of adopting any strategy with different degrees of probability, depending on the conditions of the game (see Do et al., 2017, for a survey). For instance, Manshaei, Zhu, Alpcan, Bacşar, and Hubaux (2013) posit a case where attacker strategies can range from fixed attack updates to adaptive strategies based on the defender's decisions (van Dijk, Juels, Oprea, & Rivest, 2013), or on expectations of the attack's *persistence* and *stealthiness* to defender detection and remediation capabilities (Smeets, 2018). Attacker/defender equilibrium forces are further analyzed by Zhang and Zhuang (2019), who study optimal defensive strategies in the presence of adaptive attackers and multiple attack types, for which different probabilities of success and impact on the defended infrastructure lead to different defensive outcomes. Whereas Zhang and Zhuang (2019) do not focus on cyberattackers, the cyber-threat landscape of mass attackers and APT-level attackers poses similar challenges for the strategic allocation of defensive resources. Recent papers have also focused on systemic issues; for example, Kuper, Massacci, Shim, and Williams (2020) develop a static game in which network structure plays a role in the equilibrium actions of attackers. However, the dynamics of adjustments in attacker effort has not yet been explored.

Attacks against large pools of "similar" targets (e.g., by geographical distribution, or system configuration) adapt to the state of the *population* of potential targets (as opposed to one specific target), for which attack technologies developed "ad-hoc" are not always viable (Ransbotham & Mitra, 2009). For an attacker sensitive to the cost of engineering a technical or social exploit, not all attack types make sensible avenues for investment (Herley, 2013). This is supported by empirical evidence showing that attack tools actively used by attackers embed only an handful of exploits (Kotov & Massacci, 2013), and that the vast majority of attacks recorded in the wild are driven by only a small fraction of known vulnerabilities (Allodi, 2015; Nayak et al., 2014). Some reward must be forthcoming, as the level of effort required to implement and deliver the attack observed in the wild is not negligible, as demonstrated by the presence of an underground market

where vulnerability exploits are rented to attackers ("exploitation-as-a-service") as a form of revenue for exploit writers (Grier et al., 2012). In fact, recent studies on different samples (Allodi & Massacci, 2014) have challenged the automatic transfer of the technical assessment of the "exploitability" of a vulnerability into actual attacks against end users: There is a substantial lack of correlation between the observed attack signatures in the wild and metrics (such as the Common Vulnerability Scoring System [CVSS)aintained and employed by the NIST; First.org, 2015) providing an assessment of the vulnerability severity. The current trend in industry is to use metrics such as CVSS as proxies for risk and demanding immediate action (Beattie et al., 2002), but evidence suggests this may be neither sensible, nor effective (Allodi & Massacci, 2014).

The risk analysis literature has also identified the importance of contextual factors to the realization of adversarial risks (Rios Insua et al., 2009; Serra, Jajodia, Pugliese, Rullo, & Subrahmanian, 2015a), particularly when aiming at building probabilistic and quantitative models of attack arrival (Allodi & Massacci, 2017; Brown & Cox Jr, 2011). The current lack of a sound model describing the generation of new cyberattacks is effectively preventing the literature to fully move from a qualitative/semiquantitative risk framework to a quantitative one (Cherdantseva et al., 2016).

## 3. THE WORK-AVERSE ATTACKER

Our model captures the update process of attacks deployed at scale by a large collection of uncoordinated mass attackers. Importantly, mass attackers seldom build their own independent attack technologies, but rather fetch attacks from a shared pool of available attacks (e.g., by acquiring them through the underground economy; Allodi, 2017; Grier et al., 2012; Kotov & Massacci, 2013). The intuition behind our model is that mass attackers with similar objectives (e.g., to install botnet malware on Windows systems in the United States) will have, at large, to move over to the next available attack when those targets, on average, cannot be infected anymore with the old attack technology. Hence, the process with which attackers of the same "type" update their attack portfolio can be captured by a unified model considering all attackers of that type jointly. We model attackers to be risk neutral, and to gain revenue by making both fixed and variable cost investments in attacking a large group of independent target systems. The parameters determining the optimization problem faced by the attacker are presumed to be independent and identically distributed across the collection of attackers.

Each attacker starts their activities at time $t = 0$ by identifying a subset of vulnerabilities $V \subset \mathcal{V}$ from a universe $\mathcal{V}$ affecting a number of target systems $N$. A fraction $\theta_V$ of the $N$ systems is affected by $V$ and would be compromised by an exploit in absence of security countermeasures. Targets deploy patches, update systems, or use new signatures in antivirus or IPSs whose relative effect on attack success has been discussed in extant literature (Chen, Kataria, & Krishnan, 2011; Nappa et al., 2015). Such arrival rate is typically uncorrelated with vulnerability discovery as it depends on external scheduling by software vendors, or testing in companies. For example, a recent empirical study (Kotzias et al., 2019) showed that it may take more than six months to arrive at patching 90% of the vulnerable systems. Denoting the mitigation and patching adoption rate by $\lambda$, we define the number of systems impacted by vulnerabilities in $V$ at time $t$ as

$$N_V(t) = N\theta_V e^{-\lambda t}. \tag{1}$$

To engineer the exploits for the vulnerabilities $V$, the attacker will pay an upfront cost $C(V|\emptyset)$ and has an instantaneous stochastic profit function of

$$\Pi_V(t) = [r(t, N_V(t), V) - c(t, V)]e^{-\delta t}. \tag{2}$$

The function $r(t, N_V, V)$ is a stochastic revenue component that accounts for the probability of establishing contact with a vulnerable system (Franklin, Paxson, Perrig, & Savage, 2007), making a successful infection given a contact (Allodi, Kotov, & Massacci, 2013), and monetizing the infected system (Grier et al., 2012; Rao & Reiley, 2012); the factor $c(t, V)$ is the variable costs of maintaining the attack (payload obfuscation to avoid detection; Kotov & Massacci, 2013) or renew the domain names to prevent domain blacklisting (Stone-Gross et al., 2009), both subject to a discount rate $\delta$. We do not make any assumption on the accounting unit for revenues from successful attacks. For instance, revenues can also be in the form of kudos on hacker forums (Ooi, Kim, Wang, & Hui, 2012), or revenues from trading victim's assets in black markets (Campobasso & Allodi, 2020).

At some point, the attacker might decide to perform a refresh of the attacking capabilities by

**Table I.** Parameters and Variables from the Model

| Parameter Variable | Description |
|---|---|
| $t$ | Continuous time index. |
| $T_i$ | An update time when an attacker updates the vulnerabilities, indexed by $i \in \{0, 1, 2, \ldots\}$. $n$ is the last update (if it is finite). |
| $\mathcal{V}$ | The universe of known vulnerabilities affecting all systems. |
| $N$ | The total number of target machines affected by vulnerabilities in $\mathcal{V}$. |
| $V$ | The subset of vulnerabilities in $\mathcal{V}$ identified by attackers for exploitation. |
| $\theta_V$ | The fraction of $N$ affected by $V \in \mathcal{V}$. |
| $r(t, N_V, V)$ | Revenue function from successful attacks. |
| $c(t, V)$ | Variable cost function for deploying attacks. |
| $C_i$ | Fixed cost of adding at $T_i$ new vulnerabilities to be exploited by the attacker. |
| $\lambda$ | Arrival rate of vulnerability patches to the universe of systems. |
| $\delta$ | Discount rate of the attacker. |
| $\Pi_V(t)$ | Profit function for a given set of vulnerabilities $V$. |

introducing a new vulnerability and engineering its exploit by incurring an upfront cost of $C(v|V)$. This additional vulnerability will produce a possibly larger revenue $r(t, N_{V\cup\{v\}}(t), V \cup \{v\})$ at a marginal cost $c(t, V \cup \{v\})$. As the cost of engineering an exploit is large with respect to maintenance ($C(v|V) \gg c(t, V \cup \{v\})$) and neither successful infection (Allodi et al., 2013) nor revenues are guaranteed (Allodi, Corradin, & Massacci, 2015; Rao & Reiley, 2012), the attacker faces a problem of deciding action versus inaction in the presence of fixed initial costs as described by Stokey (2008). The optimal strategy is to deploy the new exploit only when the old vulnerabilities no longer guarantee a suitable expected profit. This decision problem is then repeated over time for $n$ newly discovered vulnerabilities, and $n$ refresh times denoted by $T_i$. Model parameters are summarized in Table I.

We denote by $C_0 = C(V|\emptyset)$ the initial development cost and by $C_{i+1} \equiv C(v_{i+1}|V \cup \{v_1 \ldots v_i\})$ the cost of developing the new exploits, given the initial set $V$ and the additional vulnerabilities $v_1 \ldots v_i$. We denote by $N_i(t) \equiv N_{V\cup\{v_1,\ldots,v_i\}}(t)$ the number of systems affected by adding the new vulnerability at time $t$. We make no assumption on the particular order over the vulnerabilities $v_i$. We simply assume that there is some sequence in which they are engineered and that sequence will be determined empirically. Similarly, we define $r_i(t)$ and $c_i(t)$ as, respectively, the revenue and the marginal cost of the vulnerability set $V \cup \{v_1, \ldots, v_i\}$. The critical tipping point is when the instant marginal cost is equal to the instant marginal revenue $r_i(T_{i+1}, N_i(T_{i+1})) = c_i(T_{i+1})$, and at this point, the attacker will need to refresh the set of exploited vulnerabilities in order to continue making a profit, thus identifying all action points $T_{i+1} > T_i$. Since the maintenance of malware, for example, through "packing" and obfuscation (i.e., techniques that change the aspect of malware in memory to avoid detection) is minimal does not depend on the particular vulnerability (Brand, Valli, & Woodward, 2010; Kotov & Massacci, 2013), and can be automated in matter of minutes (Castro, Schmitt, & Rodosek, 2019), the maintenance costs are negligible relative to the fixed costs of updating, hence $c_i(t) \ll C_i$, and thus the next interval tends to infinity, $T_{n+1} \to \infty$. A change of technological constraints (e.g., widely deployed detection techniques capable of identifying any variant of the same attack) would require to at least partially revise these assumptions, and therefore the model, in the future.

Empirical evidence indicates the mass attacker faces a decision problem with repeated peak actions with random revenues followed by long periods of quasi-inaction (Allodi, 2015; Nayak et al., 2014). Whereas problems of this type are oftentimes analytically intractable, Stokey (2008) provides a framework offering a series of approximating solutions that can be applied to generic formulations of processes with "quasi-inaction."[1] Accordingly, we assume an history-less payoff with a risk-neutral preference so that expected pay-off and expected utility coincide and risk preferences are then encapsulated in the discount factor. See Stokey (2008) and Birge and Louveaux (2011) for a discussion.

The expected payoff from deployed malware at time $t$ (where $t \geq T$ is the amount of time since the attacker updated the menu of attacks by engineering new exploits at time $T$) is then as follows:

$$r(t, N_{V\cup\{v\}}(t)) = rN\Big(\theta_V e^{-\lambda t} + (\theta_{V\cup\{v\}} - \theta_V)e^{-\lambda(t-T)}\Big). \quad (3)$$

The first term in the parentheses measures the systems' vulnerability to the set $V$ of exploited vulnerabilities that have been already partly patched, while the second term accounts for the new, alterna-

[1] In practice, these approximations allow to pass from the Riemann stochastic integral that emerges from the problem formulation, to a standard Leibniz-style integral so that the resulting optimization problem is more tractable.

tive systems that can now be exploited by adding $v$ to the pool of vulnerabilities being targeted. For the latter systems, the unpatched fraction restarts from one at time $T$. Together the terms deliver the instant expected revenue process across their campaign. The attackers' decision problem is then to establish the timing of when to implement $v$. Indeed, from an empirical observation of malware in the wild, it is clear that technology diffusion is not strictly continuous (Allodi & Massacci, 2013; Bilge & Dumitras, 2012; Nayak et al., 2014).

The attacker faces the problem of choosing a sequence of update times indexed from $n \to \infty$

$$\{T_1^*, \dots, T_n^*\} = \underset{\{T_1,\dots,T_n\}}{\arg\max} \sum_{i=0}^{n} (\Pi(T_{i+1}, T_i) - C_i) e^{-\delta T_i}, \quad (4)$$

$$\begin{aligned} \Pi(T_{i+i}, T_i) &= \int_{T_i}^{T_{i+1}} (r_i(t, N_i(t)) - c_i(t)) e^{-\delta t} dt \\ &= \frac{rN}{\lambda + \delta} \cdot \left(\theta_i - \theta_{i-1} + \theta_{i-1} e^{-\lambda T_i}\right) \cdot \quad (5) \\ &\quad \left(1 - e^{-(\lambda+\delta)(T_{i+1} - T_i)}\right), \end{aligned}$$

where $\theta_{-1} \equiv 0$, $\theta_0 \equiv \theta_V$, and $\theta_i \equiv \theta_{V \cup \{v_1 \dots v_i\}}$.

Note that, from the above formulation, relatively large discount rates lead to an exponential decrease of the impact of update decisions. This is a common observation also in dynamic planning problems (see DeGroot, 2005, for an extended discussion), and provides us with a clear rationale for restricting our attention to cases when $T_1^* > 0$ (since the optimal subsequent update $T_2^*$ is then sufficiently far into the future to not disturb the first update $T_1^*$). Indeed, for cases when $T_1^* > 0$ and $T_2^* \to \infty$, a closed-form solution for the next update is easily obtained by manipulation of the first-order conditions for $T_1^*$ holding $T_2^*$ as constant. Due to the high uncertainty of future vulnerability discoveries and achievable attack reliability (Allodi & Massacci, 2014; Bozorgi, Saul, Savage, & Voelker, 2010), it is reasonable to assume that attackers generally operate under an assumption of sufficiently high discount rates for the above to hold.

**Proposition 1.** *A risk-neutral attacker focusing on the next update with decreasing effectiveness due to patching and antivirus updates, a negligible cost of maintenance for each exploit, and a marginal profit at least equal to the marginal revenue for each machine ($\partial\Pi/\partial T \geq r(0, N_V(0), V)/N_V(0)$) will renew her exploit at time $T^\star$*

$$T^\star = \frac{1}{\delta} \log\left(\frac{C(v|V)}{r} - \frac{\delta}{\lambda + \delta}(\theta_{V\cup\{v\}} - \theta_V)N\right) \quad (6)$$

*under the condition $\frac{C(v|V)}{rN} \geq \frac{1}{N} + \frac{\delta}{\lambda+\delta}(\theta_{V\cup\{v\}} - \theta_V)$.*

This condition provides a lower bound for the trade-off provided by the cost of introducing a new exploit ($C(v|V)$), and the expected revenue across infected systems ($rN$); as the latter decreases as systems get patched, the cost of introducing a new exploit becomes justified and leads to the satisfaction of the condition, and hence to the existence of an optimal update time $T^*$. The proof for Equation (6) is available in the Supporting Information (Allodi, Massacci, & Williams, 2017).

The *"all–powerful" attacker* is still admitted as a particular case when the attacker cost function $\frac{C(v|V)}{r}$ for weaponizing a new vulnerability goes to zero. In this case, Proposition 1 predicts that the attacker could essentially deploy the new exploit at an arbitrary time $[0, +\infty]$ even if the new exploit would not yield a large impact.

## 4. EMPIRICAL MODEL DERIVATION

To empirically evaluate this model, we would need to measure the time $T^\star$ of introduction of new exploits by attackers at scale. This is clearly not possible without the attackers' cooperation. To avoid this identification problem, we use the time in between two consequent attacks $\mathcal{T}$ as a suitable proxy. Fig. 1 reports a pictorial representation of the transformation. Each curve represents the decay in time of number of attacks against two different vulnerabilities. The first attack (blue line) is introduced at $t = 0$, and the second (red line) at $t = T^\star$. The number of received attacks is described by the area below the curve. Let $\mathcal{U}(\Theta_V \cup \{v\}, t, \mathcal{T})$ represent the number of systems that receive two attacks $\mathcal{T}$ days apart, at times $t - \mathcal{T}$ and $t$, respectively. Setting the number of attacks at time $t - \mathcal{T}$ as $\mathcal{U}(\theta_v, t - \mathcal{T}) = N\theta_v e^{-\lambda(t-\mathcal{T})}$ and the attacks received on the second vulnerability at time $t$ as $\mathcal{U}(\theta_{V\cup\{v\}}, t) = N\theta_{V\cup\{v\}} e^{-\lambda(t-T^\star)}$, we obtain

$$\begin{aligned} \mathcal{U}(\theta_{V\cup\{v\}}, t, \mathcal{T}) = \min\left(N\theta_v e^{\lambda\mathcal{T}}, N\theta_{V\cup\{v\}} e^{\lambda T^\star}\right) \cdot \\ \cdot \int_{\max(\mathcal{T}, T^\star)}^{\infty} e^{-\lambda t} dt. \quad (7) \end{aligned}$$

Solving for the two cases $T^\star > \mathcal{T}$ and $T^\star < \mathcal{T}$, we formulate the following claim:

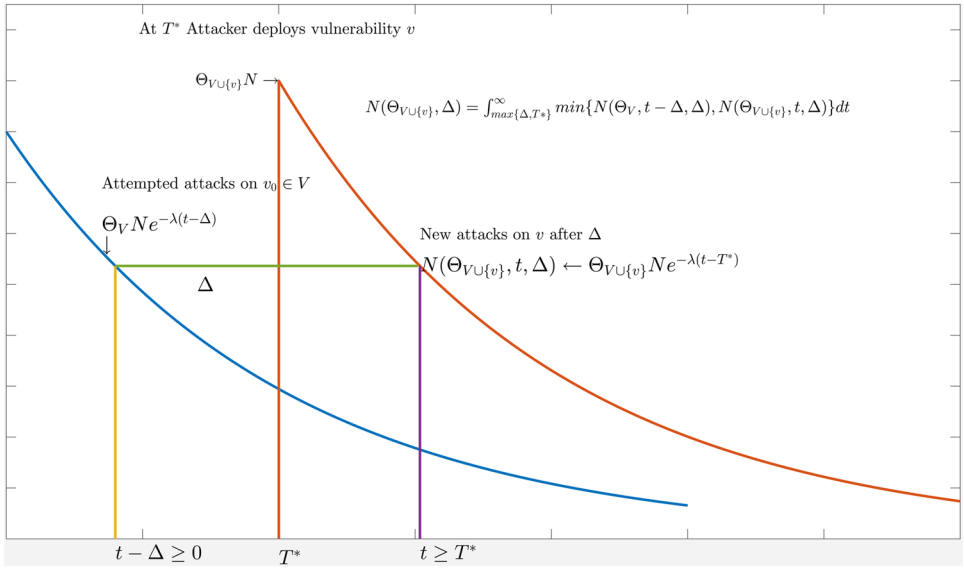


**Fig 1.** Computing the delay ($\mathcal{T}$) between attacks against different vulnerabilities.
*Note*: Change in the number of attacked systems for two attacks against different systems $\Delta = \mathcal{T}$ days apart. The first attack happens at $t - \mathcal{T} \geq 0$ and the number of attacked systems $\mathcal{U}(\Theta_V \cup \{v\}, t, \mathcal{T})$ is derived from Equation (1) as $\Theta_V N e^{-\lambda(t-\mathcal{T})}$. The number of systems attacked by the new exploit introduced at $T^\star$ is derived as $\mathcal{U}(\Theta_{V\cup\{v\}}, t, T^\star) = N\Theta_{V\cup\{v\}}e^{-\lambda(t-T^\star)}dt$.
*Note*: Change in the number of attacked systems for two attacks against different systems $\Delta = \mathcal{T}$ days apart. The first attack happens at $t - \mathcal{T} \geq 0$ and the number of attacked systems $\mathcal{U}(\Theta_V \cup \{v\}, t, \mathcal{T})$ is derived from Equation (1) as $\Theta_V N e^{-\lambda(t-\mathcal{T})}$. The number of systems attacked by the new exploit introduced at $T^\star$ is derived as $\mathcal{U}(\Theta_{V\cup\{v\}}, t, T^\star) = N\Theta_{V\cup\{v\}}e^{-\lambda(t-T^\star)}dt$.

**Claim 1.** *The sign of the coefficient for $\mathcal{T}$ oscillates from positive to negative as $\mathcal{T}$ increases.*

$$\log \mathcal{U}\ (\theta_{V\cup\{v\}}, t, \mathcal{T}) = \log\frac{N}{\lambda} + \begin{cases} -\lambda(T^\star - \mathcal{T}) + \log\theta_v & \text{if } T^\star > \mathcal{T} \\ +\lambda(T^\star - \mathcal{T}) + \log\theta_{V\cup\{v\}} & \text{if } T^\star < \mathcal{T} \end{cases} \quad (8)$$

The proof is available in the online SSRN report (Allodi et al., 2017). As the empirical evidence indicates (see Fig. 2) that $\mathcal{T}$ is substantial, we infer $T^\star < \mathcal{T}$. Hence, by substituting the corresponding term for $T^\star$ from Equation (6), we obtain the number of expected attacked systems after $\mathcal{T}$ days:

$$\log \mathcal{U}\ = -\lambda\mathcal{T} + \log\frac{N}{\lambda} + \log\theta_{V\cup\{v\}} + \lambda\left[\frac{1}{\delta}\log\left(\frac{C(v|V)}{r} - \frac{\delta}{\lambda+\delta}(\theta_{V\cup\{v\}} - \theta_V)N\right)\right] \quad (9)$$

## 5. HYPOTHESIS DERIVATION

Proposition 1 and Equation (9) can be used to define suitable empirical hypotheses. At first, we notice that for two vulnerabilities of the same software version, $\theta_{V\cup\{v\}} = \theta_V$, and therefore we hypothesize the following,

Hypothesis 1. A work-averse attacker has only one reliable exploit per software version.

The practical implications for mitigation mechanisms is significant: If attackers are likely to exploit different vulnerabilities of the same software, the *only* secure solution would be update the whole system. If only one vulnerability is exploited, one can resort to filtering those specific attacks by an intrusion prevention system (IPS), or deploying other vulnerability-specific defenses at the system level. For industrial control systems, that cannot be updated, deploying an IPS is the approach used in practice. Hypothesis 1 shows that this may actually be an effective defensive strategy, vastly reducing the scope of the threat caused by mass attackers. In turn, this simplifies decisions on defensive resource allocation and allows defenders to more effectively focus on defenses for different attackers and attack types (e.g., APT protection), as opposed to wasting resources to protect against *all* vulnerabilities for which an attack at scale will not, most likely, materialize (Zhang & Zhuang, 2019).

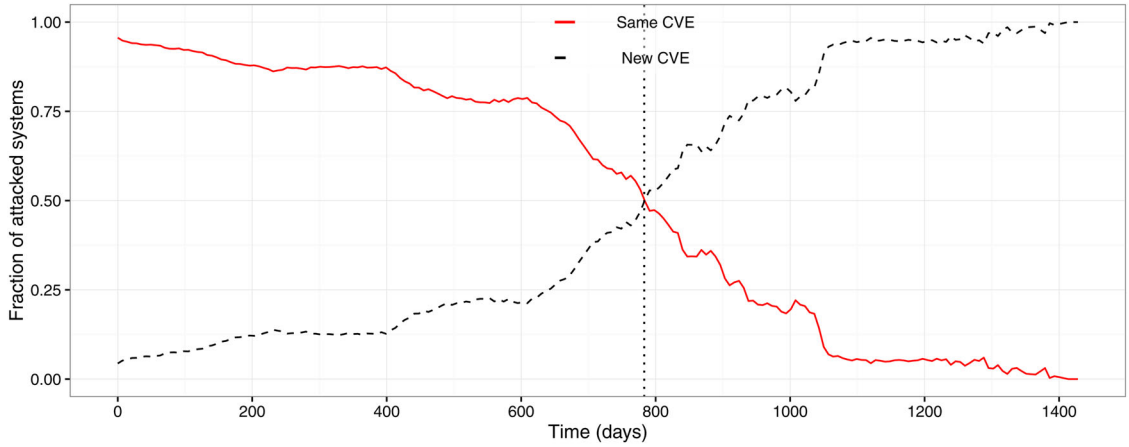


**Fig 2.** Distribution of time between of subsequent attacks with similar signatures.
*Note*: Fraction of systems receiving the same attack repeatedly in time (red, solid) compared to those receiving a second attack against a different vulnerability (black, dashed). The vertical line indicates number of days after the first attacks where it is more likely to receive an attack against a new vulnerability rather than against an old one
*Note*: Fraction of systems receiving the same attack repeatedly in time (red, solid) compared to those receiving a second attack against a different vulnerability (black, dashed). The vertical line indicates number of days after the first attacks where it is more likely to receive an attack against a new vulnerability rather than against an old one

When two vulnerabilities cover essentially the same fraction of the population ($\theta_{V\cup\{v\}} - \theta_V \approx \epsilon$), a low cost would make quick exploit development more appealing for an attacker because it would match the marginal condition $(C(v|V)/rN \leq \delta/(\lambda+\delta)(\theta_{V\cup\{v\}} - \theta_V) \approx \epsilon)$ when the attacker would consider deploying an exploit to have a positive marginal benefit. To capture this aspect, we observe that we have a suitable proxy among our parameters to capture development costs. The technical term of "exploit complexity" used by the CVSS standard refers to the technical possibility of easily developing a reliable exploit that works at all times without a "complex" engineering effort to cater for random factors (e.g., the specific system configuration, memory layout) that are outside the control of the attacker. Hence low complexity significantly decreases the fixed costs of development.

Hypothesis 2. A work-averse attacker has exploits with similar low complexity for similar popular software.

Assuming costs and rewards over $[0, T_i^*]$ are measured in the same numèraire and approximately within the same order of magnitude, the model implies that the discount factor (the term $1/\delta$ in Equation (6)) plays a leading role in determining the optimal time for the new exploit deployment. Microeconomics literature (Frederick, Loewenstein, & O'donoghue, 2002) sets $e^{\delta} - 1$ to vary between 1% and 20%. Hence, a lower bound on $T_1^*$ would be $\approx [100, 400]$ when time is measured in days.

Hypothesis 3. The time interval after which a new exploit would economically dominate an existing exploit is large (e.g., $T_1^* > 100$ days).

Since $\partial T^\star/\partial((\theta_v - \theta_V)N) < 0$, a larger number of attacked systems $\mathcal{U}$ on *different* versions $(\theta_v \neq \theta_V)$ would imply a lower delay $\mathcal{T}$ (as there is an attractive number of new systems that guarantee the profitability of new attacks). In contrast, the baseline rate of attacks impacts negatively the optimal time $\mathcal{T}$ as $\partial T^\star/\partial(\theta_V N) > 0$, since a larger pool of vulnerable machines makes it more profitable to continue with existing attacks (as per Hypothesis 1). The unconditional fraction of attacked systems with new updates from the WINE data set is illustrated in Fig. 2, where the crossover point of half the systems receiving attacks with the same signature but a new vulnerability targeted is around 800 days. It shows the key idea behind Hypothesis 4: If a good old exploit works, attackers will keep using it for a long time, even if a new exploit could be used.

Hypothesis 4. The possibility of launching a large number of attacks against systems for which an exploit already exists lengthens the time for weaponizing a new vulnerability ($\mathcal{N} \cdot (\texttt{Ver}_0 = \texttt{Ver}_v) \uparrow$ implies $\mathcal{T} \uparrow$), whereas an increase in potential attacks on

different systems is an incentive toward a shorter weaponization cycle ($\mathcal{N} \cdot (\texttt{Ver}_0 \neq \texttt{Ver}_v) \uparrow$ then $\mathcal{T} \downarrow$).

When considering the effects of costs, we observe that, as $\partial T^\star / \partial C(v|V) > 0$, the presence of a vulnerability with a low attack complexity implies $dC(v|V) < 0$, and therefore reflects a drop in the delay $\mathcal{T}$ between the two attacks. We have already discussed this possibility as Hypothesis 2. As for revenues, $\partial T^\star / \partial r < 0$ implies that a lower profit results in a longer time before it makes sense to engineer a new exploit targeting a new vulnerability. When the time to engineer a new exploit is substituted into the equation of the number of attacked machines that are needed to make a profit, a dual phenomenon takes place: An increase in revenue per attack means that less machines are needed to achieve the profit condition. We however cannot precisely measure the increase in revenues, as of course no telemetry data can reveal the exact revenue extracted from a system.[2]

Hypothesis 5. Vulnerabilities with higher impact increase revenue and therefore decrease the number of attacks ($\texttt{Imp}_{\texttt{CVE}_2} > \texttt{Imp}_{\texttt{CVE}_1}$ implies $\mathcal{U} \downarrow$).

## 6. DATA SET

To reconstruct the delay between arrival of multiple (identical, new) attacks on real systems, we build a data set where each row is a pair of attacks (targeting the same or different vulnerabilities) registered on similar systems deployed worldwide. The objective is to construct a data set that represents the decisions of attackers of the same "type" (see discussion on the model intuition at the beginning of Section 3) to update the attacked vulnerability (e.g., because those attackers target MS Windows machines in a specific region, and old attacks became ineffective against these targets). To construct these data, we merge information from three data sources:

[2] A possible proxy is to consider the *technical impact* of the attack as the level of system compromise that is possible to achieve by exploiting the vulnerability. Intuitively, the higher the level of access, the greater the potential revenue the attacker could extract from the system. A possible limitation is that such information might not correspond to the actual revenue for some specific attackers. For example, vulnerabilities that only compromise the availability of a system are typically scored low according to standard metrics such as the CVSS. (First.org, 2015) Yet, for an hacker offering Distributed Denial of Service (DDoS) targeted attack against online gamers, these vulnerabilities maybe the most interesting source of revenues (Hutchings & Clayton, 2016).

First, the National Vulnerability Database (NVD) is the vulnerability database maintained by the US NIST. Known and publicly disclosed vulnerabilities are published in this data set along with descriptive information such as publication date, affected software, and a technical assessment of the vulnerability as provided by the CVSS. Vulnerabilities reported in NVD are identified by a Common Vulnerabilities and Exposures identifier (CVE-ID) that is unique for every vulnerability.

Second, the Symantec threat report database (SYM) reports the list of attack signatures detected by Symantec's products along with a description in plain English of the attack. Among other information, the description reports the CVE-ID exploited in the attack, if any.

Third, the WINE, maintained by Symantec, reports attack signatures detected in the wild by Symantec's products. In particular, WINE is a representative, anonymized sample of the operational data Symantec collects from users that have opted in to share telemetry data (Dumitras & Shou, 2011). WINE comprises attack data from more than one million hosts, and for each of them, we are tracking up to three years of attacks. Attacks in WINE are identified by an ID that identifies the attack signature triggered by the detected event according to Symantec's threat database. To obtain the exploited vulnerability, we match the attack signature ID in WINE with the CVE-ID reported in SYM.

The data extraction involved three phases: (1) reconstruction of WINE users' attack history, (2) building the controls for the data, and (3) merging and aggregating data from (1) and (2). Because of user privacy concerns and ethical reasons, we did not extract from the WINE data set any potentially identifying information about its hosts. For this reason, it is useful to distinguish two types of tables: tables *computed* from WINE, namely, intermediate tables with detailed information that we use to build the final data set; and *extracted* tables, containing only aggregate information on user attacks that we use in this research. The full list of variables included in our data set is described in Table II. The full data set computed from WINE was collected in July 2013 and is available for sharing at Symantec Research Labs (under NDA clauses for access to the WINE repository) under the reference *WINE-2012-008*. A full replication guide is also available in Allodi et al. (2017).

We are interested in the new vulnerability $v$ whose mass exploit is being attempted in the wild after an exploit for $V$ vulnerabilities have been

**Table II.** Variables Included in Our Data Set

| Variable | Description |
|---|---|
| $\texttt{CVE}_{1,2}$ | The identifier of the previous and the current vulnerability $v$ exploited on the user's machine. |
| $\mathcal{T}$ | The delay expressed in fraction of year between the first and the second attacks. |
| $\mathcal{N}$ | The number of detected attacks for the pair *previous attack, actual attack*. |
| $\mathcal{U}$ | The number of systems attacked by the pair. |
| `Compl` | The Complexity of the vulnerability as indicated by its CVSS assessment. Can be either `High`, `Medium`, or `Low` as defined by CVSS(v2) Mell, Scarfone, and Romanosky (2007). |
| `Imp` | The impact of the vulnerability measured over the loss in confidentiality, integrity, and availability of the affected information. It is computed on a scale from 0 to 10 where 10 represents maximum loss in all metrics, and 0 represents no loss. Mell et al. (2007). |
| `Day` | The date of the vulnerability publication on the National Vulnerability Database. |
| `Sw` | The name of the software affected by the vulnerability. |
| `Ver` | The last version of the affected software where the vulnerability is present. |
| `Geo` | The country where the user system is at the time of the second attack. |
| `Hst` | The profile of the user or "host." |
| `Frq` | The average number of attacks received by a user per day. |
| `Pk` | The maximum number of attacks received by a user per day. |

**Table III.** Summary Excerpt from Our Data Set

| $\texttt{CVE}_1$ | $\texttt{CVE}_2$ | $\mathcal{T}$ | $\mathcal{U}$ | $\mathcal{N}$ | `Geo` | `Hst` |
|---|---|---|---|---|---|---|
| 2003-0533 | 2008-4250 | 83 | 186 | 830 | IT | Up |
| 2003-0818 | 2003-0818 | 146 | 1 | 1 | US | Rm |
| 2003-0818 | 2009-4324 | 616 | 1 | 1 | CH | Ev |
| 2003-0818 | 2009-4324 | 70 | 52 | 55 | US | Ev |

*Note*: We provide an example useful to interpret these data. Looking at the third row, one WINE system ($\mathcal{U}$= 1) located in Switzerland (`Geo`= CH) suffered only once ($\mathcal{N}$= 1) from an attack targeting the vulnerability $\texttt{CVE}_2$ = CVE-2009-4324 that was preceded by an attack targeting $\texttt{CVE}_1$ = CVE-2003-0818 almost two years earlier ($\mathcal{T}$= 616). In the fourth row, $\mathcal{U}$= 52 systems in the United States (`Geo`= US) received $\mathcal{N}$= 55 times the first attack on $\texttt{CVE}_1$ followed by the second attack on $\texttt{CVE}_2$ just two months apart ($\mathcal{T}$= 70). In both cases, the systems considered are of type EVOLVE, indicating that the affected systems have been upgraded and moved from some other country to the country listed in `Geo` during our observation period.

already engineered and attempted in the recent past. Our goal is to empirically evaluate whether this past is indeed more or less recent. To do so, we initially (1) extract from WINE two attack signatures received by a system (host) monitored by Symantec at different moments in time, (2) associate each attack signature to the corresponding vulnerability whose exploit is attempted (Combining WINE, SYM, and NVD), and (3) collect from WINE some features of the host, which suffered such attacks as control variables. We use the host's profile in terms of countries it connects to the Internet from, whether the host moves geographically, and whether the host upgraded to a new version of the operating system because users with profiles that change in time may look different to the attacker, and may therefore be subject to different attacks and attack volumes (Chen et al., 2011; Baltazar, 2011; Kotov & Massacci, 2013).

Table III reports an excerpt from the data set, with only selected columns for brevity. Each row represents a pair of detected attack signatures. The columns $\texttt{CVE}_1$ and $\texttt{CVE}_2$ report, respectively, the CVE-ID of the attacked vulnerability in $v$ and in the novel attack against $V$. Column $\mathcal{T}$ reports the time delay, measured in days, between the two attacks. Column $\mathcal{N}$ reports the overall number of attacks detected for $\texttt{CVE}_2$ after an attack against $\texttt{CVE}_1$; $\mathcal{U}$ reports the number of single systems receiving the same pair of attacks. Column `Geo` reports the country in which the second attack was recorded. Finally, `Hst` reports the type of user affected by the attack. Additional information regarding both attacked CVEs is extracted from the NVD (not reported in Table III): For each CVE, we collect publication date (`Day`), vulnerable software (`Sw`), last vulnerable version (`Ver`), and an assessment of the `Compl` of the vulnerability exploitation and of its `Imp`, provided by CVSS (v2).

As we mentioned, we associate an attack signature to the corresponding CVE-ID by combining information from WINE with Symantec own database of attack signatures (SYM). However, attack signatures as reported by Symantec have varying degrees of generality, meaning that they can be triggered by attacks that targets different vulnerabilities but still follow some common pattern. For this reason, some signatures reference more than one vulnerability. In this case, we have no means to know which of the vulnerabilities was effectively targeted by the attack. Of 1,573 different attack signatures, 112 involve more than one vulnerability; to avoid introducing counting errors on the number of attacks per CVE, we dropped these attack signatures from further consideration.

**Table IV.** Sample Attack Scenarios and Compatibility with Work-Aversion Hypothesis

| Type | Condition | Hypothesis |
|---|---|---|
| $A_1$: First attack and second attack affect precisely the same vulnerability and, consequently, the same software version | $\texttt{CVE}_1 = \texttt{CVE}_2$ | Often for Hypothesis 3 as $T^\star \to \infty$ |
| $A_2$: First attack and second attack affect the same software but different CVEs and different software versions. | $\texttt{CVE}_1 \neq \texttt{CVE}_2$, $\texttt{Sw}_{\texttt{CVE}_1} = \texttt{Sw}_{\texttt{CVE}_2}$, $\texttt{Ver}_{\texttt{CVE}_1} \neq \texttt{Ver}_{\texttt{CVE}_2}$ | Less frequent for Hypothesis 1 and Hypothesis 2 as $0 < T^\star < \infty$ |
| $A_3$: First and second attacks affect the same software and the same version but exploit different vulnerabilities | $\texttt{CVE}_1 \neq \texttt{CVE}_2$, $\texttt{Sw}_{\texttt{CVE}_1} = \texttt{Sw}_{\texttt{CVE}_2}$, $\texttt{Ver}_{\texttt{CVE}_1} = \texttt{Ver}_{\texttt{CVE}_2}$ | Almost never for Hypothesis 1 as $\theta_{V\cup\{v\}} = \theta_V$ |

*Note*: We expect the majority of attacks generated by the work-averse attacker to be of type $A_1$. Attack $A_2$ should be less frequent than $A_1$, as it requires to engineer a new exploit. $A_3$ contradicts the work aversion hypothesis and should be the least common type.

## 7. EMPIRICAL ANALYSIS

Prior to conducting any correlative analysis, we illustrate some scenarios that provide *prima facie* statistical evidence on the validity of the hypotheses identified from our theoretical model.

According to Equation (6), the attacker, will *postpone* the choice of weaponizing a vulnerability $v$ if the ratio between the cost of developing the exploit and the maximal marginal expected revenue is larger than the discounted increase in the fraction of exploited vulnerabilities, namely, $C(v|V)/rN > \delta/(\lambda+\delta)(\theta_{V\cup\{v\}} - \theta_V)$. Empirically, this means that the attacker should prefer to (i) attack the same vulnerability multiple times rather than for only a short period of time and (2) create a new exploit only when they want to attack a new software version. To evaluate these scenarios, we identify three types of attack pairs that are summarized in Table IV: In the first type of attack pair ($A_1$), the first attack and the second attack affect the same vulnerability and, consequently, the same software version; in the second pair ($A_2$), the first attack and the second attack affect the same software, but different CVEs and different software versions; finally in the third pair, the first and second attacks affect the same software and the same version but exploit different vulnerabilities ($A_3$). According to our hypothesis, we expect that attacks of type $A_1$ should be more common than $A_2$ (in particular when the delay between the attacks is small), while $A_3$ should be the least common of the three.

To evaluate the relative ordering of these attack types, it is important to consider that users have diverging models of software security (Wash, 2010), and different software have different update patterns, frequencies, and attack vectors (Nappa et al., 2015; Provos, Mavrommatis, Rajab, & Monrose, 2008). For example, an attack against a browser may only require the user to visit a webpage, while an attack against a word processing application may need the user to actively open a file on the system (see also the definition of the `Attack Vector` metric in the CVSS standard). As these clearly require a different attack process, we further classify `Sw` in four categories: SERVER, PLUGIN, PROD(-uctivity), and Internet Explorer (the only browser represented in our WINE data). The categories are defined by the software names in the database. For example, SERVER environments are typically better maintained than "consumer" environments and are often protected by perimetric defenses such as firewalls or IDSs. This may in turn affect an attacker's attitude toward developing new exploits. This may require the attacker to engineer different attacks for the same software version in order to evade the additional mitigating controls in place. Hence we expect the difference between $A_2$ and $A_3$ to be narrower for the SERVER category. Fig. 3 reports a fitted curve of targeted machines as a function of time by software category. As expected, $A_1$ dominates in all software types. The predicted order is valid for PLUGIN and PROD. For PROD software, we find no attacks against new vulnerabilities for different software versions, therefore $A_2 = A_3 = 0$. This may be an effect of the low update rate of this type of software and relatively short timeframe considered in our data set (three years), or of a scarce attacker interest in this software type. Results for SERVER are mixed: The difference between $A_2$ and $A_3$ is very narrow and $A_3$ is occasionally higher than $A_2$. Since oscillations occur within confidence intervals, they might be due to chance.

Internet Explorer is an interesting case in itself. Here, contrary to our prediction, $A_3$ is higher than

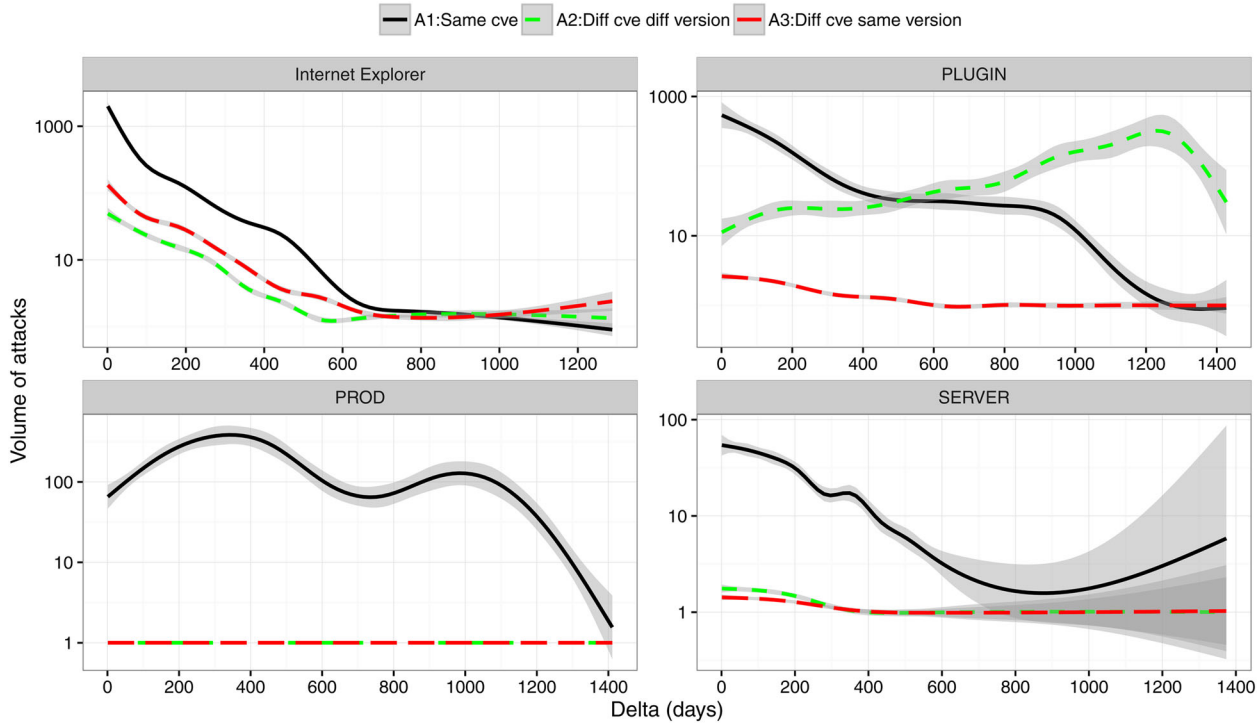


**Fig 3.** Loess regression of volume of attacks in time.
*Note*: Volume of received attacks as a function of time for the three types of attack. $A_1$ is represented by a solid black line, $A_2$ by a long-dashed red line, $A_3$ by a dashed green line. The gray areas represent 95% confidence intervals. For Internet Explorer vulnerabilities, the maximum $\mathcal{T}$ between two attacks is 1,288 days; for SERVER it is 1,374 days; PROD 1,411; PLUGIN 1,428. This can be determined by the timing of first appearance of the attack in the WINE database.
*Note*: Volume of received attacks as a function of time for the three types of attack. $A_1$ is represented by a solid black line, $A_2$ by a long-dashed red line, $A_3$ by a dashed green line. The gray areas represent 95% confidence intervals. For Internet Explorer vulnerabilities, the maximum $\mathcal{T}$ between two attacks is 1,288 days; for SERVER it is 1,374 days; PROD 1,411; PLUGIN 1,428. This can be determined by the timing of first appearance of the attack in the WINE database.

$A_2$. By further investigating the data, we find that the reversed trend is explained by one single outlier pair: $\texttt{CVE}_1$ = CVE-2010-0806 and $\texttt{CVE}_2$ =CVE-2009-3672. These vulnerabilities affect Internet Explorer version 7 and have been disclosed 98 days apart. More interestingly, they are very similar: They both affect a memory corruption bug in Internet Explorer 7 that allows for an heap-spray attack resulting in arbitrary code execution. Two observations are particularly interesting:

1. Heap spray attacks are unreliable attacks that may result in a significant drop in exploitation success. This is reflected in the `Access Complexity:Medium` assessment assigned to both vulnerabilities by the CVSS v2 framework. In our model, this would imply a lower return $r(t, N_V(t), V)$ for the attacker, as the unreliable exploit may yield control of fewer machines among the vulnerable ones.
2. The exploitation code found on Exploit-DB[3] is essentially the same for these two vulnerabilities. The code for $\texttt{CVE}_2$ is effectively a rearrangement of the code for $\texttt{CVE}_1$, with different variable names. In our model, this would indicate that the cost $C(v|V) \approx 0$ to build an exploit for the second vulnerability is negligible, as most of the exploitation code can be reused from $\texttt{CVE}_1$ (see the Appendix for details).

Hence, this vulnerability pair is only an apparent exception: The very nature of the second exploit for Internet Explorer 7 is coherent with our model and in

[3]See Exploit-DB (http://www.exploit-db.com, last accessed January 15, 2017.), which is a public data set for vulnerability proof-of-concept exploits. $\texttt{CVE}_1$ corresponds to exploit 16547 and $\texttt{CVE}_2$ corresponds to exploit 11683.

**Table V.** Summary of Predictions Derived from the Model

| Model Variable | Regressor | Expectation | Hypothesis | Rationale |
|---|---|---|---|---|
| $\mathcal{T}$ | $\mathcal{T}$ | $\beta_1 < 0$ | 3 and 4 | Shorter exploitation times are associated with more vulnerable systems, hence $\mathcal{T} \uparrow \Rightarrow \mathcal{U} \downarrow$. |
| $C(V\|v)$ | $\texttt{Compl}_{CVE2,L}$ | $\beta_2 < 0$ | 1, 4, and 2 | The introduction of a new reliable, low-complexity exploit minimizes implementation costs, thus $C \downarrow \Rightarrow \mathcal{U} \downarrow$. |
| $\theta_{V\cup\{v\}}$ | $\texttt{Imp}_{CVE2,H}$ | $\beta_3 > 0$ | 5 and 4 | High impact vulnerabilities allow the attacker for a complete control of the attacked systems, hence $\theta_{V\cup\{v\}} \uparrow \Rightarrow \mathcal{U} \uparrow$. |
| $r$, $(\theta_{V\cup\{v\}} - \theta_V)$ | $\texttt{Imp}_{CVE2} > \texttt{Imp}_{CVE1}$ | $\beta_4 < 0$ | 5 | Selecting a higher impact exploit for a new vulnerability increases the expected revenue and increases the fraction of newly controlled systems with respect to the old vulnerability. $r \uparrow \Rightarrow \mathcal{U} \downarrow$ and $(\theta_{V\cup\{v\}} - \theta_V) \uparrow \Rightarrow \mathcal{U} \downarrow$. |

line with Hypothesis 1 and Hypothesis 2. Removing the pair from the data confirms the order of attack scenarios identified in Table IV.

## 8. DATA ANALYSIS

Table V summarizes the predictions implied by the solution to the model given in Equation (9). $\mathcal{T}$ can be measured directly in our data set; the cost of development of an exploit $C(v|V)$ can be estimated by the proxy variable $\texttt{Compl}_{\texttt{CVE}_2}$, as by definition the complexity associated with exploit development requires additional engineering efforts (and is thus related to an increase in development effort). We cannot directly measure the revenue $r$ and the number of systems $N$ affected by the vulnerability, but we can estimate the effect of an attack on a population of users by measuring the impact (Imp) of that vulnerability on the system: Higher impact vulnerabilities (i.e., $\texttt{Imp}_{CVE2} > \texttt{Imp}_{CVE1}$) allow the attacker to control a *higher fraction* of the vulnerable system, and therefore extract higher revenue $r$ from the attack. Similarly, the introduction of an attack with a higher impact can approximate the difference in attack penetration $(\theta_{V\cup\{v\}} - \theta_V)N$ for the new set of exploits as it allows the attacker for a higher degree of control on the affected systems. Finally, high impact vulnerabilities ($\texttt{Imp}_{CVE2,H}$), for example, allowing remote execution of arbitrary code on the victim system, leave the $\Theta_{V\cup\{v\}}N$ systems under complete control of the attacker; in contrast, a low impact vulnerability, for example, causing a denial of service, would allow for only a temporary effect on the machine and therefore a lower degree of control. It is important to note that other vulnerability characteristics, such as requirements on attacker positioning (e.g., local to the system, or remote) and preexistent privileges required for the attack to work may have an impact on the decisions of an attacker. On the other hand, previous research showed that, considering mass attackers, only certain types of vulnerability are effectively exploited at scale (Allodi & Massacci, 2014): Mass attackers generally attack from remote, do not have preexistent privileges on the vulnerable system, and prefer vulnerabilities for which no user interaction is required (to avoid detection, and therefore maintain exploit functionality, for longer; Allodi, 2017). Using CVSS as the framework of reference to evaluate vulnerability characteristics, most of the variability in vulnerabilities exploited at scale is captured by the relationship between attack complexity and vulnerability impact (Allodi & Massacci, 2014). Hence, these are the main factors we capture in our model. A limitation of our data set is that we cannot ascribe a specific collection of observations to a specific individual. This unobserved variable may bias our result. To attempt to correct for this statistical feature, we identify commonalities in attacks by including a number of additional components based on the type of target victim: receiving thousands of attacks a day versus an handful a year, moving in space or upgrading their software, and geographical location.

**Table VI.** Ordinary Least Squares and Robust Regression Results

| Dependent Variable: Natural Logarithm of the Number of Attacked Machines $\log(\mathcal{U}_i)$ | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Model 1 | | | | Model 2 | | | | Model 3 | | | |
| | OLS | | Robust | | OLS | | Robust | | OLS | | Robust | |
| | $Z1:Z8$ | | $Z1:Z8$ | | $Z1:Z8$ | | $Z1:Z8$ | | $Z1:Z8$ | | $Z1:Z8$ | |
| c | 0.927 | 0.006 | 0.731 | 0.096 | 1.065 | 0.122 | 0.845 | 0.171 | 0.933 | −0.106 | 0.783 | 0.039 |
| | (0.001) | (0.003) | (0.001) | (0.003) | (0.001) | (0.003) | (0.001) | (0.003) | (0.004) | (0.005) | (0.003) | (0.004) |
| $\mathcal{T}$ | 0.018 | −0.051 | 0.012 | −0.044 | −0.006 | −0.092 | −0.003 | −0.071 | −0.005 | −0.091 | −0.004 | −0.071 |
| | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) |
| $\texttt{Compl}_{\text{CVE2 L}}$ | | | | | −0.326 | −0.479 | −0.228 | −0.324 | −0.313 | −0.464 | −0.22 | −0.314 |
| | | | | | (0.002) | (0.002) | (0.001) | (0.001) | (0.002) | (0.002) | (0.001) | (0.001) |
| $\texttt{Imp}_{\text{CVE2 H}}$ | | | | | | | | | 0.144 | 0.236 | 0.063 | 0.131 |
| | | | | | | | | | (0.003) | (0.003) | (0.003) | (0.003) |
| $\texttt{Imp}_{\text{CVE 2}} > \texttt{Imp}_{CVE1}$ | | | | | | | | | −0.088 | −0.209 | 0.012 | −0.087 |
| | | | | | | | | | (0.003) | (0.003) | (0.002) | (0.002) |
| Z1: Geo North. Amer. | | 0.604 | | 0.37 | | 0.679 | | 0.422 | | 0.671 | | 0.419 |
| | | (0.002) | | (0.001) | | (0.002) | | (0.001) | | (0.002) | | (0.001) |
| Z2: Geo West. Eu. | | 0.155 | | 0.105 | | 0.17 | | 0.116 | | 0.163 | | 0.114 |
| | | (0.002) | | (0.002) | | (0.002) | | (0.002) | | (0.002) | | (0.002) |
| Z3: Hst EVOLVE | | 0.191 | | 0.129 | | 0.208 | | 0.141 | | 0.223 | | 0.149 |
| | | (0.002) | | (0.002) | | (0.002) | | (0.002) | | (0.002) | | (0.002) |
| Z4: Hst UPGRADE | | 0.112 | | 0.072 | | 0.116 | | 0.076 | | 0.113 | | 0.075 |
| | | (0.002) | | (0.002) | | (0.002) | | (0.002) | | (0.002) | | (0.002) |
| Z5: Frq HIGH | | 0.24 | | 0.147 | | 0.212 | | 0.127 | | 0.279 | | 0.157 |
| | | (0.003) | | (0.003) | | (0.003) | | (0.003) | | (0.003) | | (0.003) |
| Z6: Frq MEDIUM | | 0.328 | | 0.227 | | 0.358 | | 0.246 | | 0.41 | | 0.271 |
| | | (0.002) | | (0.002) | | (0.002) | | (0.002) | | (0.002) | | (0.002) |
| Z7: Pk HIGH | | 0.513 | | 0.442 | | 0.567 | | 0.49 | | 0.531 | | 0.477 |
| | | (0.004) | | (0.003) | | (0.004) | | (0.003) | | (0.004) | | (0.003) |
| Z8: Pk MEDIUM | | 0.379 | | 0.274 | | 0.412 | | 0.299 | | 0.411 | | 0.301 |
| | | (0.003) | | (0.002) | | (0.003) | | (0.002) | | (0.003) | | (0.002) |
| *Pseudo* $R^2$ | – | – | 0.326 | 0.341 | – | – | 0.331 | 0.347 | – | – | 0.331 | 0.347 |
| $R^2$ | 0.00 | 0.093 | – | – | 0.016 | 0.126 | – | – | 0.017 | 0.13 | – | – |
| $F$ | 348.66 | 26,551.47 | – | – | 18,548.25 | 33,422.78 | – | – | 9,989.88 | 28,915.60 | – | – |
| Obs. | 2324500 | 2324500 | 2324500 | 2324500 | 2324500 | 2324500 | 2324500 | 2324500 | 2324500 | 2324500 | 2324500 | 2324500 |

*Note*: Model 1: $\log(\mathcal{U}_i) = \beta_0 + \beta_1 \mathcal{T}_i + \epsilon_i$
Model 2: $\log(\mathcal{U}_i) = \beta_0 + \beta_1 \mathcal{T}_i + \beta_2 \texttt{Compl}_{i,CVE2,L} + \epsilon_i$
Model 3: $\log(\mathcal{U}_i) = \beta_0 + \beta_1 \mathcal{T}_i + \beta_2 \texttt{Compl}_{i,CVE2,L} + \beta_3 \texttt{Imp}_{i,CVE2,H} + \beta_4 \texttt{Imp}_{i,CVE2} > \texttt{Imp}_{i,CVE1} \epsilon_i$
The three model equations reflect the definition of the expected (log) number of affected machines after an interval $\mathcal{T}$. The regression model formulation is derived from prime principle from Equation (9). The expected coefficient signs are given in Table V. For each model, we run four sets of regressions. OLS and robust regressions are provided to addresses heteroscedasticity in the data. $R^2$ and $F$-statistics are reported for the OLS estimations. Note that the pseudo-$R^2$ are computed for the robust regressions, using the McFadden-adjusted approach $R^2 = 1 - (\log(LL_{full}) - K)/\log(LL_{int})$, where $\log(LL_{full})$ is the log likelihood for the full model minus the number of slope parameters $K$ versus the log likelihood of the intercept alone and should not be compared directly to the OLS $R^2$. Coefficient estimations of the two sets of regressions are consistent. All coefficient signs for the three models reflect the work-averse attacker model predictions, with the only exception of the estimation for $\mathcal{T}$ with no controls for which the prediction for $\beta_1$ is inverted. This may indicate that user characteristics are relevant factors for the arrival time of exploits when other factors related to the system are not accounted for. The introduction of Compl in Model 2 significantly changes the estimate for $\beta_1$, whereas Imp in Model 3 leaves the estimates for Compl and $\mathcal{T}$ unchanged. High Imp vulnerabilities tend to increase volume of attacks. We report only standard errors without starring $p$-values as all coefficients are significant due to the number of observations in the data set. All standard errors are estimated using the Huber–White approach.

Descriptive statistics of these variables are provided in the Supporting Information (Allodi et al., 2017).

We present the estimates of Equation (9) from data in Table VI, with a number of conditioning variables. These range from just a constant (Model 1, first column) to Model 3 where we include all available conditioning variables to extract systematic attack characteristics. It is important to note that for the model to be consistent with the properties of the observed empirical data the coefficient predictions

from Hypotheses 1–5 summarized in Table V must be satisfied. All predictions are confirmed by the data. We utilize two estimators as we have little information on the error structure of the regression model and we are subject to certain statistical issues caused by the right truncation of the data, that is, we do not observe $\mathcal{T}$ asymptotically by construction. First is a simple ordinary least squares (OLS) estimator with Huber–White standard errors and second is a robust fit model that utilizes a weighted least squares (WLS) type estimator with iterative reweighting and we implement the sandwich form standard error from the WLS iterations. The weighting function for the iterative reweighting is a bisquare function, experimentation with spectral and Andrews-type weightings suggest the regressions are insensitive to kernel and tuning function. For the robust fit, we compute a McFadden adjusted pseudo-$R^2$, which sets the numerator as the log likelihood function at the estimate and the denominator as the log likelihood of just the intercept alone. Note that it is not appropriate to compare directly the pseudo-$R^2$ and the $R^2$ from the OLS estimates, which suggests that the model captures roughly 10% of the variation in numbers of attacked machines, as opposed to explaining 35% of the model likelihood for the pseudo-$R^2$.

The set of OLS and robust regressions returns very similar estimations. We also experimented with various regression estimators (e.g., 2SLS, 3SLS) and they produced markedly similar results to OLS, subject to the standard caveats on misidentification. The introduction of the controls only change the sign of $\beta_1$ from positive to negative for Model 1. This may indicate that the type of user is a significant factor in determining the number of delivered attacks, which is consistent with previous findings (Nappa et al., 2015). Interestingly, the factor that introduces the highest change in the estimated coefficient $\beta_1$ for $\mathcal{T}$ is `Compl` (Model 2), whereas its estimate remains essentially unchanged in Model 3. This may indicate that the cost of introduction of an exploit has a direct impact on the time of delivery of the exploit. The coefficients for all other regressors are consistent across models, and their magnitude changes only slightly with the introduction of the controls. This observation is to be expected: User characteristics should not influence the characteristics of the vulnerabilities present on the system; as such, the distribution of attacks in the wild seems to depend mostly on system characteristics rather than user type.

The signs of coefficients for the `Imp` variables suggest that both impact of a new vulnerability and its relation with the impact of previous vulnerabilities have an effect on the number of attacked systems. Interestingly, a *high* impact encourages the deployment of attacks and increases the number of attacked systems, whereas the introduction of a *higher* impact vulnerability requires the infection of a smaller number of systems as revenues extracted from each machine increase. Hence, when introducing a new exploit, the attacker will preferably choose one that grants a higher control over the population of users ($\theta_{V\cup\{v\}} > \theta_V$) and use it against a large number of system. This is consistent with recent findings suggesting that vulnerability severity alone is not a good predictor for exploitation in the wild (Allodi & Massacci, 2014; Bozorgi et al., 2010). Other factors such as software popularity may play a role (Nayak et al., 2014).

## 9. SUMMARY OF FINDINGS AND LIMITATIONS

This article implements a model of the *Work-Averse Attacker* as a new conceptual framing to understand cyber threats. Our model presumes that an attacker is a resource-limited actor with fixed costs that has to choose which vulnerabilities to exploit to attack the "mass of Internet systems." Work aversion simply means that effort for the attacker is costly (in terms of cognition and opportunity costs), hence a trade-off exists between effort exerted on new attacking technologies and the anticipated reward schedule from these technologies. As systems in the wild get patched unevenly and often slowly in time (Nappa et al., 2015), we model the production of new vulnerability exploits following Stokey's "economy of inaction," whereby "doing nothing" up to a certain time is the best strategy. A cost constraint driving the attacker's exploit selection strategy naturally emerges from the model. In particular, we find theoretical and empirical evidence as follows:

1. First, an attacker massively deploys only one exploit per software version. The only exception we found is for Internet Explorer; the exception is characterized by a very low cost to create an additional exploit, where it is sufficient to essentially copy and paste code from the old exploit, with only few modifications, to obtain the new one. This finding is predicted by the model and supports Hypothesis 1.
2. Second, low complexity vulnerabilities for which a reliable exploit can be easily engineered lower the production costs and favor the

deployment of the exploit. This finding supports Hypothesis 2.

3. Third, the attacker deploys new exploits relatively slowly over time, driven by a slowly decreasing instantaneous profit function; empirically, we find that attacks 1,000 days apart are still driven by the same exploits in about 20% of the cases, and that the effect of the passage of time in between attacks ($\mathcal{T}$) on the number of affected system is indeed negative and very small. This finding supports Hypothesis 3 and Hypothesis 4.
4. Fourth, the presence of a high impact vulnerability increases the incidence of exploitation in the wild. Similarly, gaining a higher control over attacked systems heightens the attacker's revenue and decreases the number of systems that need to be infected to balance costs. This supports Hypothesis 5.

Such findings should be considered in the framework of the limitations of the data that we have collected, and the theory we have developed. The "Work-Averse Attacker" may be only one of the possible explanations of the distribution of exploits in the wild previously noted in the literature (Allodi, 2015; Nayak et al., 2014). For example, it could be that only a handful of individuals possess the technical skills to develop (and subsequently distribute to the mass of attackers) working exploits. The strong skew in the exploit distributions could then be explained by those individuals not being work-averse, but only in terms of available capacity for exploit production. To evaluate this possibility in the data is hard or impossible as it would require to identify (all) exploit developers and observe the exploit production process. On the other hand, the presence of competitive underground markets where multiple actors trade different, but long lived, exploits, and malware techniques does not appear compatible with the hypothesis of only a handful of productive but time-limited exploit developers (Allodi, 2017). Alternatively, explanations for the data could be identified in the efficiency of the *supply chain* of the components needed to engineer and deploy an attack: If finding producers of the necessary attack components (or establishing business relationships with them) is hard, the inefficiency of the required supply chain could explain the observed delays in the exploit deployment process, and the scarcity of available exploits for attacks at scale. The existence of composed services for the delivery of attacks is clear evidence of the existence of this supply chain, at least for attack provision and delivery (Campobasso & Allodi, 2020; Grier et al., 2012). These inefficiencies could push attackers to strategize on which exploits to develop, leading to similar output dynamics as those considered in this article. A rigorous evaluation of the supply chain of cyberattacks is hard to perform (Bhalerao, Aliapoulios, Shumailov, Afroz, & McCoy, 2019), but may shed additional light on the bottlenecks or hardship of development of exploits for attacks at scale. Importantly, work-averse dynamics may still emerge, from this setting, underlying that the complexity of the problem requires a deeper empirical understanding of the ecosystem enabling attacker operations at scale.

Other limitations concern the nature of the data. Records of attacks detected over a user's machine are necessarily conditioned over the user's proneness in receiving a particular attack. For example, a user may be inclined to open executable email attachments, but not in visiting suspicious websites. Thus, there may be a disassociation between the observed attacks and those engineered by the attacker. For our empirical data set, this limitation is mitigated by *WINE* reporting attack data on a very large representative sample of Internet users (Dumitras & Shou, 2011). Albeit we do have some system-level information (e.g., geographic location, system evolution), we do not have all possible conditioning user variables (e.g., educational level), which are very difficult or close to impossible to gauge at the scale of data needed for this type of analysis. Similarly, software versioning information is known to be unreliable at times with respect to vulnerability existence (Nguyen, Dashevskyi, & Massacci, 2015). Further, software versions cannot be easily "ordered" throughout software types, as different vendors adopt different naming schemes for software releases (for an overview, see, e.g., Christey & Martin, 2013). We cannot therefore order software versions over time easily. Another limitation of our data set is the market penetration of Symantec. In 2016 (i.e., around the time of the data collection), Symantec self-reported that it is the largest security vendor for the last 15 years by market share in antivirus and overall software security, and hence has a broad coverage recording attacks on customers. However, third-party verifiable measurement of these claims are difficult, hence replication studies across different security vendors would be welcome.

## 10. CONCLUSIONS AND IMPLICATIONS

This article develops the thesis that an attacker will generally "avoid to work" until the perceived utility of the deployment of a new attack becomes positive w.r.t. expectations derived from previous attacks. This economic perspective has been previously employed in game-theoretic approaches (Manshaei et al., 2013), and it typically considers two actors (namely, the defender and the attacker) that react to each other's strategies. The realistic threat modeling is of key importance in this context, and has been identified multiple times in the system resilience (Gisladottir et al., 2017; Guikema et al., 2015) and security (Do et al., 2017; Hewett, Rudrapattana, & Kijsanayothin, 2014) literature. This article is the first to propose and validate this approach for the "mass attacker" that deploys attacks against the vast Internet population. In this respect, this contribution provides a better theoretical and empirical understanding for the behavior of "untargeted" mass attackers: *A slow periodic update of an attacker's arsenal with selected picks of low hanging fruits* seems to be the theoretical and empirically winning strategy. This finding is particularly interesting because recent, game-theoretic work on APTs has also shown that periodic renewal strategies might also be dominant strategies for targeted attacks (van Dijk, Juels, Oprea, & Rivest, 2013). This dominance, in the Nash equilibrium sense, remains even in the case where the attacker can reliably evaluate some characteristics of the defender's setting such as system configuration changes or an average patching rates (Nappa et al., 2015).

From the perspectives of cyber-risk assessment, this means that several alternative strategies might be equally successful than the "upgrade to the last version" (or "do nothing" if such upgrade is not possible) strategy, which currently dominates risk mitigation best practices. For example, maintaining intrusion detection systems (IDS/IPS) signatures for the low hanging fruit vulnerabilities might be a better option than updating the software, because one IDS signature eliminates the majority of risks faced be that system; a software patch may 'overdo-it" by fixing more vulnerabilities than necessary at a severe functional costs (Dashevskyi, Brucker, & Massacci, 2018; Huang, Borges, Bugiel, & Backes, 2019). An important assumption in this respect is the cost of exploit maintenance being negligible as some empirical research implies (Castro et al., 2019; Kotov & Massacci, 2013). If the maintaining the success of exploits would become a significant part of the cost, attackers might still be work averse but different dynamics or mechanisms may emerge. We leave this work for future research.

Another major implication for this research work is the current policy discussion on the timing of vulnerability disclosures. The United States Department of Commerce NTIA set up multistakeholders forum to discuss procedures and timings of the vulnerability disclosure process.[4] This discussion is not currently guided by a theoretical framework for decisionmakers to estimate effect in terms of the effective increase in risk of attacks that follows the disclosure (Mitra & Ransbotham, 2015). Our findings would indicate that there is a limited risk in additional disclosures of minor vulnerabilities for the same software version (i.e., Hypothesis 1). Further, the time/space dimension may also be relevant to evaluate from a policy perspective, for example, by asynchronously releasing patches to users or by deploying different versions across systems. By diversifying software (Chen et al., 2011; Homescu, Neisius, Larsen, Brunthaler, & Franz, 2013), the defender can effectively decrease the number of systems the attacker can compromise with one exploit, effectively making the existence conditions for Equation (6) hard to satisfy. For example, a random distribution of patches would simply decrease the fraction of attackable systems regardless of the attacker's choice in which vulnerability to exploit. Moreover, diversifying defenses may be in fact less onerous than recompiling code bases (when possible) or maintaining extremely diverse operational environments. More studies are needed to evaluate cascading effects of generalized strategies against "mass attackers" on exposure to attacks of other types (e.g., perpetrated by APT-level attackers capable of adapting to specific system conditions). In general, a more precise and data-grounded understanding of the attacker poses a strategic advantage for the defender (Dey et al., 2015). This article is a step in this direction.

## ACKNOWLEDGMENTS

The authors would like to thank Matthew Elder and Symantec for their support in accessing and interpreting the data in the WINE Data set. The anonymous reviewers and the participants to the numer-

[4]The NTIA forum can be found at https://www.ntia.doc.gov/other-publication/2016/multistakeholder-process-cybersecurity-vulnerabilities, last accessed October 2020

ous seminars (Dartmouth, KULeuven, Rome "La Sapienza," UCL, USC-ISI) and events (WEIS, SRA Symposium) where this article was presented provided very useful feedback that greatly helped improving the article.

The authors contributed equally to the article. The research behind this article was partly funded by the European Union under the H2020 Programme Grant Agreement Numbers 830929 (CyberSec4Europe -https://cybersec4europe.eu/) and 952647 (AssureMOSS - https://assuremoss.eu), the Government of the Netherlands under the Sectorplan, and the ITEA3 program through the DEFRAUDIfy project funded by Rijksdienst voor Ondernemend Nederland (Grant Number ITEA191010). Graphical abstract by Anna Formilan (https://annaformilan.com), used with permission.

## REFERENCES

Allodi, L. (2015). The heavy tails of vulnerability exploitation. In Piessens Frank Caballero Juan & Bielova Nataliia *Proceedings of the 2015 Engineering Secure Software and Systems Conference (ESSoS'15)*. Lecture Notes in Computer Science, *8978*, Springer, Cham.

Allodi, L. (2017). Economic factors of vulnerability trade and exploitation. In Bhavani Thuraisingham David Evans Tal Malkin & Dongyan Xu *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 1483–1499). New York, NY, USA: Association for Computing Machinery.

Allodi, L., Corradin, M., & Massacci, F. (2015). Then and now: On the maturity of the cybercrime markets the lesson that black-hat marketeers learned. *IEEE Transactions on Emerging Topics in Computing*, *4*(1), 35–46.

Allodi, L., Kotov, V., & Massacci, F. (2013). Malwarelab: Experimentation with cybercrime attack tools. In *Proceedings of the 2013 6th Workshop on Cybersecurity Security and Test*. Washington, D.C., USA: USENIX Associationm.

Allodi, L., & Massacci, F. (2013). *How CVSS is dossing your patching policy (and wasting your money)*. USA: BlackHat. arXiv:1301.1275 [cs.CR].

Allodi, L., & Massacci, F. (2014, August). Comparing vulnerability severity and exploits using case-control studies. *ACM Transaction on Information and System Security (TISSEC)*, *17*(1), Article 1.

Allodi, L., & Massacci, F. (2017). Security events and vulnerability data for cybersecurity risk estimation. *Risk Analysis*, *37*(8), 1606–1627.

Allodi, L., Massacci, F., & Williams, J. M. (2017). Online supplement of the work-averse cyber attacker model: Theory and evidence from two million attack signatures. Available on SSRN at https://ssrn.com/abstract=2862299.

Baltazar, J. (2011). More traffic, more money: Koobface draws more blood. Technical report, TrendLabs, Retrieved from https://www.trendmicro.co.nz/cloud-content/us/pdfs/security-intelligence/reports/rpt_koobface-draws-more-blood.pdf.

Beattie, S., Arnold, S., Cowan, C., Wagle, P., Wright, C., & Shostack, A. (2002). Timing the application of security patches for optimal uptime. In Alva Couch *Proceedings of LISA '02: Sixteenth Systems Administration Conference*, (Vol. *2*, pp. 233–242). Berkeley, California, US: USENIX Association.

Bhalerao, R., Aliapoulios, M., Shumailov, I., Afroz, S., & McCoy, D. (2019). Mapping the underground: Supervised discovery of cybercrime supply chains. In *2019 APWG Symposium on Electronic Crime Research (eCrime)* (pp. 1–16). Pittsburgh, PA, USA: IEEE.

Bilge, L. & Dumitras, T. (2012). Before we knew it: An empirical study of zero-day attacks in the real world. In Yu Ting Danezis George & Gligor Virgil *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS'12)* (pp. 833–844). New York, NY, USA: Association for Computing Machinery.

Birge, J. R. & Louveaux, F. (2011). *Introduction to stochastic programming*. New York: Springer Science & Business Media.

Bozorgi, M., Saul, L. K., Savage, S., & Voelker, G. M. (2010). Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In Rao Bharat Krishnapuram Balaji Tomkins Andrew & Yang Qiang *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery.

Brand, M., Valli, C., & Woodward, A. (2010). Malware forensics: Discovery of the intent of deception. *The Journal of Digital Forensics, Security and Law: JDFSL*, *5*(4), 31.

Brito, J. & Watkins, T. (2011). Loving the cyber bomb-the dangers of threat inflation in cybersecurity policy. *Harvard National Security Journal*, *3*, 39.

Brown, G. G. & Cox Jr, L. A. T. (2011). How probabilistic risk assessment can mislead terrorism risk analysts. *Risk Analysis*, *31*(2), 196–204.

Campobasso, M. & Allodi, L. (2020). Impersonation-as-a-service: Characterizing the emerging criminal infrastructure for user impersonation at scale. In Ligatti Jay Ou Xinming Katz Jonathan & Vigna Giovanni *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 1665–1680). New York, NY, USA: Association for Computing Machinery.

Castro, R. L., Schmitt, C., & Rodosek, G. D. (2019). Armed: How automatic malware modifications can evade static detection? In *2019 5th International Conference on Information Management (ICIM)*, (pp. 20–27).

Chen, P.-y., Kataria, G., & Krishnan, R. (2011). Correlated failures, diversification, and information security risk management. *MIS Quarterly*, *35*(2), 397–422.

Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., & Stoddart, K. (2016). A review of cyber security risk assessment methods for Scada systems. *Computers & Security*, *56*, 1–27.

Christey, S. & Martin, B. (2013, July). Buying into the bias: why vulnerability statistics suck. Retrieved from https://www.blackhat.com/us-13/archives.html#Martin.

Cox Jr, L. A. T. (2008). Some limitations of "risk = threat × vulnerability × consequence" for risk analysis of terrorist attacks. *Risk Analysis*, *28*(6), 1749–1761.

Dashevskyi, S., Brucker, A. D., & Massacci, F. (2018). A screening test for disclosed vulnerabilities in foss components. *IEEE Transactions on Software Engineering*, *45*(10), 945-966.

DeGroot, M. H. (2005). *Optimal statistical decisions* (Vol. *82*). New York: John Wiley & Sons.

Dey, D., Lahiri, A., & Zhang, G. (2015). Optimal policies for security patch management. *INFORMS Journal on Computing*, *27*(3), 462–477.

Do, C. T., Tran, N. H., Hong, C., Kamhoua, C. A., Kwiat, K. A., Blasch, E., … Iyengar, S. S. (2017). Game theory for cyber security and privacy. *ACM Computing Surveys (CSUR)*, *50*(2), 1–37.

Dolev, D. & Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, *29*(2), 198–208.

Dumitras, T. & Shou, D. (2011). Toward a standard benchmark for computer security research: The worldwide intelligence network environment (WINE). In Kirda Engin & Holz Thorsten *Proceedings of the First Workshop on Building*

*Analysis Datasets and Gathering Experience Returns for Security* (pp. 89–96). New York, NY, USA: Association for Computing Machinery.

First.org (2015). Common vulnerability scoring system v3.0: Specification document. Technical report, FIRST, Available at http://www.first.org/cvss, https://www.first.org/cvss/.

Franklin, J., Paxson, V., Perrig, A., & Savage, S. (2007). An inquiry into the nature and causes of the wealth of internet miscreants. In Ning Peng De Capitani di Vimercati Sabrina & Syverson Paul *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, (pp. 375–388). New York, NY, USA: Association for Computing Machinery.

Frederick, S., Loewenstein, G., & O'donoghue, T. (2002). Time discounting and time preference: A critical review. *Journal of economic literature*, *40*(2), 351–401.

Gisladottir, V., Ganin, A. A., Keisler, J. M., Kepner, J., & Linkov, I. (2017). Resilience of cyber systems with over- and underregulation. *Risk Analysis*, *37*(9), 1644–1651. https://doi.org/10.1111/risa.12729

Grier, C., Ballard, L., Caballero, J., Chachra, N., Dietrich, C. J., Levchenko, K., … Voelker, G. M. (2012). Manufacturing compromise: The emergence of exploit-as-a-service. In Yu Ting Danezis George & Gligor Virgil *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS'12)* (pp. 821–832). New York, NY, USA: Association for Computing Machinery.

Guikema, S., McLay, L., & Lambert, J. H. (2015). Infrastructure systems, risk analysis, and resilience: Research gaps and opportunities. *Risk Analysis*, *35*(4), 560–561. https://doi.org/10.1111/risa.12416

Herley, C. (2013). When does targeting make sense for an attacker?. *IEEE Security and Privacy*, *11*(2), 89–92.

Hewett, R., Rudrapattana, S., & Kijsanayothin, P. (2014). Cybersecurity analysis of smart grid scada systems with game models. In Abercrombie, Robert K. & McDonald, J. Todd *Proceedings of the 9th Annual Cyber and Information Security Research Conference* (pp. 109–112). New York, NY, USA: Association for Computing Machinery.

Homescu, A., Neisius, S., Larsen, P., Brunthaler, S., & Franz, M. (2013). Profile-guided automated software diversity. In Davidson, Jack, Eeckhout, Lieven & McKinley, Kathryn S. *2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)* (pp. 1–11). IEEE, ACM.

Huang, J., Borges, N., Bugiel, S., & Backes, M. (2019). Up-to-crash: Evaluating third-party library updatability on android. In *4th IEEE European Symposium on Security and Privacy (EuroS&P'19)*. IEEE.

Hutchings, A. & Clayton, R. (2016). Exploring the provision of online booter services. *Deviant Behavior*, *37*(10), 1163–1178. https://doi.org/10.1080/01639625.2016.1169829

Kannan, K., Rahman, M. S., & Tawarmalani, M. (2016). Economic and policy implications of restricted patch distribution. *Management Science*, *62*(11), 3161–3182.

Kotov, V. & Massacci, F. (2013). Anatomy of exploit kits. In Jürjens, J., Livshits, B., & Scandariato, R. (Eds.), *Engineering secure software and systems: 5th International Symposium, ESSoS 2013*, (pp. 181–196). Berlin, Heidelberg: Springer. 10.1007/978-3-642-36563-8_13 Berlin, DE: Springer, Berlin, Heidelberg.

Kotzias, P., Bilge, L., Vervier, P.-A., & Caballero, J. (2019). Mind your own business: A longitudinal study of threats and vulnerabilities in enterprises. In Trent Jaeger Alina Oprea & Dongyan Xu *Network and Distributed Systems Security (NDSS) Symposium 2019*. Reston, VA, USA: Internet Society.

Kuper, G., Massacci, F., Shim, W., & Williams, J. (2020). Who should pay for interdependent risk? Policy implications for security interdependence among airports. *Risk Analysis*, *40*(5), 1001–1019.

Lee, C. H., Geng, X., & Raghunathan, S. (2016). Mandatory standards and organizational information security. *Information Systems Research*, *27*(1), 70–86.

Manshaei, M. H., Zhu, Q., Alpcan, T., Bacşar, T., & Hubaux, J.-P. (2013). Game theory meets network security and privacy. *ACM Computing Surveys*, *45*(3), 25:1–25:39. https://doi.org/10.1145/2480741.2480742

Mell, P., Scarfone, K., & Romanosky, S. (2007). A complete guide to the common vulnerability scoring system version 2.0. In *Published by FIRST-forum of incident response and security teams* (Vol. *1*, p. 23).

Mitra, S. & Ransbotham, S. (2015). Information disclosure and the diffusion of information security attacks. *Information Systems Research*, *26*(3), 565–584.

Murdoch, S. J., Drimer, S., Anderson, R., & Bond, M. (2010). Chip and pin is broken. In Ulf Lindqvist David Evans & Giovanni Vigna *2010 IEEE symposium on security and privacy* (pp. 433–446). Oakland, CA, USA: IEEE.

Nappa, A., Johnson, R., Bilge, L., Caballero, J., & Dumitras, T. (2015). The attack of the clones: A study of the impact of shared code on vulnerability patching. In Sean Peisert Lujo Bauer & Vitaly Shmatikov *2015 IEEE Symposium on Security and Privacy* (pp. 692–708). San Jose, CA, USA: IEEE.

Nayak, K., Marino, D., Efstathopoulos, P., & Dumitraş, T. (2014). Some vulnerabilities are different than others. In A. Stavrou H. Bos & G. Portokalidis *Proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses* (pp. Lecture Notes in Computer Science, *8688*, 426–446). Springer, Cham.

Nguyen, V. H., Dashevskyi, S., & Massacci, F. (2015). An automatic method for assessing the versions affected by a vulnerability. *Empirical Software Engineering*, *21*, 2268–2297.

Nikiforakis, N., Maggi, F., Stringhini, G., Rafique, M. Z., Joosen, W., Kruegel, C., … Zanero, S. (2014). Stranger danger: exploring the ecosystem of ad-based url shortening services. In Chin-Wan Chung Andrei Broder & Kyuseok Shim *Proceedings of the 23rd International Conference on World Wide Web* (pp. 51–62). New York, NY, US: Association for Computing Machinery.

Okhravi, H. & Nicol, D. (2008). Evaluation of patch management strategies. *International Journal of Computational Intelligence: Theory and Practice*, *3*(2), 109–117.

Ooi, K. W., Kim, S. H., Wang, Q.-H., & Hui, K.-L. (2012). Do hackers seek variety? An empirical analysis of website defacements. In Ming-Hui Huang Gabe Piccoli & Vallabh Sambamurthy *Proceedings of the 33rd International Conference on Information Systems (ICIS 2012)*. AIS.

Paté-Cornell, E. (2012). On "black swans" and "perfect storms": Risk analysis and management when statistics are not enough. *Risk Analysis*, *32*(11), 1823–1833.

Provos, N., Mavrommatis, P., Rajab, M. A., & Monrose, F. (2008). All your iframes point to us. In Paul Van Oorschot *Proceedings of the 17th USENIX Security Symposium* (pp. 1–15). USENIX Association.

Ransbotham, S. & Mitra, S. (2009). Choice and chance: A conceptual model of paths to information security compromise. *Information Systems Research*, *20*(1), 121–139. https://doi.org/10.1287/isre.1080.0174

Rao, J. M. & Reiley, D. H. (2012, April). The economics of spam. *Journal of Economic Perspectives*, *26*(3), 87–110.

Rao, N. S. V., Poole, S. W., Ma, C. Y. T., He, F., Zhuang, J., & Yau, D. K. Y. (2016). Defense of cyber infrastructures against cyber-physical attacks using game-theoretic models. *Risk Analysis*, *36*(4), 694–710. https://doi.org/10.1111/risa.12362

Research, V. (2018). 2018 data breach investigation report. Technical report, Verizon, Retrieved from https://www.phishingbox.com/assets/files/images/Verizon-Data-Breach-Investigations-Report-2018.pdf.

Rios Insua, D., Rios, J., & Banks, D. (2009). Adversarial risk analysis. *Journal of the American Statistical Association*, *104*(486), 841–854.

Schneier, B. (2008). Inside the twisted mind of the security professional. *Wired Magazine*. https://www.schneier.com/essays/archives/2008/03/inside_the_twisted_m.html.

Serra, E., Jajodia, S., Pugliese, A., Rullo, A., & Subrahmanian, V. (2015a). Pareto-optimal adversarial defense of enterprise systems. *ACM Transactions on Information and System Security (TISSEC)*, *17*(3), 11.

Serra, E., Jajodia, S., Pugliese, A., Rullo, A., & Subrahmanian, V. S. (2015b). Pareto-optimal adversarial defense of enterprise systems. *ACM Transactions on Information and System Security*, *17*(3), 11:1–11:39. https://doi.org/10.1145/2699907

Smeets, M. (2018). A matter of time: On the transitory nature of cyberweapons. *Journal of Strategic Studies*, *41*(1-2), 6–32. https://doi.org/10.1080/01402390.2017.1288107

Stock, B., Lekies, S., & Johns, M. (2013). 25 million flows later-large-scale detection of dom-based xss. Ahmad-Reza Sadeghi Virgil Gligor & Moti Yung *20th CCS*. New York, NY, US: Association for Computing Machinery.

Stokey, N. L. (2008). *The economics of inaction: Stochastic control models with fixed costs*. Princeton, NJ: Princeton University Press.

Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., … Vigna, G. (2009). Your botnet is my botnet: Analysis of a botnet takeover. In Ehab Al-Shaer Somesh Jha & Angelos D. Keromytis *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*. New York, NY, US: Association for Computing Machinery.

van Dijk, M., Juels, A., Oprea, A., & Rivest, R. L. (2013). Flipit: The game of "stealthy takeover." *Journal of Cryptology*, *26*(4), 655–713. https://doi.org/10.1007/s00145-012-9134-5

Verizon (2011). 2011 data breach investigation report. Technical report, Verizon, Retrieved from https://www.wired.com/images_blogs/threatlevel/2011/04/Verizon-2011-DBIR_04-13-11.pdf.

Wash, R. (2010). Folk models of home computer security. In Lorrie Faith Cranor *Proceedings of the Sixth Symposium on Usable Privacy and Security*. New York, NY, US: Association for Computing Machinery.

Zhang, J. & Zhuang, J. (2019). Modeling a multi-target attacker-defender game with multiple attack types. *Reliability Engineering & System Safety*, *185*, 465–475.

## APPENDIX A: INTERNET EXPLORER EXPLOIT CODE

Here, we report the exploit code for $\texttt{CVE}_1$ =CVE-2010-0806 and $\texttt{CVE}_2$ =CVE-2009-3672 both affecting Internet Explorer version 7. On Exploit-DB[5] $\texttt{CVE}_1$ corresponds to exploit 16547 and $\texttt{CVE}_2$ corresponds to exploit 11683.

[5]See Exploit-DB (http://www.exploit-db.com, last accessed April 12, 2018.)

The exploit code fragments (10+ lines of code out of 260+ for $\texttt{CVE}_1$ and 130+ for $\texttt{CVE}_2$) below illustrates the difference between the two exploits as scripted for the Metasploit engine. The exploit code for $\texttt{CVE}_2$ is effectively a rearrangement of the exploit code for $\texttt{CVE}_1$, with different variable names (e.g. by replacing `j_memory` with `var_memory`, `j_shellcode` with `var_shellcode`) and repositioned at the appropriate memory addresses (0x …).

```
function j_function1()
2 ...
3 j_memory = new Array();
4 var j_shellcode = unescape(...);
5 var j_slackspace = 0x86000 -
(j_shellcode.length*2);
6 while(j_nops.length< j_slackspace/2)
7 j_nops+=j_nops;
8 for(j_counter=0; j_counter<270;
j_counter++)
9 j_memory[j_counter] = j_fillblock +
j_fillblock + j_shellcode;

function var_body()
2 ...
3 var var_memory = new Array();
4 var var_shellcode = var_unescape(..);
5 var var_ss = 20 + var_shellcode.length;
6 while (var_spray.length < var_ss)
7 var_spray+=var_spray;
8 var_bk = var_spray.substring(..);
9 while(var_bk.length+var_ss < 0x100000)
10 var_bk = var_bk + var_bk + var_fb;
11 for (var_i=0; var_i<1285; var_i++)
12 var_memory[var_i]= var_bk +
var_shellcode;
```

## SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Online Appendix