Toward a Theory of Vulnerability Disclosure Policy: A Hacker's Game[*]

Taylor J. Canann

CEPA, McCombs School of Business, The University of Texas at Austin, Austin TX 78712, USA
taylor.canann@mccombs.utexas.edu

Abstract. A game between software vendors, heterogeneous software users, and a hacker is introduced in which software vendors attempt to protect software users by releasing updates, i.e. disclosing a vulnerability, and the hacker is attempting to exploit vulnerabilities in the software package to attack the software users. The software users must determine whether the protection offered by the update outweighs the cost of installing the update. Following the model is a description of why the disclosure of vulnerabilities can only be an optimal policy when the cost to the hacker of searching for a Zero-Day vulnerability is small. The model is also extended to discuss Microsoft's new "extended support" disclosure policy.

Keywords: Game Theory · Welfare Analysis · Vulnerability Disclosure Policy

1 Introduction

In May of 2017 the WannaCry attacks infected over 300,000 systems in 150 countries and the approximate estimated cost that these attacks is $4 billion. One month later, the NotPetya attacks, another major global attack that primarily targeted Ukrainian systems, began. The approximated costs of the NotPetya attacks were even larger than the WannaCry attacks and have been estimated at around $10 billion. Following the NotPetya attacks, the Retefe banking Trojan began leveraging the EternalBlue exploit in September. Finally, in August of 2018 the Taiwan Semiconductor Manufacturing Company, an Apple chip supplier, was hit by a new variant of the WannaCry attack that cost the company approximately $170 million. The problem was not that Windows is an inherently flawed system, but instead that these attacks could have been avoided if users/firms had only updated. In March of 2017, Microsoft patched this vulnerability in their monthly, second Tuesday, update.

This is not just a problem with Microsoft software, every piece of software, no matter what care is taken by a software vendor, is riddled with vulnerabilities, which leaves users of the software open to attack by hackers. To protect users, software vendors release patches to address these found vulnerabilities, but this is a double-edged sword. Releasing updates, a.k.a. vulnerability disclosure, may in fact increase the susceptibility of current users to attack, in particular, those who chose not to immediately install the updates. This is due to the fact that the update can be reverse engineered quite easily by hackers. These types of hacks have been gaining in prevalence over the last few of years.

[*]I am grateful to Richard Evans, Kerk Phillips, the BYU MCL workshops, Brennan Platt, Brad Greenwood, Robert Mrkonich, Samuel Kaplan, Kenneth Judd, Chase Coleman, Ryne Belliston, Jan Werner, David Rahman, and Aldo Rustichini for very helpful comments and to Alexander Pingry for excellent research assistance. Additional comments and proofs can be found in the online mathematical appendix.

In a seminal paper in the field of vulnerability disclosure, [1] asked if finding vulnerabilities is optimal for social welfare. Since then, vulnerability disclosure policy has been greatly debated in the literature. The model outlined in this paper explores the decisions made by both the network of users and a hacker given the different policy regimes that could be implemented by the vendor. The interaction between vendors and software users was first modeled by [2], in which they find that vendors will always want to delay the release of patches, but this action is not socially optimal. However, [2] do not pose an answer to whether a vendor should engage in disclosing vulnerabilities, which is the main focus of this paper.

One of the first papers on the economic modeling of hacker behavior was developed in [3], where they attempt to estimate the effects of the fixed costs of hacking on the incentives of a profit maximizing hacker. Much of the recent literature that has attempted to model hacker behaviour, e.g. see [4], follow models similar to the Becker model of criminal behavior ([5]), but this approach assumes that: (i) Law enforcement can easily track and find a hacker and (ii) Hackers can easily be prosecuted. These assumptions are the exception, not the rule . To break from this convention, the hacker is modeled as a profit maximizing agent in order to contribute hacker behavior into the vulnerability disclosure debate.

Both the network framework of software users and the hacker behaving as a profit maximizing agent are extensions of the work in [6], where they focus on the welfare effects of disclosure policy for a representative set of users with the vendor facing a monopolistically competitive market. This paper follows the notation in [6] rather closely so as to maintain a consistent notational scheme within the vulnerability disclosure literature.

Others have examined how attack propensity changes under different disclosure regimes (e.g. [7]), and have found that releasing patches tends to increase the number of attacks. This model identifies the reasons for this observed increase in attacks as being driven by the decisions of both the hacker and the software users. The hacker's decision is driven by parameters such as the probability of a successful attack, as well as the costs associated of finding new vulnerabilities in the software package. The software users must balance the value they place on using the software relative to the expected damages of an attack and the cost of updating their machine. Therefore, this model is able to give a causal relationship between attack propensity and disclosure regimes which strengthen the story behind these correlations.

In order to describe the best type of disclosure policy, a model of a heterogeneous network made up of an interconnected set of software users that are attempting to defend themselves against a profit-maximizing hacker is developed in this paper. Within my model, there are three decisions to be made: (i) The strategy of attack to be played by the hacker, (ii) The optimal disclosure policy, and (iii) The updating decision made by the software user. Following the model setup, welfare maximizing policies are formulated to decrease a hacker's efforts in infiltrating networks and increase the software users' utility.

The optimal strategy for the software vendor, the optimal policy, is to maximize the egalitarian sum of utilities of the software users, i.e. the vendor acts as a social planner. The optimal policy is dependent both on the distribution of software users on the network and how costly finding a previously unknown vulnerability, i.e. a Zero-Day vulnerability, is for the hacker. Software users that do not expect to bear the majority of the burden of an attack, known as low-type users, do not want vulnerabilities to be disclosed, i.e. a Non-Disclosure policy, since they will not update their machines, deeming it too costly. Also, if the cost of searching for a Zero-Day exploit is higher than the expected payoff of the exploit, then the hacker is not willing to expend the energy searching for a Zero-Day, and Non-Disclosure is the optimal policy. Therefore, the only case in which Disclosure can be an optimal policy is when search costs are low and there are enough users that desire to update their machines.

Starting in January of 2020, Microsoft will no longer support Windows 7, unless the users enroll in Extended Support[1]. The final result of the paper is that Microsoft's new policy increases the cost of exploiting the disclosed vulnerability, and, even though the policy increases the cost of updating, under certain parameter values the policy causes the software uses to receive higher payoffs. This new approach to disclosure policy can increase the overall welfare relative to the policy of disclosing all vulnerabilities.

The sections of the paper are as follows: the model is introduced as well as the first main contribution: A discussion of optimal policy when the hacker are decision making agents in Section 2. Following is the newly proposed policy by Microsoft in Section 4, then concluding is in Section 5.

2 Static Game

The players within this static game are the software vendor, a hacker, and the software users. The software vendor follows a welfare maximizing disclosure policy, and thus determines the rules of the game. The hacker maximizes his profits by choosing a hacking strategy of exploiting either a Zero-Day, the patch released by the software vendor, i.e. an N-Day attack, or he can exit the game. Lastly, each software user must decide whether or not to update her machine if a vulnerability is disclosed, i.e. an update is released.

The vendor of the software package is only concerned with maximizing software user welfare in an egalitarian manner, similar to a social planner. The vendor is unable to detect all vulnerabilities

Table 1: Notations used in the paper

α	Probability the software vendor finds a vulnerability, $\alpha \in (0,1)$
$D,\ ND$	Disclosure or Non-Disclosure policy, respectively
$I = \{1, \dots, m\}$	Set of interconnected software users
θ	Set of software user weights, $\theta = \{\theta_1, \dots, \theta_m\}$
θ_i	i^{th} software user's weight, $0 < \theta_1 \leq \theta_2 \leq \cdots \leq \theta_m < 1$
$\theta_i v$	Value obtained by the i^{th} software user for using the software package, $v > 0$
$\theta_i D$	Damage done by the hacker to the i^{th} software user via a hack, $D > 0$
c_u	Opportunity cost of updating, $c_u > 0$
$u,\ nu$	Software user choice of whether to update or not update, respectively
$\delta,\ \widehat{\delta}$	Probability the hacker successfully finds a Zero-Day under D and ND, respectively, $\delta > \widehat{\delta}$
c_s	Opportunity cost of searching for a Zero-Day, $c_s > 0$
$E,\ S,\ X$	Hacker choice to (E)xploit the N-Day, (S)earch for a Zero-Day, or e(X)it the game

[1]See `https://www.microsoft.com/en-us/microsoft-365/blog/2018/09/06/helping-customers-shift-to-a-modern-desktop/`

before selling the software, but the vendor, under a Disclosure policy, will attempt to find these vulnerabilities ex post. The probability of finding a vulnerability, α, can either be thought of as individual vendors searching for vulnerabilities by themselves or as bounty systems such as Microsoft's Bounty System (E.g. See: [8], [9], [10], and [11]).

The software user's weight parameter in Table 1 can be thought of as the network centrality of the software user, or as how desirable the information found on the software user's machine is to the hacker. Due to the hacker attempting to exploit the network and the inability of vendors to solve all vulnerabilities ex ante, each software user is vulnerable to an attack. The hacker is only able to extract as much information as is available to software user i. This damage can also be thought of a direct transfer from the software user to the hacker when the software user is hacked.

The vendor does not usually charge the software users to install the updates, but the updates are still costly in terms of opportunity costs, i.e. the time to install the update. Updates often require software users to stop working or even shutdown their machines, thus $c_u > 0$. For simplicity, this cost is assumed to be a fixed cost to be paid if the software user decides to update. To model the fact that some people do not update under any policy and there exists at least one software user that might update, the following assumption is made:

Assumption 1 *Let* $\theta_1 < \frac{c_u}{v+D} < \theta_m$.

A single hacker attempts to exploit vulnerabilities to maximize profits via gaining access to the network of software users. The hacker's profits function is dependent on both the chosen policy and the software users' optimal updating decision. The hacker has two types of exploitation available to them, he is able to hack via a known vulnerability, an N-Day exploit, or by a previously unknown vulnerability, a Zero-Day attack. The information available to the hacker consists of the policy played by the vendor, the distribution of software user weights, the strategies available to the software users, and the probability of a successful Zero-Day attack.

Hacking, however, is not cost-less. A constant search cost, or opportunity cost of searching for a Zero-Day, is assumed. If the hacker decides to exploit a known vulnerability, meaning to attack vulnerability that was just patched by the vendor, then the hacker's cost of hacking is assumed to be zero. This is to account for the relative ease of reverse engineering an update to find the vulnerability in the code.

Under a Non-Disclosure regime, the hacker is only able to search for Zero-Day exploits or exit the game; while the software user makes no decision under this regime. If policy dictates that a Disclosure regime is optimal, then the hacker can still search for Zero-Day exploits or exit the game as in the Non-Disclosure regime, but he can also choose to exploit the updated vulnerability on all machines that have not had the patch installed. Given a disclosed vulnerability, each software user must decide whether to install the update on her machine.

2.1 Non-Disclosure Regime

If the vendor chooses ND, or is forced to withhold this information, then software users do not make any decisions, they just use the software to gain, at most, $v\theta_i$. The hacker's action set is defined as $A_{nd} \in \{S,\ X\}$. The utility payoff of software user i, $U^i_{nd} : A_{nd} \times \theta_i \to \mathbb{R}$, is equal to $v\theta_i$ if they are not exploited or $-D\theta_i$ if the hacker was successful in finding a Zero-Day.

Via Table 2, the hacker will only choose (S) if the cost of searching for a Zero-Day is low, i.e. the expected payoff is positive. If $c_s < \delta D \sum_{i \in I} \theta_i$, then the Nash Equilibrium is that hacker will

Table 2: Hacker Expected Payoff Functions: Non-Disclosure

Hacker Action	Payoff
(S)	$\mathit{\Pi}_S^{ND}(\theta) = \delta(n)\left[D\sum_i \theta_i\right] - c_s.$
(X)	$\mathit{\Pi}_X^{ND}(\theta) = 0$

search for Zero-Days, $A_{nd}^* = (S)$. However, if $c_s > \delta D \sum_{i \in I} \theta_i$, then under Non-Disclosure, the unique Nash equilibrium is to exit the game, $A_{nd}^* = (X)$.

2.2 Disclosure Regime

If the vendor chooses D, the vendor releases updates every time they find a vulnerability. Each software user then must choose whether to update, and thus endogenously define the two sets $\mathit{\Gamma}_{nu}$ and $\mathit{\Gamma}_u$ as the set of software users that do not update and the set of software users that do update, respectively, and $\xi = |\mathit{\Gamma}_u|$, the number of software users that update under a Disclosure policy,. When a software user chooses to update, she protects her machine from N-Day exploits, but is still vulnerable to Zero-Days. Due to the costly nature of updating, some software users may choose not to update leaving their computers open to both Zero-Day and N-Day hacks (E.g. See [12]).

Now there are two stages within the game, the first being the possible release of updates by the vendor, which happen with probability α, followed by the game between the hacker and the software users. When the vendor is unable to find a vulnerability, the game is identical to that of the Non-Disclosure regime in Section 2.1. The hacker's action set when the vendor is unable to find a vulnerability within the Disclosure regime is denoted as $A_d^{1-\alpha} \in \{S,\ X\}$. When the vendor finds a vulnerability and releases an update, then both the hacker and the software user must choose their actions, $A_d^{\alpha} \in \{E,\ S,\ X\}$ and $A^i \in \{u,\ nu\}$, respectively.

The expected utility of the software user i is defined as the function $U_d^i : A_d^{\alpha} \times A_d^{1-\alpha} \times A^i \times \theta_i \to \mathbb{R}$, where she receives $v\theta_i$ if her machine is not exploited, $-D\theta_i$ if the hacker is successful in attacking her machine, and $-c_u$ if she decides to update.

Table 3: Hacker Expected Payoff Functions: Disclosure

Hacker Action	Payoff
$(E,\ S)$	$\mathit{\Pi}_{(E,S)}^D(\theta, \{\mathit{\Gamma}_u, \mathit{\Gamma}_{nu}\}) = \alpha\left[D\sum_{i \in \mathit{\Gamma}_{nu}} \theta_i\right] + (1-\alpha)\left[\delta D \sum_i \theta_i - c_s\right]$
$(E,\ X)$	$\mathit{\Pi}_{(E,X)}^D(\theta, \{\mathit{\Gamma}_u, \mathit{\Gamma}_{nu}\}) = \alpha\left[D\sum_{i \in \mathit{\Gamma}_{nu}} \theta_i\right]$
$(S,\ S)$	$\mathit{\Pi}_{(S,S)}^D(\theta, \{\mathit{\Gamma}_u, \mathit{\Gamma}_{nu}\}) = \left[\alpha\widehat{\delta} + (1-\alpha)\delta\right] D\sum_i \theta_i - c_s$
$(S,\ X)$	$\mathit{\Pi}_{(S,X)}^D(\theta, \{\mathit{\Gamma}_u, \mathit{\Gamma}_{nu}\}) = \alpha\left[\widehat{\delta} D\sum_i \theta_i - c_s\right]$
$(X,\ S)$	$\mathit{\Pi}_{(X,S)}^D(\theta, \{\mathit{\Gamma}_u, \mathit{\Gamma}_{nu}\}) = (1-\alpha)\left[\delta D\sum_i \theta_i - c_s\right]$
$(X,\ X)$	$\mathit{\Pi}_{(X,X)}^D(\theta, \{\mathit{\Gamma}_u, \mathit{\Gamma}_{nu}\}) = 0$

There are three main drivers of the Nash equilibria under Disclosure:

(a) Do there exist any software users that choose not to update when an update is released? (Notice that this is always satisfied via Assumption 1.)

(b) If there is no update released, does the cost of finding a Zero-Day exceed the expected profits of searching? I.e.

$$c_s \lesseqgtr \delta D \sum_{i \in I} \theta_i. \tag{1}$$

(c) If a vulnerability is disclosed, does the cost of finding a Zero-Day exceed the expected profits of searching? I.e.

$$c_s \lesseqgtr \widehat{\delta} D \sum_{i \in I} \theta_i. \tag{2}$$

The first case to examine is when the cost of searching is high, i.e. $c_s > \delta D \sum_i \theta_i$. Since both the hacker and the software users know whether an update has been released, then the solution can be split into the Non-Disclosure and the Disclosure sub-games. Similar to the Non-Disclosure case when there are high search costs, in the Disclosure game when no vulnerability is found $A_d^{1-\alpha *} = (X)$ is the equilibrium of the sub-game.

Since the search costs are high for the hacker and there exists at least one software user that does not update, then $A_d^{\alpha *} = (E)$ is the only strategy to survive elimination of strictly dominant strategies for the hacker, and is thus the only strategy in the best response for the hacker. Given the hacker strategy (E), the best response of software user i is to not update, i.e. $i \in \Gamma_{nu}^*$, if $\theta_i < \frac{c_u}{v+D}$. Otherwise, for software user j such that $\theta_j > \frac{c_u}{v+D}$, updating is optimal, $j \in \Gamma_u^*$. If $\theta_i = \frac{c_u}{v+D}$, then she is indifferent between any mixture $p_j \in [0, 1]$ of *Update* and *Not Update*.

Therefore, the Nash equilibrium of the Disclosure game under high search costs is

$$((A_d^{\alpha *}, A_d^{(1-\alpha)*}), (A_i^*)_{i \in I}) = ((E, X), (nu)_{i \in \Gamma_{nu}^*}, (u)_{j \in \Gamma_u^*}) \tag{3}$$

Where $\Gamma_{nu}^* = \left\{ i \in I | \theta_i < \frac{c_u}{v+D} \right\}$ and $\Gamma_u^* = \left\{ j \in I | \theta_j > \frac{c_u}{v+D} \right\}$.

The next case, denoted the medium search cost case, is when searching is profitable when the vendor is unable to find the vulnerability but not when the vulnerability is disclosed by the vendor, i.e. $\widehat{\delta} D \sum_{i \in I} \theta_i \leq c_s < \delta D \sum_{i \in I} \theta_i$. If the vendor is unable to find a vulnerability, the cost of searching is still exceeded by the expected profits of searching, and thus $A_d^{(1-\alpha)*} = (S)$ is his best response. However, when the vendor finds a vulnerability, then the expected profits of (S) are surpassed by the cost of (S), then the action of (S) when a vulnerability is disclosed yields a strictly lower payoff then (X). Since there always exist software users that do not update, then the best action for the hacker to play is $A_d^{\alpha *} = (E)$.

Then, notice that all software users such that $\theta_i < \frac{c_u}{v+D}$ will be in Γ_{nu}^*, and all software users $\theta_j > \frac{c_u}{v+D}$ will be in Γ_u^*. Therefore, the Nash equilibrium of the medium search cost case is

$$((A_d^{\alpha *}, A_d^{(1-\alpha)*}), (A_i^*)_{i \in I}) = ((E, S), (nu)_{i \in \Gamma_{nu}^*}, (u)_{j \in \Gamma_u^*}) \tag{4}$$

Where $\Gamma_{nu}^* = \left\{ i \in I | \theta_i < \frac{c_u}{v+D} \right\}$ and $\Gamma_u^* = \left\{ j \in I | \theta_j > \frac{c_u}{v+D} \right\}$.

The final case is to determine what happens when searching yields positive profits under both branches of the game, i.e. $c_s < \hat{\delta} D \sum_i \theta_i$. In this low search cost case , with probability $1 - \alpha$, we obtain the same solution as in the Non-Disclosure game in Section 2.1, i.e. $A_d^{(1-\alpha)*} = (S)$.

Next is to determine the best response of both the hacker and each software user when an update is released. The first thing to notice is that (X) is never a best response since exiting gives a payoff of zero while (S) and (E) both yield positive expected payoffs. Given the hacker strategy (E), $i \in \Gamma^*_{nu}$, is the software user i's best response so long as $\theta_i < \frac{c_u}{v+D}$.

If $\theta_j > \frac{cu}{v+D}$, then software user j's best response is $j \in \Gamma^*_u$. Whenever the hacker plays (S), updating will not protect the software user from a hack, and thus, $i \in \Gamma^*_{nu}$ is the best response for all $i \in I$.

Allowing for the hacker to use mixed-strategies introduces the probability $\rho \in (0,1)$, where ρ is the probability that the hacker chooses (E) and $(1-\rho)$ gives (S). Using the expected payoffs of the software users given ρ, then any software user i's best response is $i \in \Gamma^*_{nu}$ when $\theta_i < \frac{c_u}{\rho(v+D)}$. Notice that for any $\rho \in [0, \frac{c_u}{\theta_m(v+D)})$, (nu) is the best response for all software users. For all software users j such that $\theta_j > \frac{c_u}{\rho(v+D)}$, $j \in \Gamma^*_u$ is their optimal action. For any software user k such that $\theta_k = \frac{c_u}{\rho(v+D)}$, the software user is indifferent between updating and not updating, and will mix with probability $p_k \in [0,1]$, where p_k is the probability of choosing (u).

Now to examine the best response of the hacker when a vulnerability is disclosed given the software users' strategies. If all of the software users update, i.e. $\Gamma_u = I$, then the best response for the hacker is $A^{\alpha*}_d = (S)$. Similarly, if the software user strategy is $\Gamma_{nu} = I$, then $A^{\alpha*}_d = (E)$ is the only strategy in the best response for the hacker.

Due to the monotonicity of the software users, and thus their optimal actions, all that is left to do is to split I between high- and low-type users. Define $\Omega \equiv \left\{ j \in I | \theta_j \geq \frac{c_u}{v+D} \right\}$ as the set of high-type software users, i.e. the users that will update if the hacker chooses (E). For some $k \in \Omega$, define $\Gamma^k_{nu} = \{i \in I | \theta_i < \theta_k\}$ and $\Gamma^k_u = \{j \in I | \theta_j > \theta_k\}$. Given a software user strategy of $(\Gamma^k_{nu}, (p_k(u), (1-p_k)(nu)), \Gamma^k_u)$, for some mixed strategy $p_k \in [0,1]$ for software user k, then the hacker's expected payoff of mixing with $\rho \in [0,1]$ between exploiting and searching is

$$\rho\left[D \sum_{i \in \Gamma^k_{nu}} \theta_i + (1-p_k) D\theta_k\right] + (1-\rho)\left[\widehat{\delta} D \sum_{i \in I} \theta_i - c_s\right] \tag{5}$$

For all $\rho \in [0,1]$, if

$$c_s > \widehat{\delta} D \sum_{i \in I} \theta_i - D \sum_{i \in \Gamma^k_{nu}} \theta_i - \big(1-p_k\big) D\theta_k \tag{6}$$

then $\rho^* = 1$ is the best response for the hacker given the software users' strategy.

However, if for every value $\rho \in [0,1]$,

$$c_s < \widehat{\delta} D \sum_{i \in I} \theta_i - D \sum_{i \in \Gamma^k_{nu}} \theta_i - \big(1-p_k\big) D\theta_k \tag{7}$$

then the hacker will send ρ^* to zero.

The last case is if there exists a $p_k \in [0,1]$ such that Inequality 6 holds with equality, i.e.

$$c_s = \widehat{\delta} D \sum_{i \in I} \theta_i - D \sum_{i \in \Gamma^k_{nu}} \theta_i - \big(1-p_k\big) D\theta_k \tag{8}$$

then any $\rho^* \in [0,1]$ is the hacker's best response to the software users' strategy of $(\Gamma^k_{nu}, (p_k(u), (1-p_k)(nu)), \Gamma^k_u)$.

Theorem 1 *Let $k_{min} \in \Omega$ be the minimal software user in Ω. If Inequality 6 holds for $p_{k_{min}} = 1$, then the Nash Equilibrium is*

$$((A_d^{\alpha *}, A_d^{(1-\alpha)*}), (A_i^*)_{i \in I}) = ((E, S), (nu)_{i \in \Gamma_{nu}^*}, (u)_{j \in \Gamma_u^*}) \tag{9}$$

Where $\Gamma_{nu}^* = \left\{ i \in I | \theta_i < \frac{c_u}{v+D} \right\}$ and $\Gamma_u^* = \left\{ i \in I | \theta_i > \frac{c_u}{v+D} \right\}$.

Otherwise, there exists a pivotal software user, $k^* \in \Omega$, and a mixed strategy for software user k^*, $p_{k^*}^* \in [0, 1]$, such that Equation 8 holds, and the Nash equilibrium is

$$((A_d^{\alpha *}, A_d^{(1-\alpha)*}), (A_i^*)_{i \in I}) = ((\rho^*(E, S), (1-\rho^*)(S, S)), (nu)_{i \in \Gamma_{nu}^{k*}}, (p_{k^*}^*(u), (1-p_{k^*}^*)(nu)), (u)_{j \in \Gamma_u^{k*}}) \tag{10}$$

Where $\rho^* = \frac{c_u}{\theta_{k^*}(v+D)}$, $\Gamma_{nu}^{k*} = \{ i \in I | \theta_i < \theta_{k^*} \}$, and $\Gamma_u^{k*} a = \{ i \in I | \theta_i > \theta_{k^*} \}$.

3 Welfare Analysis

In this section, the "Optimal Disclosure Policy" is first defined followed by solving for the optimal policy for each of the different search cost scenarios found in Section 2.

Definition 1. *The optimal policy $\Psi^* \in \{Disclosure,\ Non-Disclosure\}$ is chosen such that:*

$$\Psi^* = argmax_{\psi \in \{d,\ nd\}} \left\{ \sum_{i \in I} U_d(A_d^{\alpha *}, A_d^{(1-\alpha)*}, A_i^*, \theta_i),\ \sum_{i \in I} U_{nd}(A_{nd}^*, \theta_i) \right\} \tag{11}$$

Where $((A_d^{\alpha *}, A_d^{(1-\alpha)*}), (A_i^*)_{i \in I})$ and (A_{nd}^*) are the Nash equilibria under Disclosure and Non-Disclosure, respectively.

Under High Search Costs, recall that in the Nash equilibrium the hacker chooses to exploit the N-Day under Disclosure and to exit the game under Non-Disclosure. Under Disclosure, all low-type software users, the software users in Γ_{nu}^*, are hacked if a vulnerability is found; while all other software users must pay the cost of updating, which is assumed to be strictly greater than zero. Under Non-Disclosure, the hacker exits the game, and all software users obtain $\theta_i v$. Then the optimal policy is Non-Disclosure.

If $\widehat{\delta} D \sum_{i \in I} \theta_i \leq c_s < \delta D \sum_{i \in I} \theta_i$, i.e. the medium search cost case, then under a Non-Disclosure regime the hacker searches for a Zero-Day. However, under Disclosure, the hacker chooses to exploit the released vulnerability. Then, solving for the optimal policy is dependent on

$$\sum_{i \in \Gamma_{nu}^*} \theta_i + \xi^* \frac{c_u}{v+D} \lessgtr \delta \sum_{i \in I} \theta_i \tag{12}$$

As the expected losses from a Zero-Day exceed the cost of the low type software users being hacked since they did not update and the cost of updating for all $\xi^* = |\Gamma_u^*|$ of the high type software users, then Disclosure is the optimal policy. Thus, the optimal policy under medium search costs is Disclosure if $\sum_{i \in \Gamma_{nu}^*} \theta_i + \xi^* \frac{c_u}{v+D} < \delta \sum_{i \in I} \theta_i$. Otherwise, the optimal policy is Non-Disclosure if $\sum_{i \in \Gamma_{nu}^*} \theta_i + \xi^* \frac{c_u}{v+D} > \delta \sum_{i \in I} \theta_i$.

The last case to examine is that of low search costs. Recall that the Nash equilibrium of the Non-Disclosure game is $A_d^{(1-\alpha)*} = (S)$, while the Nash equilibrium of the Disclosure game takes the form of mixing between (E) and (S) for the hacker while the software users split into $(\mathit{\Gamma}_{nu}^{k*}, (p_k^*(u), (1-p_k^*)(nu)), \mathit{\Gamma}_u^{k*})$. The analysis begins with the optimal policy for all low-type software users, followed by the optimal policy for all high-type software users. To conclude the section the combined results of both high- and low-type software users are used to find the optimal policy.

For all software users $i \in \mathit{\Gamma}_{nu}^{k*}$, then we are able to analyze which policy they would prefer by solving

$$-\delta D \sum_{i \in \mathit{\Gamma}_{nu}^{k*}} \theta_i + (1-\delta)\, v \sum_{i \in \mathit{\Gamma}_{nu}^{k*}} \theta_i \lesseqgtr -\rho^* D \sum_{i \in \mathit{\Gamma}_{nu}^{k*}} + (1-\rho^*) \left[-\widehat{\delta} D \sum_{i \in \mathit{\Gamma}_{nu}^{k*}} \theta_i + (1-\widehat{\delta}) v \sum_{i \in \mathit{\Gamma}_{nu}^{k*}} \theta_i \right] \tag{13}$$

Disclosure is the optimal policy for all software users that do not update so long as

$$\begin{aligned} \rho^* &< \frac{\left[-\widehat{\delta} D + (1-\widehat{\delta}) v\right] - \left[-\delta D + (1-\delta) v\right]}{(1-\widehat{\delta})(v+D)} \\ \iff \rho^* &< \frac{\delta - \widehat{\delta}}{1-\widehat{\delta}} \end{aligned} \tag{14}$$

Notice that both the left-hand side and the right-hand side are strictly positive. Thus, the software users that do not update, software users in $\mathit{\Gamma}_{nu}^{k*}$, will sometimes want the policy to be Disclosure.

High-type software users, $j \in \mathit{\Gamma}_u^{k*}$, then face the welfare decision of

$$\begin{aligned} -\delta D \sum_{j \in \mathit{\Gamma}_u^{k*}} \theta_j + (1-\delta)\, v \sum_{j \in \mathit{\Gamma}_u^{k*}} \theta_j \lesseqgtr & \rho^* \left(v \sum_{j \in \mathit{\Gamma}_u^{k*}} \theta_j - \xi c_u \right) \\ & + (1-\rho^*) \left[-\widehat{\delta} D \sum_{j \in \mathit{\Gamma}_u^{k*}} \theta_j + (1-\widehat{\delta}) v \sum_{j \in \mathit{\Gamma}_u^{k*}} \theta_j - \xi^* c_u \right] \end{aligned} \tag{15}$$

For software users $i \in \mathit{\Gamma}_{nu}^{k*}$, Disclosure decreases the probability of being hacked by a Zero-Day, but it also increases their probability of being hacked since the hacker can exploit the N-Day vulnerability that these software users are not willing to defend against. However, software users $j \in \mathit{\Gamma}_u^{k*}$ are more likely to want a Disclosure regime since they both obtain the benefit of hackers having less vulnerabilities to search over as well as protection from the N-Day exploits since they will sometimes update.

Now to examine the welfare over all the software users by comparing the sum of all software users' utility functions. The optimal policy condition can be written as

$$\sum_{i \in \mathit{\Gamma}_{nu}^{k*}} \theta_i + \left(\frac{D}{v+D} - \widehat{\delta} + \xi^* \right) \theta_k \lesseqgtr \left(\frac{\delta - (1-\rho^*)\widehat{\delta}}{\rho^*} \right) \sum_{i \in I} \theta_i \tag{16}$$

Hence, the optimal policy under low search costs is Disclosure if $\sum_{i \in \mathit{\Gamma}_{nu}^{k*}} \theta_i + \left(\frac{D}{v+D} - \widehat{\delta} + \xi^* \right) \theta_k < \left(\frac{\delta - (1-\rho^*)\widehat{\delta}}{\rho^*} \right) \sum_{i \in I} \theta_i$, or the optimal policy is Non-Disclosure if $\sum_{i \in \mathit{\Gamma}_{nu}^{k*}} \theta_i + \left(\frac{D}{v+D} - \widehat{\delta} + \xi^* \right) \theta_k > \left(\frac{\delta - (1-\rho^*)\widehat{\delta}}{\rho^*} \right) \sum_{i \in I} \theta_i$.

4 Microsoft's "Extended Support"

This section contains an analysis of the forthcoming change to Microsoft 7 and 10's updating procedures and how this change alters the game described in Sections 2 and 3. The game is altered such that the software vendor, Microsoft, introduces a new monthly charge to receive updates. Microsoft intends to implement this policy starting on January 14th, 2020, which is the same day that Windows 7 will no longer be supported. But with a large number of Windows users still using Windows 7, Microsoft needed to come up with a policy to protect these users and maintain their market share.

Table 4: New notation for Microsoft's "Extended Support"

ϕ_u	New service charge paid by the software user and hacker to access the update (also called the exploitation cost), $\phi_u > 0$
c_v	Cost to switch to the new version of the software package, $c_v > 0$
v	Software user's choice to install the new version of the software package

Updating is no longer the only available choice to the software user. The software user can also choose to shift toward using a different version, i.e. Windows 10, for which the software user must pay a cost $c_v > 0$. If the software user shifts toward using the new version of the software, then the hacker is not able to attack the software user, not even via Zero-Days.

Assumption 2 *Let* $\frac{c_v}{\delta(v+D)} \in (\theta_1, \theta_m)$.

If the hacker wants to gain access to the disclosure of the vulnerability, the hacker must pay the subscription fee for the "Extended Support", ϕ_u. However, the hacker does not have to pay c_u since the hacker could easily enroll an old computer in the updating scheme in order to be notified of vulnerabilities. Consequently, the cost of exploiting N-Days has increased since $\phi_u > 0$. To be clear, Microsoft's new policy is fascinating since it has the potential to increase the cost of exploiting N-Days while also decreasing the effectiveness of Zero-Days against Windows 7.

Following the notation of the game in Section 2, this new policy can be explicitly defined. The first case to be described is the Non-Disclosure regime. The vendor was unable to find a vulnerability, and thus the hacker is only able to an action $A_M^{(1-\alpha)} \in \{S,\ X\}$. Searching for a Zero-Day is not as effective as in the above games due to the fact that software users are now able to change their software version to avoid being attacked. The software user choice is an action $A_{M,i}^{(1-\alpha)} \in \{v,\ nu\}$. The utility of software user i, $U^i_{M;nd} : A_M^{(1-\alpha)} \times A_{M,i}^{(1-\alpha)} \times \theta_i$. All players that use the old software are contained in $\mathit{\Gamma}_{nu}$, and all software users that switch versions are in $\mathit{\Gamma}_v$.

The next step is to formalize the Disclosure sub-game. The hacker has the same set of actions in this case as in the Disclosure case above to pick from: $A_M^{\alpha} \in \{E,\ S,\ X\}$. The action set for the software users is now $A_{M,i}^{\alpha} \in \{v,\ u,\ nu\}$. The utility of software user i is now $U^i_{M;d} : A_M^{\alpha} \times A_{M,i}^{\alpha} \times \theta_i$.

Table 5: Hacker Expected Payoff Functions: Microsoft

Hacker Action	Payoff
$(E,\ S)$	$\Pi^{M}_{(E,S)}(\theta, \{\Gamma_{nu}, \Gamma_u, \Gamma_v\}) = \alpha \left[D \sum_{i\in\Gamma_{nu}} \theta_i - \phi_u\right] + (1-\alpha)\left[\delta D \sum_{i\in\Gamma_{nu}\cup\Gamma_u} \theta_i - c_s\right]$
$(E,\ X)$	$\Pi^{M}_{(E,X)}(\theta, \{\Gamma_{nu}, \Gamma_u, \Gamma_v\}) = \alpha \left[D \sum_{i\in\Gamma_{nu}} \theta_i - \phi_u\right]$
$(S,\ S)$	$\Pi^{M}_{(S,S)}(\theta, \{\Gamma_{nu}, \Gamma_u, \Gamma_v\}) = \alpha \left[\widehat{\delta} D \sum_{i\in\Gamma_{nu}\cup\Gamma_u} \theta_i\right] + (1-\alpha)\left[\delta D \sum_{i\in\Gamma_{nu}\cup\Gamma_u} \theta_i\right] - c_s$
$(S,\ X)$	$\Pi^{M}_{(S,X)}(\theta, \{\Gamma_{nu}, \Gamma_u, \Gamma_v\}) = \alpha \left[\widehat{\delta} D \sum_{i\in\Gamma_{nu}\cup\Gamma_u} \theta_i - c_s\right]$
$(X,\ S)$	$\Pi^{M}_{(X,S)}(\theta, \{\Gamma_{nu}, \Gamma_u, \Gamma_v\}) = (1-\alpha)\left[\delta D \sum_{i\in\Gamma_{nu}\cup\Gamma_u} \theta_i - c_s\right]$
$(X,\ X)$	$\Pi^{M}_{(X,X)}(\theta, \{\Gamma_{nu}, \Gamma_u, \Gamma_v\}) = 0$

There are five main drivers of the Nash equilibria in this model: the three stated in Section 2 and the following two conditions.

(d) Does the cost of updating exceed the cost of switching to the new version of the software package? I.e.

$$c_v \lesseqgtr c_u + \phi_u \tag{17}$$

(e) Does the cost of searching for an N-Day exceed the payoff?

$$\phi_u \lesseqgtr D \sum_{i\in I} \theta_i \tag{18}$$

When search costs exceed the expected payoff of search under the Non-Disclosure sub-game, the hacker will always play (X). Given the hacker strategy of exiting the game, all software users will not update. Therefore, the equilibrium is $\left(A_M^{(1-\alpha)*}, \left(A_{M,i}^{(1-\alpha)*}\right)_{i\in I}\right) = ((X), (nu)_{i\in I})$.

If $c_s < \delta D \sum_{i\in I} \theta_i$, then via the best responses of both software users and the hacker, the Nash equilibria under medium search costs are as follows in Theorem 2. Define $\Omega_M \equiv \left\{k \in I | \theta_k \geq \frac{c_v}{\delta(v+D)}\right\}$.

Theorem 2 *Let $k_{min} \in \Omega_M$ be the minimal software user in Ω_M. Then under low search costs in the Non-Disclosure sub-game, if*

$$c_s < \delta D \sum_{i\in I\setminus\Omega_M} \theta_i \tag{19}$$

Then the Nash equilibrium is

$$\left(A_M^{(1-\alpha)*}, \left(A_{M,i}^{(1-\alpha)*}\right)_{i\in I}\right) = \left((S), ((nu)_{i\in\Gamma_{nu}^{k_{min},nd*}}, (v)_{j\in\Gamma_v^{k_{min},nd*}})\right) \tag{20}$$

Where $\Gamma_{nu}^{k_{min},nd*} = \{i \in I | \theta_i < \theta_{k_{min}}\}$, and $\Gamma_v^{k_{min},nd*} = \{j \in I | \theta_j \geq \theta_{k_{min}}\}$.

Otherwise, there exists a pivotal software user $k^* \in \Omega_M$ and a mixed strategy for software user k^* strategy, $p_{k^*}^{v*} \in [0,1]$, such that

$$c_s = \delta\left(D \sum_{i\in\Gamma_{nu}^{k^*,nd*}} \theta_i + (1 - p_k^{v*}) D\theta_{k^*}\right) \tag{21}$$

Then the Nash equilibrium is

$$\left(A_M^{(1-\alpha)*}, \left(A_{M,i}^{(1-\alpha)*}\right)_{i\in I}\right) = \Big(\big(\rho^*(S), (1-\rho^*)(X)\big), \big((nu)_{i\in \Gamma_{nu}^{k^*,nd*}}, (p_{k^*}^{v*}(v), (1-p_{k^*}^{v*})(nu)), \\ (v)_{j\in \Gamma_v^{k^*,nd*}}\big)\Big) \tag{22}$$

Where $\rho^* = \frac{c_v}{\theta_{k^*}\delta(v+D)}$, $\Gamma_{nu}^{k^*,nd*} = \{i \in I | \theta_i < \theta_{k^*}\}$, *and* $\Gamma_v^{k^*,nd*} = \{j \in I | \theta_j > \theta_{k^*}\}$.

Now to solve for the Nash equilibria under the Disclosure sub-game. Notice that both the hacker and the software users have three actions they could each take. In Section 2.2, the equilibria cases followed from the relation between the cost of searching and the expected payoffs from searching. However, due to the new action available to the software users, (v), and the enrollment fee, ϕ_u, there now exist extra cases dependent on Equations 17 and 18.

If there are both high or medium search costs and high exploitation costs, i.e. $c_s > \widehat{\delta}D\sum_{i\in I}\theta_i$ and $\phi_u > D\sum_{i\in I}\theta_i$, then notice that both searching for Zero- and N-Days are too costly, therefore, the hacker will always exit the game. Given this strategy, the workers will all not update. Hence, the Nash equilibrium is $(A_M^{\alpha*}, (A_{M,i}^{\alpha*})_{i\in I}) = ((X), (nu)_{i\in I})$.

The last case to examine is when the exploitation costs of the N-Day are low.

Theorem 3 *If* $c_s > \widehat{\delta}D\sum_{i\in I}\theta_i$ *and* $\phi_u \leq D\sum_{i\in I}\theta_i$, *while the software users face* $c_v < c_u + \phi_u$, *and*

$$\phi_u < D\sum_{i\in I\setminus \Omega_M}\theta_i \tag{23}$$

Then the Nash equilibrium is

$$(A_M^{\alpha*}, (A_{M,i}^{\alpha*})_{i\in I}) = \Big((E), \big((nu)_{i\in \Gamma_{nu}^{d*}}, (v)_{j\in \Gamma_v^{d*}}\big)\Big) \tag{24}$$

Where $\Gamma_{nu}^{d*} = \{i \in I \setminus \Omega_M\}$ *and* $\Gamma_v^{d*} = \{j \in \Omega_M\}$.

Otherwise if $c_s > \widehat{\delta}D\sum_{i\in I}\theta_i$ *and* $\phi_u \leq D\sum_{i\in I}\theta_i$, *while the software users face* $c_v < c_u + \phi_u$, *and there exists* $k^* \in \Omega_M$ *and a mixed strategy for software user* k^*, $p_{k^*}^{v*} \in [0,1]$, *such that*

$$\phi_u = D\sum_{i\in \Gamma_{nu}^*}\theta_i + (1-p_{k^*}^{v*})D\theta_{k^*} \tag{25}$$

Then the Nash equilibrium of the game is

$$(A_M^{\alpha*}, (A_{M,i}^{\alpha*})_{i\in I}) = \Big(\big(\rho^*(E), (1-\rho^*)(X)\big), \big((nu)_{i\in \Gamma_{nu}^{d*}}, (p_{k^*}^{v*}(v), (1-p_{k^*}^{v*})(nu)), (v)_{j\in \Gamma_v^{d*}}\big)\Big) \tag{26}$$

Where $\Gamma_{nu}^{d*} = \{i \in I | \theta_i < \theta_{k^*}\}$, $\Gamma_v^{d*} = \{j \in I | \theta_j > \theta_{k^*}\}$, *and* $\rho^* = \frac{c_v}{\theta_{k^*}(v+D)}$.

4.1 Welfare Analysis

Now to investigate whether this new "Extended Coverage" will be a welfare improving policy. This section flows as follows: First, define the optimal policy; Then, the welfare improving policy will be solved for each of the different cost scenarios.

Definition 2. *The optimal policy $\Psi^* \in \{Microsoft,\ Disclosure, Non-Disclosure\}$ is chosen such that:*

$$\Psi^* = argmax_{\psi \in \{M,d,nd\}} \Big\{ \sum_{i\in I} U_M(A_M^{\alpha *}, A_M^{(1-\alpha)*}, A_{M,i}^{\alpha *}, A_{M,i}^{(1-\alpha)*}, \theta_i),\ \sum_{i \in I} U_d(A_d^{\alpha *}, A_d^{(1-\alpha)*}, A_i^*, \theta_i), \sum_{i\in I} U_{nd}(A_{nd}^*, \theta_i) \Big\} \tag{27}$$

Where $((A_d^{\alpha *}, A_d^{(1-\alpha)*}), (A_i^*)_{i\in I})$, (A_{nd}^*), and $((A_M^{\alpha *}, A_M^{(1-\alpha)*}), (A_{M,i}^{\alpha *}, A_{M,i}^{(1-\alpha)*})_{i \in I})$ are the Nash equilibria of the Disclosure, Non-Disclosure, and Microsoft policies, respectively.

Beginning with the high search cost case, recall that the equilibria of the Microsoft policy game are split into two sub-cases. These two cases can be identified by Inequality 18. If $\phi_u > D\sum_{i\in I}\theta_i$, then both Microsoft and Non-Disclosure are optimal policies. However, if $\phi_u \leq D\sum_{i\in I}\theta_i$, then Non-Disclosure is the optimal policy.

Therefore, for the new policy to be effective under high search costs, the extended service fee must be large. Also notice that if $\phi_u \leq D\sum_{i\in I}\theta_i$, i.e. the exploitation fee is low, then the Nash equilibrium of the hacker exit when a vulnerability is not found and to mix between exploitation of the N-Day and exiting the game. Then, Microsoft is preferred to Disclosure when

$$\rho_M^* \left[\sum_{i \in \Gamma_{nu}^{M*}} \theta_i + (1 - p_{k^*}^{M*})\theta_{k^*} \right] + \xi^{M*} c_v < (v + D) \sum_{i \in \Gamma_{nu}^{d*}} \theta_i \tag{28}$$

Given medium search costs and high exploitation costs, the welfare equation for the software users is

$$\sum_{i\in I} U_M(A_M^{\alpha *}, A_M^{(1-\alpha)*}, A_{M,i}^{\alpha *}, A_{M,i}^{(1-\alpha)*}, \theta_i) = v \sum_{i \in I}\theta_i \tag{29}$$

Therefore, compared to Disclosure, the software users do not need to either update or be hacked via the released patch, and compared to Non-Disclosure, the hacker is not going to be searching for a Zero-Day, and thus the software users will not bear the burden of the expected damages. Hence, as discussed in Theorem 4, the new policy proposed by Microsoft is optimal.

The next case to discuss is when the exploitation cost is low, $\phi_u \leq D\sum_{i\in I}\theta_i$, and the cost of installing the new version is less than the cost of updating, $c_v \leq c_u + \phi_u$. Comparing the new Microsoft policy to Disclosure and Non-Disclosure, the following inequality describes when the new Microsoft policy is optimal.

$$\alpha \rho_M^* (1-\delta) \left[\sum_{i \in \Gamma_{nu}^{M*}} \theta_i + (1 - p_{k^*}^{v*})\theta_{k^*} \right] + \xi_v^* \frac{c_v}{v+D} \leq \min \Big\{ \delta \sum_{i\in I} \theta_i, \alpha \sum_{i \in \Gamma_{nu}^{d*}} \theta_i + (1-\alpha)\delta \sum_{i\in I}\theta_i + \xi_v^* \frac{c_u}{v+D} \Big\} \tag{30}$$

Finally, if $c_v > c_u + \phi_u$, then, under the Disclosure sub-game, the high-type software users will update. Whereas, in the Non-Disclosure sub-game, the high-type software users will install the new version of the software to protect their computers, hence $\rho_M^{d*} \neq \rho_M^{nd*}$. Thus yields the following condition for when "Extended Support" of Windows 7 is the optimal policy.

$$\alpha \rho_M^{d*} \left[\sum_{i \in \Gamma_{nu,nd}^{M*}} \theta_i + (1 - p_{k^*}^{u*})\theta_{k^*} \right] + (1-\alpha)\rho_M^{nd*} \left[\sum_{i \in \Gamma_{nu,nd}^{M*}} \theta_i + (1 - p_{k^*}^{v*})\theta_{k^*} \right] + \xi^* \frac{\alpha(c_u + \phi_u) + (1-\alpha)c_v}{v + D}$$
$$\leq \min \left\{ \delta \sum_{i \in I} \theta_i,\ \alpha \sum_{i \in \Gamma_{nu}^{d*}} \theta_i + (1-\alpha)\delta \sum_{i \in I} \theta_i + \xi^* \frac{c_u}{v + D} \right\} \tag{31}$$

Theorem 4 *Let $\widehat{\delta} D \sum_{i \in I} \theta_i \leq c_s < \delta D \sum_{i \in I} \theta_i$. Then the cases satisfying Inequality 18 are*

1. *If $\phi_u > D \sum_{i \in I} \theta_i$, then Microsoft is the optimal policy.*
2. *If $\phi_u \leq D \sum_{i \in I} \theta_i$, $c_v \leq c_u + \phi_u$, and Inequality 30 is satisfied, then Microsoft is an optimal policy.*
3. *If $\phi_u \leq D \sum_{i \in I} \theta_i$, $c_v \leq c_u + \phi_u$, and Inequality 30 is not satisfied, then Microsoft is not an optimal policy.*
4. *If $\phi_u \leq D \sum_{i \in I} \theta_i$, $c_v > c_u + \phi_u$, and Inequality 31 is satisfied, then Microsoft is an optimal policy.*
5. *If $\phi_u \leq D \sum_{i \in I} \theta_i$, $c_v > c_u + \phi_u$, and Inequality 31 is not satisfied, then Microsoft is not an optimal policy*

Notice that ϕ_u can be used as a weapon to harm hackers. In order for Microsoft's new policy to be effective under medium search costs, the optimal extended service fee and cost of installing the new version are interdependent. The first way for Microsoft to maximize software user welfare is to pick a very large support fee, i.e. high exploitation costs. This prices the hacker out of the market, while also allowing for the software users to not have to pay to install updates or update their software version since the hacker is priced out of the exploitation market. However, under low exploitation costs, for the Microsoft policy to maximize software user welfare they must choose c_v such that either Inequality 30 or Inequality 31 hold.

5 Conclusion

Sun Tzu said: "Know thy self, know thy enemy. A thousand battles, a thousand victories." This sentiment is just as relevant in cybersecurity as it was in the 5^{th} century BC. The optimal policy debate should be centered around how policies influence both the hacker's and software users' behavior. The ease with which the hacker is able to infiltrate the network can be decreased via appropriate disclosure policies. Since the cost of searching for Zero-Days has drastically increased over the last couple of years, the hacker desires more disclosure to decrease his costs. Hence, Disclosure can only be an optimal policy in cases when the cost to the hacker of searching for a Zero-Day vulnerability is small. The policies of Non-Disclosure and Microsoft's new policy both decrease hacker interference in the network as well as increase overall software user welfare.

The idea of this paper is to push the vulnerability disclosure literature toward thinking about the appropriate assumptions faced by hackers, software users, and software vendors. As the title implies, this is a simplified explanation of the problem that firms face. For example, the Equifax hack can be traced to an unpatched vulnerability, however there is more at play than is discussed in this static model. Many firms do not immediately update their software packages since doing so may inadvertently negatively affect other software packages. This is beyond the scope of this paper, as this is an introduction to a theoretical approach to the problem, and will be a focus of future research.

Bibliography

[1] Eric Rescorla. Is finding security holes a good idea? Security Privacy, IEEE, 3(1):14–19, 2005.

[2] Ashish Arora, Rahul Telang, and Hao Xu. Optimal policy for software vulnerability disclosure. Management Science, 54(4):642–656, 2008.

[3] Ivan PL Png, Candy Q Tang, and Qiu-Hong Wang. Hackers, users, information security. WEIS Conference Precedings, 2006.

[4] Ye Hong and William Neilson. Cybercrime and punishment: A rational victim model. Working Paper, 2018.

[5] Gary S. Becker. Crime and punishment: An economic approach. Springer, 1968.

[6] Jay Pil Choi, Chaim Fershtman, and Neil Gandal. Network security: Vulnerabilities and disclosure policy. The Journal of Industrial Economics, 58(4):868–894, 2010.

[7] Ashish Arora, Anand Nandkumar, and Rahul Telang. Does information security attack frequency increase with vulnerability disclosure? an empiricial analysis. Information Systems Frontiers, 8(5):350–362, 2006.

[8] Andy Ozment. Bug auctions: Vulnerability markets reconsidered. Workshop on the Economics of Information Security, 2004.

[9] Christopher Coyne and Peter Leeson. Who's to protect cyberspace? Journal of Law, Economics & Policy, 2:473–496, 2005.

[10] Aron Laszka, Mingyi Zhao, and Jens Grossklags. Banishing misaligned incentives for validating reports in bug-bounty platforms. European Symposium on Research in Computer Security, pages 161–178, 2016.

[11] Andreas Kuehn and Milton Mueller. Analyzing bug bounty programs: An institutional perspective on the economics of software vulnerabilities. TPRC Conference Paper, 2016.

[12] Iulia Ion, Rob Reeder, and Sunny Consolvo. "...no one can hack my mind": Comparing expert and non-expert security practices. Symposium on Usable Privacy and Security (SOUPS), 2015.

[13] Priyanka Goyal, Sahil Batra, and Ajit Singh. A literature review of security attack in mobile ad-hoc networks. International Journal of Computer Applications, 9(12), 2010.

[14] K. Kalambe and S. Apte. An exhaustive survey on security solutions in manets. International Journal of Computer Science and Engineering, 5, 2017.

[15] Sachin Lalar. Security in manet: Vulnerabilities, attacks & solutions. International Journal of Multidisciplinary and Current Research, 2014.

[16] Ulrik Brandes and Daniel Fleischer. Centrality measures based on current flow. Proc. 22nd Symp. Theoretical Aspects of Computer Science (STACS 05), pages 533–544, 2005.

[17] K. Stephenson and M. Zelen. Rethinking centrality: Methods and examples. Social Networks, 11:1–37, 1989.

[18] Marcin Dziubinski and Sanjeev Goyal. How do you defend a network? Theoretical Economics, 12(1):331–376, 2017.

[19] Diego Cerdeiro, Marcin Dziubinski, and Sanjeev Goyal. Individual security, contagion, and network design. Journal of Economic Theory, 170:182–226, 2017.

[20] Sanjeev Goyal and Adrien Vigier. Attack, defense and contagion in networks. Review of Economic Studies, 81(4):1518–1542, 2014.

[21] Stefan Frei, Dominik Schatzmann, Bernhard Plattner, and Brian Trammell. Modeling the security ecosystem – the dyanmics of (in)security. Economics of Information Security and Privacy Chapter 6, 2010.

[22] Jens Grossklags, Benjamin Johnson, and Nicolas Christin. The price of uncertainty in security games. Economics of Information Security and Privacy Chapter 2, 2010.

[23] Arjun K.C. Software vulnerabilities: Key factors impacting on response time of software vendors in releasing patches for software vulnerabilities. LAP LAMBERT Academic Publishing, 2012.

[24] 2018 internet security threat report (istr). Technical report, Symantec, 2018.

[25] 2016 internet security threat report (istr). Technical report, Symantec, 2016.

[26] Flipping the Economics of Attacks. Technical report, Ponemon Institute, 01 2016.