Disclose or Exploit? A Game-Theoretic Approach to Strategic Decision Making in Cyber-Warfare

Haipeng Chen, Qian Han, Sushil Jajodia, Roy Lindelauf, V. S. Subrahmanian, and Yanhai Xiong

***Abstract*—Today, countries are engaged in the cyber-arms race. With over 16 K new hardware/software vulnerabilities reported in 2018 alone, an important question confronts senior government decision makers when their cyber-warfare units discover a new vulnerability. Should they disclose the vulnerability to the vendor who produced the vulnerable product? Or should they "stockpile" the vulnerability, holding it for developing exploits (i.e., cyber-weapons) that can be targeted at an adversary? Choosing the first option may be important when the affected company is a corporation in the nation state that discovers the vulnerability and/or if that nation state would have a big exposure to that vulnerability. Choosing the second option has obvious advantages to the discovering nation's defense. We formulate the cyber-competition between countries as a repeated cyber-warfare game (RCWG), where two countries (players) compete over a series of vulnerabilities by deciding, at the time of vulnerability discovery, 1) whether to exploit or disclose it and 2) how long to exploit it if they decide to exploit. We define the equilibrium state of the RCWG as a pure strategy Nash equilibrium, and propose a learning-while-competing framework to compute the pure strategy Nash equilibrium of the formulated RCWG. Although testing our results with real data in the murky world of cyber-warfare is challenging, we were able to obtain real statistics from other sources and demonstrate the effectiveness of our proposed algorithm through a set of simulation results under different scenarios using these third-party statistics. We also report on our DiscX system that can help support government decision makers in their decision whether to disclose or exploit a vulnerability that they find.**

***Index Terms*—Cyber-security, decision support system, game theory, national defense and security.**

I. Introduction

During the last few years, cyberattacks have become a powerful weapon on the modern battlefield. For instance, the Stuxnet worm, which was discovered in 2010, is reported by New York Times' national security correspondent David [19] to have been jointly developed by the U.S. and Israel in order to disrupt Iran's nuclear program.

As the space of connected devices increases, more and more vulnerabilities will be discovered in devices whose manufacturers had never considered cyber-security to be an issue. In fact, according to [18], the number of new software and hardware vulnerabilities disclosed every year have been steadily increasing over the past decade. In total, 16 555 vulnerabilities were disclosed in the year 2018—a number that is almost triple than that from 2008. Once a vulnerability is discovered by a government of a country (e.g., in the USA, the National Security Agency, and the U.S. Cyber Command can be viewed as the leaders in the hunt for cyber-vulnerabilities; in the U.K., it is government communications headquarters (GCHQ)), there are two major options. One option is to disclose the vulnerability. This option is significant when one of the two cases is true: 1) the manufacturer of the product containing the vulnerability is based in the same country—in this case, the government has at least some responsibility to ensure their well-being from attacks that might use that vulnerability and damage the manufacturer's credibility, and/or 2) the product in which the vulnerability exists is widely deployed in the same country in which case the government has some obligation to protect those entities using the product from an adversary who might seek to use the vulnerability. The second option is to hold the discovery of the vulnerability secret and to develop an exploit, which may be used to an adversary country.

The U.S. Government's Vulnerability Equities Process (VEP)[1] is a pioneering and rare public statement on the "disclose or exploit" decision-making process: should a government disclose the zero-day vulnerabilities they discover in software and hardware products or should they exploit them for offensive cyber-operations? The European union (EU) has also taken action on this with two nations, Latvia and the Netherlands, at the forefront.[2] The UK's GCHQ agency has also released a report detailing their VEP process.[3]

The U.S. Government's VEP program, though not unexpectedly secretive, appears to be based on an interagency task force called the Equities Review Board (ERB) that meets monthly in

Manuscript received May 2, 2019; revised September 3, 2019, December 2, 2019, and January 3, 2020; accepted January 4, 2020. This work was supported in part by the Office of Naval Research under Grant N00014-16-1-2896 and Grant N00014-15-1-2007. *(Corresponding author: V. S. Subrahmanian.)*

H. Chen, Q. Han, V. S. Subrahmanian, and Y. Xiong are with the Department of Computer Science, Dartmouth College, Hanover, NH 03755 USA (e-mail: haipeng.chen@dartmouth.edu; qian.han@dartmouth.edu; vs@darmouth.edu; yanhai.xiong@dartmouth.edu).

S. Jajodia is with the Center for Secure Information Systems, George Mason University, Fairfax, VA 22030 USA (e-mail: jajodia@gmu.edu).

R. Lindelauf is with the Military Operational Art and Science, Netherlands Defense Academy, 4811 XC Breda, Netherlands (e-mail: RHA.Lindelauf.01@NLDA.NL).

Digital Object Identifier 10.1109/JSYST.2020.2964985

[1]Vulnerabilities Equities Policy and Process for the United States Government, November 15, 2017. https://www.whitehouse.gov/sites/whitehouse.gov/files/images/External%20-%20Unclassified%20VEP%20Charter%20FINAL.PDF

[2]https://www.hackerone.com/blog/Software-Vulnerability-Disclosure-Europe-Summary-and-Key-Highlights-European-Parliament-CEPS

[3]https://www.gchq.gov.uk/information/equities-process

order to assess the pros and cons of disclosing each vulnerability. It is not clear what, if any, advanced scientific methods are used to support this process. The goal of this article is to develop the mathematical and computational foundations, together with our developed prototype system called DiscX to help support and inform this process with advanced artificial intelligence (AI) techniques.

The answer to the "disclose or exploit" question depends upon many factors some of which can be given as follows.

1) How much would disclosure help a nation's corporations and/or protect users of the affected technology in that nation?
2) What is the probability that either an adversary or a third party, e.g., independent hacker or cyber-security company, will discover and disclose the vulnerability, making it available for exploitation by the nation's adversaries?
3) What is the probability that an adversary will exploit the vulnerability and potentially affect sales and/or increase the liability of the nation's companies?
4) In the event that a decision to exploit the vulnerability is made, should there also be a plan and/or process to disclose it in the future and if so, when?

We propose a *Repeated Cyber-Warfare Game* (RCWG)-based formulation of the problem of optimal decision making for a series of vulnerabilities that emerge over time. Our RCWG framework is further decomposed into multiple one-stage games. Each one-stage game corresponds to one vulnerability, where the players are the two entities (i.e., countries), the strategy is a two-dimensional variable, which specifies 1) whether or not to exploit the vulnerability and 2) how long should the exploit be used if the player decides to exploit. We, then, derive explicit payoff functions for the players. We, then, define a pure strategy Nash equilibrium to describe the status of the one-stage game in which no player is able to gain any additional payoff by unilaterally changing his/her strategy.

There are two challenges in solving the RCWG. First, we show that the formulated RCWG is essentially an incomplete information game, where the probability distributions of several parameters in the two players' payoff functions are unknown. Second, even if the parameter distributions are known to the players, we show that the computation of best response for one player is a stochastic optimization problem.

We make the following contributions toward solving the formulated RCWG. First, assuming that the parameter distributions are known to the two players, we propose an alternating stochastic optimization approach to compute the pure strategy Nash equilibrium for each stage game associated with each vulnerability. Second, we propose a learning framework to update the belief of the parameter distribution for the two players.

We conduct extensive experimental evaluations on both a simulated environment. The empirical results show that

1) our proposed algorithm always converges with an average of around 220 iterations, and successfully computes an equilibrium strategy for a new vulnerability within 20 s;
2) a player who uses our framework to make a strategic decision about disclosing a vulnerability versus exploiting it will gain a significantly higher payoff compared with baseline strategies, which either always disclose or exploit a new vulnerability.

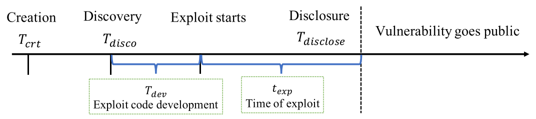


Fig. 1. Lifecycle of a software vulnerability before disclosure.

Finally, we have developed a prototype system called DiscX at Dartmouth College, Hanover, New Hampshire, USA, that supports making the "disclose or exploit" decision based on hard science, but using the expertise and data that the agencies making such decisions might have. DiscX allows the organization in question (e.g., such as the U.S. Government's ERB) to make this decision by providing the DiscX system with inputs based on their knowledge of a specific scenario; DiscX, then, plays out the RCWG proposed in this article and makes a recommendation about the following: 1) should a given vulnerability be exploited? 2) if yes, for how long?

II. Lifecycle of Vulnerabilities Before Disclosure

Before we present the cyber-warfare game model, we first briefly introduce some background information pertaining to the lifecycle of vulnerabilities. Throughout this section, we only consider the viewpoint of the organization that is interested in discovering and exploiting/disclosing the vulnerability. Fig. 1 depicts the lifecycle of a vulnerability, which usually involves the following stages.

1) *Creation time $T_{\rm crt}$ of a vulnerability:* We define the creation time $T_{\rm crt}$ of a vulnerability as the release time of either the hardware/software in which the vulnerability exists. In most such cases, the vulnerability is due to an unknown (to the vendor) security flaw in the product, although there have been accusations (notably those leveled by the U.S. Government against Huawei[4]) that certain products are sometimes knowingly released with intentionally placed zero day vulnerabilities that are unknown to the buyer. By default, we set $T_{\rm crt} = 0$ for each vulnerability.
2) *Discovery time $T_{\rm disco}$ of a vulnerability:* After the creation of a vulnerability, a certain amount of time may elapse before white/black hat hackers recognize the existence of the vulnerability. Note that discovery is distinct from disclosure because the discoverer of the vulnerability may or may not disclose it to the public.
3) *Time $T_{\rm avail}$ when exploit code that leverages the vulnerability is available:* Upon discovery of a vulnerability, exploit code might be developed either by white hat hackers for research purposes, or by black hat hackers with the intention of compromising computer systems. The former is called a "proof-of-concept" (PoC) exploit, while the latter is called a "real-world" exploit. In this article, we do not differentiate between these two types of exploits as

[4]https://www.bbc.com/news/world-us-canada-47046264

it is straightforward to turn PoC exploits into real-world exploits, once the exploit code is ready. Correspondingly, $T_{\text{dev}} = T_{\text{avail}} - T_{\text{disco}}$ is the time required to develop the exploit code.

4) *Exploit start time T_{exploit}:* After the exploit code is available, the exploit code owner can start exploiting the vulnerability. In this article, we consider attacks that use only one exploit at time (which is the case for most cyberattacks) and, hence, there is no payoff of waiting for a certain period of time. As a result, it is a dominant strategy to exploit a vulnerability as soon as the code is available, and, thus, we assume in this article that $T_{\text{exploit}} = T_{\text{avail}}$, i.e., the exploit start time is equal to the time when exploit code is available.[5]
5) *Vulnerability disclosure time T_{disclose}:* The time of disclosure is defined as the first time a vulnerability is made publicly available, either via security forums like Security Focus,[6] or organizations like the Mitre Corporation,[7] which is one of several authorities that assign common vulnerability and exposure (CVE) numbers to disclosed vulnerabilities. By vulnerability disclosure, we mean that information about the vulnerability is made available to the public and to the affected software vendors in particular so that they may produce patches for the vulnerabilities. Since it is a dominant strategy for any entity to exploit the vulnerability until it decides to disclose it, we also assume that $t_{\text{exp}} = T_{\text{disclose}} - T_{\text{exploit}}$, i.e., the time length of exploit is equal to the time of disclosure minus the time of exploit start time (or equivalently, the exploit code availability time).
6) *After disclosure:* Once a vulnerability is made public, cyber-security firms can develop signatures, the affected software vendor(s) can develop the necessary patches, and software owners can purchase and install these patches for their vulnerable software. We omit the detailed introduction of the processes after vulnerability disclosure as they are out of scope of this article.

III. RCWG Model

Game theory is one of the most widely adopted tools to model strategic behaviors of multiple entities, which are involved in a competitive situation. In this article, we formulate the cyber-warfare among two parties as an RCWG. We achieve this by first considering a one-stage game defined on a specific vulnerability, and then extending the one-stage game to a repeated game setting where the two players compete over a series of vulnerabilities. With the RCWG model, our goal is to compute, given the cyber-warfare status of two countries, what would be the equilibrium strategies in terms of 1) whether to disclose or exploit a vulnerability at the discovery, and 2) how long would the exploit last if the decision is to exploit the vulnerability.

A. One-Stage Game

For a given arbitrary but fixed vulnerability, we define a two-player one-stage cyber-warfare game as $G = \{I, \mathbf{a}, \mathbf{u}\}$, where $I = \{1, 2\}$ is the set of two players, $\mathbf{a} = \{a^1, a^2\}$ is the joint strategy of the two players, and $\mathbf{u} = \{u^1(\mathbf{a}), u^2(\mathbf{a})\}$ is the joint payoff function of the two players. In the following, we introduce the different elements in the one-stage cyber-warfare game in detail.

1) Players and Strategies: Given a vulnerability, the two *players* $I = \{1, 2\}$ are two entities (e.g., two countries) who are competing with each other in cyber-warfare. We define the *pure strategy*[8] of player $i \in I$ as a pair $a^i = (b^i, t^i_{\text{exp}})$, which consists of 1) a binary variable $b^i \in \{0, 1\}$ indicating whether to exploit the vulnerability or not and 2) the length of time t^i_{exp} for which to exploit the vulnerability if $b^i = 1$ (i.e., if player i decides to exploit the vulnerability). Thus, there is a payoff function associated with each player.

2) Payoff Function: The payoff of the players is affected by several factors, including

a) *Development cost:* If a player i decides to exploit the discovered vulnerability, there is a cost c^i_{dev} of developing the exploit code. For example, Kaspersky Lab's Roel Schouwenberg estimated that it took a team of ten people at least two to three years to create Stuxnet in its final form [16].
b) *Exploit payoff:* After successful development of exploit code, a player i is able to exploit the vulnerability, with a payoff $r^i_{\text{exp}}(t^{i,*}_{\text{exp}})$ for exploiting it for $t^{i,*}_{\text{exp}}$ days. Typically, $r^i_{\text{exp}}(t^{i,*}_{\text{exp}})$ should be a nondecreasing function with respect to $t^{i,*}_{\text{exp}}$ since the exploit payoff for each day is non-negative. Note that $t^{i,*}_{\text{exp}}$ is the **actual** exploit time of player i, which might be different from player i's action t^i_{exp} if another player is involved in the lifecycle of a vulnerability. As shown in Fig. 1, if this is the lifecycle of a vulnerability, when there is only player i, t^i_{exp} will always be equal to $t^{i,*}_{\text{exp}}$. However, if there is another player j, the exploit would be terminated once player j discloses the vulnerability.[9]
c) *Disclosure payoff:* The player i who is the first to disclose the vulnerability gets a payoff r^i_{disclose} for vulnerability disclosure. The payoff comes from three aspects: 1) the software vendors associated with the player would benefit from the disclosure—for instance, if the vulnerability is in a CISCO router, then CISCO would benefit from such a disclosure as they can improve their product, 2) the entities associated with the player who have exposure to the vulnerability (e.g., all users of CISCO routers if

[5]There are cases where it may make sense to wait before using an exploit. For instance, a government may use multiple exploits in a coordinated attack (e.g., in the case of Stuxnet, four zero day exploits were used according to https://www.zdnet.com/article/stuxnet-attackers-used-4-windows-zero-day-exploits/). We defer such complex attacks to future work.

[6]https://www.securityfocus.com/

[7]https://www.mitre.org/

[8]We consider only pure strategy of the two players as it is straightforward and simple. We take it as a first step and leave the mixed strategy formulation as future research.

[9]In practice, postdisclosure exploit could still exist, the discussion of which is more complex and is omitted in this article for the sake of simplicity. We leave the formulation of a more complicated and inclusive model to future work.

the vulnerability happens to be in CISCO routers) would benefit from the disclosure. and 2) the player gains a good reputation for disclosing it.

The overall payoff of the two players can be formulated under several cases, depending on whether the two players decide to exploit the vulnerability, and which player discloses the vulnerability first. Note that we only write down the payoff of player 1 as the payoff of player 2 can be derived similarly. Recall that the disclosure time T^i_{disclose} is the sum of 1) the discovery time T^i_{disco}, 2) the develop time duration T^i_{dev}, and 3) the exploit time duration t^i_{exp}, i.e., $T^i_{\text{disclose}} = T^i_{\text{disco}} + T^i_{\text{dev}} + t^i_{\text{exp}}$.

Case 1: Neither of the two players decide to exploit, i.e., $b^1 = b^2 = 0$. In this case, the players decide to disclose the vulnerability once they discover it, and the payoff of the players depend on the discovering time. In this case, $T^1_{\text{disclose}} = T^1_{\text{disco}}$, $T^2_{\text{disclose}} = T^2_{\text{disco}}$. Thus

$$u^1(a^1, a^2) = \begin{cases} r^1_{\text{disclose}}, & \text{if} \quad T^1_{\text{disclose}} \le T^2_{\text{disclose}} \\ 0, & \text{if} \quad T^1_{\text{disclose}} > T^2_{\text{disclose}}. \end{cases}$$

Case 2: Player 1 decides to exploit, while player 2 decides not to exploit, i.e., $b^1 = 1, b^2 = 0$. In this case, $T^1_{\text{disclose}} = T^1_{\text{disco}} + T^1_{\text{dev}} + t^1_{\text{exp}}, T^2_{\text{disclose}} = T^2_{\text{disco}}$. Thus

$$u^1(a^1, a^2) =$$
$$\begin{cases} -c^1_{\text{dev}} + r^1_{\text{exp}}(t^1_{\text{exp}}) + r^1_{\text{disclose}}, & \text{if} \quad T^1_{\text{disclose}} \le T^2_{\text{disclose}} \\ -c^1_{\text{dev}} + r^1_{\text{exp}}(t^{1,*}_{\text{exp}}), & \text{if} \quad T^1_{\text{disclose}} > T^2_{\text{disclose}} \end{cases}$$

where $t^{1,*}_{\text{exp}} = T^2_{\text{disclose}} - T^1_{\text{disco}} - T^1_{\text{dev}}$. As discussed earlier, when $T^1_{\text{disclose}} > T^2_{\text{disclose}}$ (i.e., player 2 discloses the vulnerability before player 1), the *actual* exploit time $t^{1,*}_{\text{exp}}$ of player one is disclose time of player 2 minus the discovery time and code development time of player 1.

Case 3: Player 1 decides to not exploit, while player 2 decides to exploit, i.e., $b^1 = 0, b^2 = 1$. In this case, $T^1_{\text{disclose}} = T^1_{\text{disco}}, T^2_{\text{disclose}} = T^2_{\text{disco}} + T^2_{\text{dev}} + t^2_{\text{exp}}$. Thus

$$u^1(a^1, a^2) = \begin{cases} r^1_{\text{disclose}}, & \text{if} \quad T^1_{\text{disclose}} \le T^2_{\text{disclose}} \\ 0, & \text{if} \quad T^1_{\text{disclose}} > T^2_{\text{disclose}}. \end{cases}$$

Case 4: Both the players decide to exploit, i.e., $b^1 = b^2 = 1$. In this case, $T^1_{\text{disclose}} = T^1_{\text{disco}} + T^1_{\text{dev}} + t^1_{\text{exp}}, T^2_{\text{disclose}} = T^2_{\text{disco}} + T^2_{\text{dev}} + t^2_{\text{exp}}$. We have

$$u^1(a^1, a^2) =$$
$$\begin{cases} -c^1_{\text{dev}} + r^1_{\text{exp}}(t^1_{\text{exp}}) + r^1_{\text{disclose}}, & \text{if} \quad T^1_{\text{disclose}} \le T^2_{\text{disclose}} \\ -c^1_{\text{dev}} + r^1_{\text{exp}}(t^{1,*}_{\text{exp}}), & \text{if} \quad T^1_{\text{disclose}} > T^2_{\text{disclose}} \end{cases}$$

where $t^{1,*}_{\text{exp}} = T^2_{\text{disclose}} - T^1_{\text{disco}} - T^1_{\text{dev}}$.

B. Pure Strategy Nash Equilibrium for One-Stage Game

Recall that in game theory, the *best response* of a player is the strategy, which produces the most favorable outcome for that player, given fixed strategies from the other players. As a result, given the pure strategy $a^j = (b^j, t^j_{\text{exp}})$ of player $j \in I$, the best response $a^{i,*}$ of the other player $i \in I$ is

$$a^{i,*} = \arg\max_{a^i} u^i(a^i, a^j). \tag{1}$$

For rational players in a competitive game, Nash equilibrium is commonly adopted to denote the equilibrium outcome of the game, where each player is assumed to know the equilibrium strategies of the other player(s), and no player has anything to gain by changing only their own strategy. Formally, we formulate the *pure strategy Nash equilibrium* of the one-stage cyber-warfare game as

$$a^{1,*} = \arg\max_{a^1} u^1(a^1, a^{2,*}) \tag{2}$$

$$a^{2,*} = \arg\max_{a^2} u^2(a^{1,*}, a^2). \tag{3}$$

C. Repeated Game

In practice, instead of competing over only a single vulnerability, there are usually a series of vulnerabilities, which emerge over time. Naturally, we formulate this *RCWG* as a repeated game.

Suppose $\mathcal{V} = 1, 2, \ldots$ is a sequence of vulnerabilities that occur over time. We can define the RCWG as $\tilde{G} = \{G_v | v \in \mathcal{V}\}$, where each $G^v = (I, \mathbf{a}^v, \mathbf{u}^v)$ is a one-stage game corresponding to vulnerability v. In the RCWG, the objective of a player $i \in I$ is to maximize the overall payoff by playing a pure strategy $a^{v,i}$ for each vulnerability $v \in \mathcal{V}$

$$\max_{\langle a^{v,i} \rangle} \sum_{v \in \mathcal{V}} u^{v,i}(a^{v,1}, a^{v,2}) \tag{4}$$

where $u^{v,i}$ is the payoff of player $i \in I$ at vulnerability $v \in \mathcal{V}$. Correspondingly, the pure strategy Nash equilibrium of the RCWG can be formulated as

$$\langle a^{v,1,*} \rangle = \arg\max_{\langle a^{v,1} \rangle} \sum_{v \in \mathcal{V}} u^{v,1}(a^{v,1}, a^{v,2,*}) \tag{5}$$

$$\langle a^{v,2,*} \rangle = \arg\max_{\langle a^{v,2} \rangle} \sum_{v \in \mathcal{V}} u^{v,1}(a^{v,1,*}, a^{v,2}). \tag{6}$$

Since the payoff function for each stage game $v \in \mathcal{V}$ is independent of the strategies of the other stages, the solution of the repeated game can be decomposed into computing the pure strategy Nash equilibrium of each stage game in (2) and (3).

IV. Solving the RCWG

As introduced earlier, the solution of the RCWG $\tilde{G}$ can be decomposed into solving each stage game G^v for all $v \in \mathcal{V}$. In the following, we first present our solution algorithm for the one-stage game under the assumption that both players have complete information about the distribution of a set of parameters such as the vulnerability discovery time, exploit code development time length, etc. We, then, propose a novel learning and playing framework to tackle the incomplete information in the repeated game setting, which learns the probability distributions for the set of parameters of the players, while computing equilibrium strategies with the solution algorithm for the one-stage game.

Algorithm 1: Best Response Computation of Player i. Function: $BR(f(\alpha), a^j, N)$.

```
Model parameters distribution $f(\alpha)$, strategy $a^j$ of player $j$, sample size $N$ Best response $a^{i,*}$ and payoff $u^{i,*}$
$n = 1 : N$ Generate parameter vector $\alpha$ according to $f(\alpha)$
$a^i \in \mathcal{A}^i$ Compute payoff $u^i$ of player $i$ under strategy $a^j$ of player $j$
$U^i(a^i) = U^i(a^i) + u^i$
$\hat{u}^i(a^i, a^j) = U^i(a^i)/n$
$a^{i,*} = \arg\max_{a^i \in \mathcal{A}^i} \hat{u}^i(a^i, a^j)$
$u^{i,*} = \max_{a^i \in \mathcal{A}^i} \hat{u}^i(a^i, a^j)$
$(a^{i,*}, u^{i,*})$
```

Algorithm 2: Compute Nash Euilibrium of One Stage Game. Function: $NE(f(\alpha), N)$.

```
Model parameters distributions $f(\alpha)$, sample size $N$ Equilibrium strategy $a^{1,*}$, $a^{2,*}$
Randomly initialize $a^1$ and $a^2$ from $\mathcal{A}^1, \mathcal{A}^2$
$(a^{1,*}, u^{1,*}) = BR(f(\alpha), a^2, N)$
$(a^{2,*}, u^{2,*}) = BR(f(\alpha), a^1, N)$
$|a^1 - a^{1,*}| > \epsilon$ and $|a^2 - a^{2,*}| > \epsilon$
$(a^1, u^1) = (a^{1,*}, u^{1,*})$
$(a^2, u^2) = (a^{2,*}, u^{2,*})$
$(a^{1,*}, u^{1,*}) = BR(f(\alpha), a^2, N)$
$(a^{2,*}, u^{2,*}) = BR(f(\alpha), a^1, N)$
$(a^{1,*}, u^{1,*}); (a^{2,*}, u^{2,*})$
```

A. Solving One-Stage Game With Alternated Stochastic Optimization

1) Computing Best Response: As discussed in Section III-A, at each stage game G^v, the payoff function of each player depends on the joint strategy $\mathbf{a}^v$ as well as the set of temporal parameters $T_{\text{disco}}, T_{\text{dev}}$ and the set of utility parameters $c_{\text{dev}}, r_{\text{exp}}, r_{\text{disclose}}$. Suppose $\alpha = (T_{\text{disco}}, T_{\text{dev}}, c_{\text{dev}}, r_{\text{exp}}, r_{\text{disclose}})$ denotes a generic parameter vector. There is a probability distribution function $f^{i,k}(\alpha_k)$ for the kth parameter in α and each player $i \in I$. If we assume that the set of parameter probability distributions are known to both players, the best response of each player i described in (1) can be specified as

$$a^{i,*} = \arg \max_{a^i;\alpha \sim f^i(\alpha)} u^i(a^i, a^j; \alpha) \quad (7)$$

where $f^i(\alpha) = \Pi_k f^{i,k}(\alpha_k)$ is the joint probability distribution of all the parameters. We see immediately that computing the best response of a player is essentially a stochastic optimization problem, since the objective function is stochastic. To solve this stochastic optimization problem, we leverage a Monte Carlo sampling approach, which is shown in Algorithm 1. The key idea of the algorithm is to estimate the reward function $u^1(a^1, a^2; \alpha)$ with the N samples specified in lines 1–5.

The following proposition describes the computational complexity of computing the best response of one player using the proposed Algorithm 1.

Proposition 1: The computational complexity of Algorithm 1 is $\mathcal{O}(N \times |\mathcal{A}_i|)$, where $|\mathcal{A}_i|$ is the cardinality of the feasible action space $\mathcal{A}_i$ for player i.

Proof: It is easy to see that the complexity of computing payoff of one feasible action $a^i \in \mathcal{A}_i$ is $\mathcal{O}(1)$. Meanwhile, we need to do $|\mathcal{A}_i|$ computations to obtain the payoffs of all the feasible actions in $\mathcal{A}_i$, and do N computations for all the N samples. Therefore, the total computational complexity of Algorithm 1 is $\mathcal{O}(N \times |\mathcal{A}_i|)$.

2) Computing One-Stage Game: We can, therefore, compute the pure strategy Nash equilibrium of the one-stage game in Algorithm 2, which alternately computes the best response (using Algorithm 1) of each player until convergence, i.e., the best response of a player is the same for two consecutive iterations. The alternate computation process terminates when the differences $|a^1 - a^{1,*}|$ and $|a^2 - a^{2,*}|$ of the strategies for two consecutive iterations are within a small threshold $\epsilon > 0$.

Remark: Due to the usage of Monte Carlo sampling as a subroutine to compute the best response of a player, there is no guarantee that the computed best response is the exact solution. Therefore, the convergence of pure strategy Nash equilibrium in Algorithm 2 is not theoretically guaranteed. However, we will show via the empirical results in Section V that, with sufficient sampling, Algorithm 2 converges for all the one-stage games in 220 iterations, which takes, on average, within 20 s. The significance of this finding about 220 iterations is to show that despite the lack of a proof of convergence, Algorithm 2 does converge in practice in a reasonable amount of time.

B. Learning Model Parameters

By computing the pure strategy Nash equilibrium of a one-stage game $G^v \in \tilde{G}$, we assume that the probability distribution $f(\alpha)$ of the parameters α is known to both players, i.e., we assume a complete information game setting. However, in reality, one player does not have exact knowledge of the parameter values of the other, creating a major obstacle to the idea of playing an equilibrium strategy for the players.

In order to address this, we propose a learning framework to estimate the parameter distributions $f^i(\alpha)$ as shown in Algorithm 3. In this setting, each player maintains an initial set $\mathcal{V}_0$ of vulnerabilities in their stockpile, in which the parameter values ($\alpha = (T_{\text{disco}}, T_{\text{dev}}, c_{\text{dev}}, r_{\text{exp}}, r_{\text{disclose}})$) of the vulnerabilities are known (in the real world, these will likely be estimated by national security and/or defense organizations). Using $\mathcal{V}_0$, each player can learn a *prior belief* about the probability distribution of the parameters $f(\alpha)$ for the other player. During each one-stage game v in the RCWG, the other player‘s parameter values for different vulnerabilities are observed, together with a player’s own parameter values, and the stockpile of known vulnerabilities is updated as $\mathcal{V} = \mathcal{V} \cup \{v\}$. After this, the belief of the two players’ parameters is updated by refitting the probability distribution of the parameters from the new (i.e., updated) stockpile of vulnerabilities $\mathcal{V}$. This process continues iteratively until the cyber-warfare game is over.

Algorithm 3: RCWG.

```
A set of known CVEs $\mathcal{V}_0$, Sample size $N$
Initialize $\mathcal{V} \leftarrow \mathcal{V}_0$
Fit $\tilde{f}(\alpha)$ with $\mathcal{V}_0$
$v = 1 : N_v$
Get a new vulnerability $v$ with parameter $\alpha_v$;
Compute the NE and corresponding payoffs with the estimated model parameter distributions $\tilde{f}(\alpha)$ $(a^{1,*}, u^{1,*}); (a^{2,*}, u^{2,*}) = NE(\tilde{f}(\alpha), N)$
Update $\mathcal{V}$ $\mathcal{V} = \mathcal{V} \bigcup \{v\}$
Fit $\tilde{f}(\alpha)$ with $\mathcal{V}$
```

V. Experimental Evaluation

A. Experiment Setting

1) RCWG Simulation Environment: Because of the secretive nature of vulnerability discovery and exploit usage by governments, data about many of the parameters in this article is not easily available. For instance, we do not know t_{disco}, the time when vulnerabilities were first discovered by a government, even in cases where the government in question later did disclose the vulnerability as well as in cases where the government in question exploited the vulnerability (which was later discovered as in the case of Stuxnet). For some attacks, we can estimate t_{exp} as this can be obtained from the "first seen" dates maintained by many cybersecurity vendors. Moreover, governments do not seem to ever own up to carrying out cyberattacks. For instance, the U.S. and Israeli governments have not stated that they carried out the Stuxnet attacks, even though it has been widely reported that they did so by highly authoritative and respective sources such as [19].

Fortunately, we were able to get broad statistics on some of these numbers from authoritative national security sources (viz. the Rand Corporation) that we were able to use as the basis for our experiments. Moreover, we note that a government that needs to make these "disclose or exploit" decisions will have knowledge of many of the parameters involved (see Section VIII) when they make a decision.

To evaluate our proposed approach, we set up an RCWG simulation environment, with each vulnerability-related parameter α (i.e., the variables that are mentioned in the lifecycle of a vulnerability) being randomly generated following a Normal distribution $\mathcal{N}(\mu, \sigma)$, where μ and σ are, respectively, the mean and standard deviation of the Normal/Gaussian distribution.

While we do not have access to the true information of the parameter distributions of the vulnerabilities, we would like our RCWG simulation environment to be as close to the real world as possible. To this end, we set most of the vulnerability-related parameters according to the well-researched Rand Corporation report by [1]. This report is based on a rich zero-day vulnerability dataset that contains more than 200 zero-day exploits (and relevant disclosures) spanning 14 years from 2002 to 2016. The detailed setting of the parameters are described as follows.

1) t_{disco}. Consistently with the statistics in [1], we set the mean value of t_{disco} to 200 days with standard deviation as 40 days, so $t_{\text{disco}} \sim \mathcal{N}(200,\, 40)$. In practice, to avoid non-negative values for t_{disco}, we use a truncated normal distribution, and set the minimum and maximum values as 40 and 360, respectively.
2) t_{dev}. Ablon and Bogart [1] stated that the median time for developing an exploit that leverages a zero-day vulnerability is 22 days. Therefore, we set the mean value of t_{dev} as 22 days with standard deviation as 4.4 days. So $t_{\text{dev}} \sim \mathcal{N}(22,\, 4.4)$.[10] Similarly, to avoid non-negative values for t_{dev}, we set the minimum and maximum values for this variable as 4.4 and 39.6 days, respectively.[11]
3) r_{exploit}. For simplicity, we normalize all the cost/payoff related values throughout the experiment section by the payoff of the first day's exploit (i.e., we set the payoff of the first day's exploit as 1). In reality, the daily payoff of an exploit is usually high at the beginning of exploit, and decreases over time. To capture this characteristic, we model this decaying effect of exploit payoff by using an exponential decay function $0.95^{t_{\text{exp}}}$ for the daily exploit payoff.
4) c_{dev}. Ablon and Bogart [1] reported that the average cost for developing an exploit based on a zero-day vulnerability is $\$\,30\,000$. So for simplicity, we set $c_{\text{dev}} \sim \mathcal{N}(6,\, 1.2)$ in our model. The mean value is six times the payoff of the first day's exploit payoff, which is a reasonable amount.
5) r_{disclose}. Ablon and Bogart [1] stated that the average reward for disclosing a zero-day vulnerability is $\$\,50\,000$, which is $5/3$ times the cost of developing the exploit code. As a result, we set the mean value of c_{dev} as $6 \times 5/3 = 10$, and the standard deviation as $10/5 = 2$. Therefore, $r_{\text{disclose}} \sim \mathcal{N}(10, 2)$, i.e., our simulation selects values for r_{disclose} from the normal distribution with a mean of 10 and a standard deviation of 2.
6) Time span of a vulnerability. According to Ablon and Bogart [1], the time span from discovering a major zero-day vulnerability to disclosing the vulnerability takes 1.5 years, so in our model, we set the longest time period from the emergence of a vulnerability to the time of its disclosure to 600 days. This means that the players have a maximum of 600 days to discover vulnerabilities, develop the exploit code, and/or exploit the vulnerability to gain a payoff in one-stage game.
7) Initial number of vulnerabilities. We assume that both players have maintained a database of previously known zero-day vulnerabilities, which enables them to have an initial estimate of the vulnerabilities' parameter distributions. We set the default value of this number as 10 and we will evaluate settings with different numbers of initial set of vulnerabilities.

Our RCWG simulator starts with an initial set of vulnerabilities using which a strategic player would learn an initial belief

[10]We use a mean to standard deviation ratio of 5 throughout the experiments.

[11]Throughout this article, we set the min and max values of a variable to be $\mu \pm 4\sigma$. This is reasonable as 99.7% of values in a normal distribution are less than 3 standard deviations from the mean.

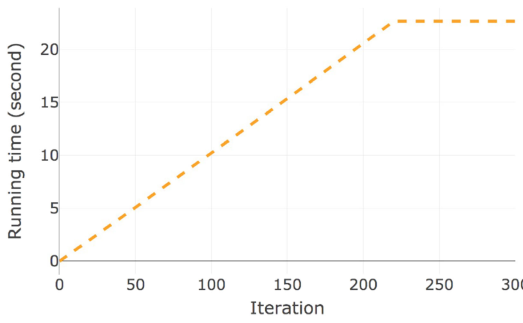


Fig. 2. Runtime analysis.

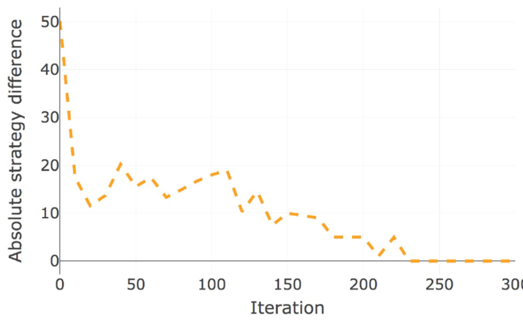


Fig. 3. Convergence analysis.

over the probability distributions of a vulnerability's various parameters. Whenever a new vulnerability emerges, the two players will compete with each other on the new vulnerability using certain strategies. Without loss of generality, we set player 1 to be the player we want to help (i.e., the good guy), while player 2 is the opponent (i.e., the bad guy). Note both the initial set of vulnerabilities and newly emerged vulnerabilities are generated according to the above settings. After a stage game [see (2) and (3)] is played by the two players, the vulnerability is added to the known set of vulnerabilities and the players' knowledge of the vulnerability parameters is updated. In total, there are 150 vulnerabilities in our simulation.

2) Methodology Related Parameters: In this part, we describe the values of other parameters in the experiments, which are related to the algorithms. In computing best response, we set the number of samples N as 100. To compute the Nash equilibrium for each one-stage game, we say that the equilibrium converges when the strategies of both players do not change in three consecutive iterations. We set the maximum number of iterations as a certain number (e.g., 300), so the program stops either an equilibrium solution is obtained or 300 iterations are reached. Each experiment described in the following session is run ten rounds to get the average payoff. A 95% confidence interval is shown for each control experiment. The best response is randomly initialized within a vulnerability's maximum lifespan, which is 600 days as described above.

B. Experiment Results

We now present the experimental evaluations of RCWG, including its convergence and runtime analysis, as well as effectiveness comparison with three baseline strategies.

1) Convergence and Runtime Evaluation:

a) Runtime analysis: Fig. 2 shows the runtime of RCWG, where the x-axis is the number of maximally allowed iterations in computing the Nash equilibrium for one-stage games (i.e., one vulnerability), and the y-axis is the running time in seconds. It is easy to see that the runtime of RCWG is linear w.r.t. the maximally allowed number of iterations until 220, and does not increase with a larger number of iterations. This is because when the number of iterations reaches 220, the algorithm converges in practice. This conjecture is also supported by the following convergence analysis results.

b) Convergence analysis: Fig. 3 presents the results of experiments to determine the convergence properties of our algorithm. The x-axis denotes the number of iterations for which we run Algorithm 2 in a one-stage game, and the y-axis denotes the absolute difference between player 1's best response ($t_{\text{exp}}^{1,*}$) of the current iteration and that of the previous iteration. We observe that as the number of iterations gets larger, the absolute value of the difference between player 1's best responses in two consecutive iterations becomes smaller, and decreases to 0 after around 220 iterations. Recall that the convergence condition in the experiments is that the strategies of both players do not change for three consecutive iterations. That is, if the absolute difference of strategies for both players is 0 for two consecutive iterations, the equilibrium converges. This, together with Fig. 2, demonstrates that although theoretically not guaranteed, RCWG usually is able to converge within 220 iterations in practice, and is able to compute the pure Nash equilibrium for one-stage game in 20 s on average.

We re-emphasize that the significance of the 220 iterations is that it shows that in practice, RCWG converges in a reasonable amount of time.

2) Effectiveness Evaluation: By default, we set the vulnerability-related parameters of both players as described in the above section. To fully evaluate the effectiveness (i.e., obtained payoff) of our proposed algorithm against the baselines under different settings, we conduct a series of control experiments. See Table I for a summary of parameters being evaluated. In the control experiments, we fix the vulnerability-related parameters of player 2, and change one of the parameters for player 1 (while keeping the others unchanged).

We compare our proposed RCWG framework (solved using the learning and competing approach) with three baseline strategies.

1) *Random:* which means the player uses a random strategy for the exploit time t_{exp}.
2) *Always exploit:* which means the player always decides to exploit the vulnerability (i.e., $b = 1$, and $t_{\text{exp}} \to \infty$).
3) *Always disclose:* which means the player always discloses the vulnerability upon discovery of it, i.e., $b = 0$.

In general, the simulation results show that our game-theoretic RCWG model consistently returns the highest payoff under the following various settings.

TABLE I
SUMMARY OF PARAMETERS BEING EVALUATED IN OUR DESIGNED SYSTEM

Parameter	t^1_{disco}/t^2_{disco}	t^1_{dev}/t^2_{dev}	c^1_{dev}/c^2_{dev}	initial #vulnerabilities	Fitting method
Value range of parameters	$1/6-3$	$1/6-3$	$1/6-3$	$10-40$	Beta/Normal distribution

Note that we fix the parameters of Player 2 and vary the parameter values of Player 1.

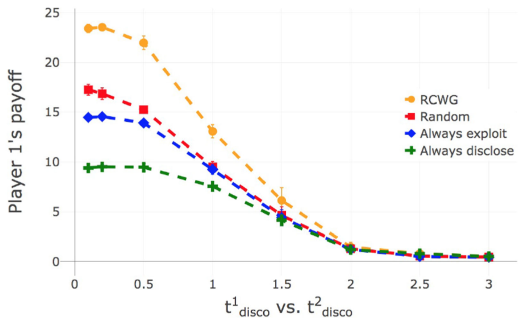


Fig. 4. Payoff comparison under varying vulnerability discovery time t_{disco}.

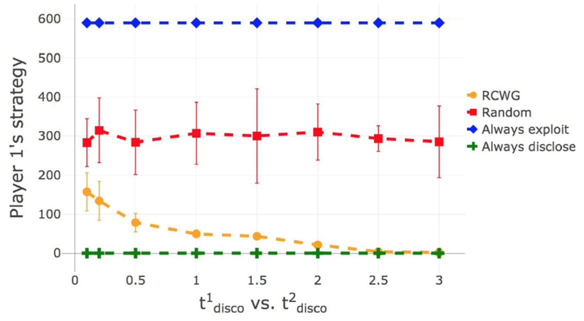


Fig. 5. Strategy comparison under varying vulnerability discovery time t_{disco}.

a) Payoff comparison under varying vulnerability discovery time t_{disco}: In this experiment, we explore how variations in the timing of vulnerability discoveries (which reveal an entity's ability to discover vulnerabilities) would affect the payoff of a player. The result is shown in Fig. 4, where the *x*-axis denotes the ratio of player 1's vulnerability discovery time t_{disco} against that of player 2. For instance, a value of 0.5 means that player 1 discovers vulnerabilities twice as fast as player 2 on average (i.e., it takes a player half the time to discover vulnerabilities on average). On the contrary, a value of 2 indicates the opposite. The *y*-axis is the simulated payoff gained by player 1 using four different strategies. We have the following key results.

1) The less time player 1 needs in order to discover a vulnerability, the higher payoff he/she gains.
2) The payoff of using our proposed learning and competing approach outperforms the three baselines mentioned earlier (random, always exploit, and always disclose).
3) We can see that the payoff difference gained by RCWG is decreasing when the t_{disco} ratio increases (especially compared with "always disclose") and when the ratio is very large, the two lines converge. This is because when the discovery time of vulnerabilities is very long, it is more likely that the other player would disclose the vulnerability even before player 1 discovers it. As a result, when player 1's ability to discover vulnerabilities is poor, the optimal strategy for player 1 is to disclose the vulnerabilities immediately after the discovery—but only if the player gets payoffs through other means (e.g., supporting his country's software industry). This observation is also supported by Fig. 5, where the *x*-axis is the same as Fig. 4, and the *y*-axis is the equilibrium strategy of player 1. We see that as player 1's discovery time increases, the equilibrium exploit time becomes shorter, and becomes 0 when $t^1_{\text{disco}} = 3t^2_{\text{disco}}$. Simply put, the implication is that countries, which do not discover cyber-vulnerabilities quickly and which have strong IT industries (e.g., Austria, some Scandinavian countries) may be better off disclosing vulnerabilities that they discover. On the other hand, countries that discover vulnerabilities relatively frequently (which probably includes the U.S., Russia, China, etc.) may have payoffs for exploitation compared to disclosure.

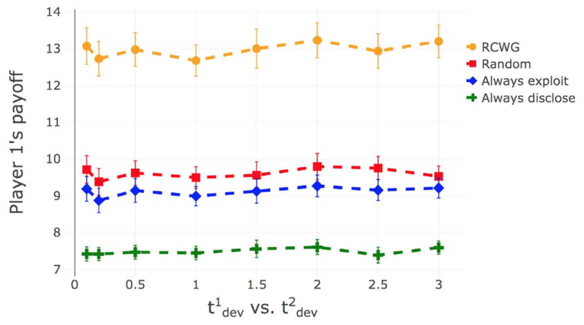


Fig. 6. Payoff comparison under varying vulnerability develop time. t_{dev}.

b) Payoff comparison under varying exploit code development time t_{dev}: Fig. 6 evaluates how t_{dev} affects the payoff of different approaches, where the *x*-axis denotes the ratio of player 1's t_{dev} against that of player 2. If a player has a relatively high exploit code development time, then this means that its ability to develop exploit code is weak. The *y*-axis is the simulated payoff. We have the following observations.

1) The payoff obtained by our proposed RCWG approach is significantly higher than the three baselines under all ratios of player 1 development time versus that of player 2.
2) When t_{dev} changes, the payoffs of different methods do not change a lot. This is because t_{dev} is very small when compared with the discovery time and the overall time span (600 days) of a vulnerability. As a result, t_{dev} has very limited effect on the obtained payoff. The same conclusion can be supported by Fig. 7, where the *x*-axis is the same as Fig. 7, and the *y*-axis is the equilibrium strategy. We can

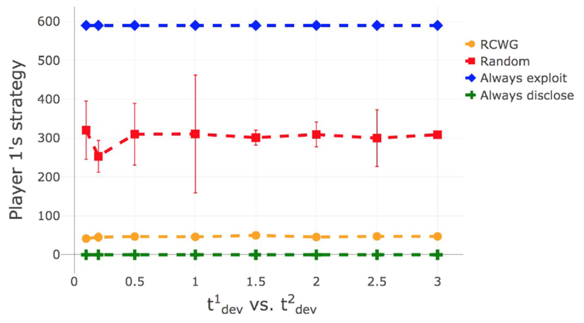


Fig. 7. Strategy comparison under varying vulnerability develop time t_{dev}.

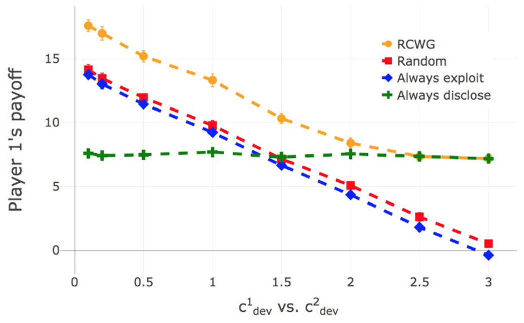


Fig. 8. Payoff comparison under varying cost of developing exploit code c_{dev}.

see that the equilibrium strategy of RCWG barely changes for different t_{dev} values of player 1.

c) Payoff comparison under varying cost of developing exploit code c_{dev}: Fig. 8 evaluates how c_{dev} influences the payoff of different approaches, where the *x*-axis denotes the ratio of player 1's development cost c_{dev} against that of player 2. For instance, this ratio might be relatively high when comparing the U.S. (player 1) versus China (player 2) as developments costs in China are likely to be cheaper than those in the U.S. because of lower salaries. Intuitively, a player who has a low development cost c_{dev} would have an advantage. The *y*-axis is still the simulated payoff. We obtained the following results.

1) When c_{dev} increases, the payoff of all approaches decreases, except for "always disclose," which is approximately constant under varying c_{dev}. This is intuitive since all the other approaches would suffer a higher loss from developing the exploit code due to the increasing c_{dev}.
2) Consistently, our proposed RCWG gains a higher payoff compared with all the baselines.
3) The line of RCWG converges with "always disclose" when c_{dev} gets too large. This is because when it is very expensive to develop exploit code, there are no gains (and there may even be losses) in exploiting a vulnerability, and, therefore, the optimal strategy in this situation is to disclose the vulnerability once it is discovered. Again, this is supported by Fig. 9—when c_{dev} of player 1 is too large, the equilibrium strategy of player 1 converges with "Always disclose."

d) Payoff comparison under varying number of vulnerabilities in the initial set: Fig. 10 evaluates the effect of the scale of the initial set of vulnerabilities that are available to the players, where the *x*-axis is the number of vulnerabilities that are initially known to the players, and the *y*-axis is the simulated payoff. Intuitively, having an initial stockpile of vulnerabilities suggests that the player with the stockpile would have a better understanding and estimation of the vulnerability parameters. We again see from Fig. 10 that our RCWG approach consistently outperforms the baselines regardless of the scale of the initial set of vulnerabilities. Similarly, we also note from Fig. 11 that the equilibrium strategy of player 1 does not change substantially with the varying number of the initial set of vulnerabilities.

e) Payoff comparison under different fitting methods: Fig. 12 compares the simulated payoff of player 1 under two

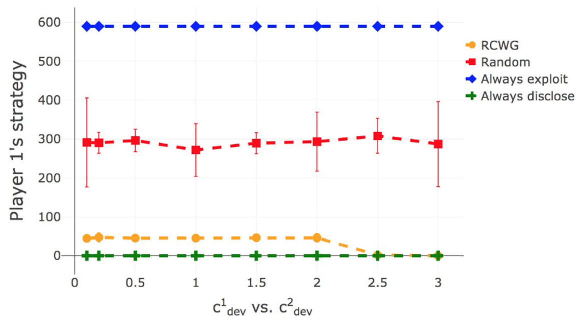


Fig. 9. Strategy comparison under varying cost of developing exploit code c_{dev}.

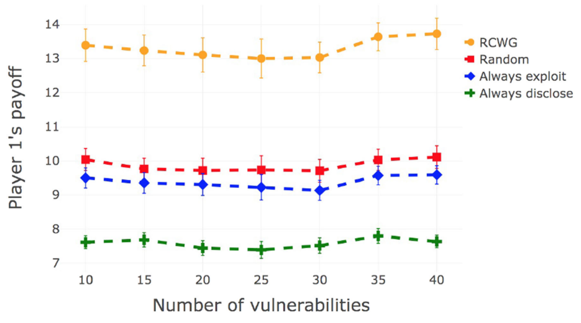


Fig. 10. Payoff comparison under varying number of vulnerabilities in the initial set.

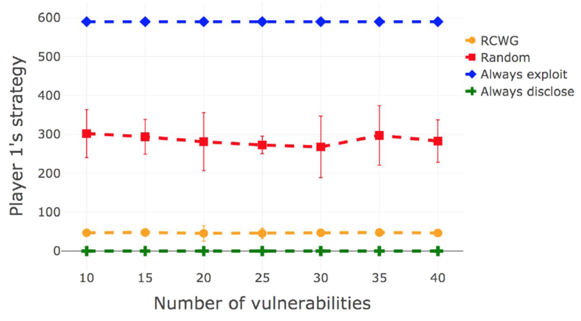


Fig. 11. Strategy comparison under varying number of vulnerabilities in the initial set.

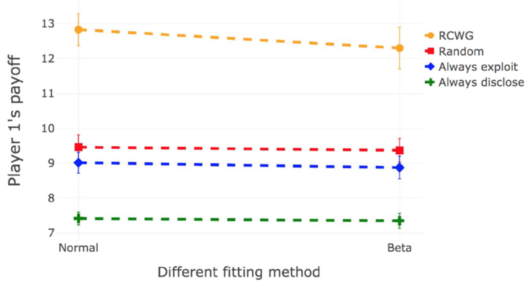


Fig. 12. Payoff comparison under different fitting methods.

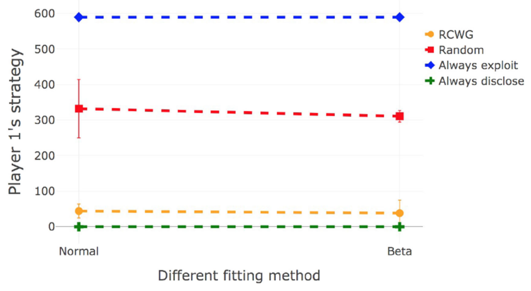


Fig. 13. Strategy comparison under different fitting methods.

different methods of fitting the vulnerability parameter distributions. As described earlier, we use the normal distribution as the ground truth probability distribution for the properties of vulnerabilities. In this experiment, we are interested in estimating what happens if we do not assume that we have prior knowledge of the type of probability distribution. We achieve this by comparing two methods for fitting the probability distribution, i.e., one using the same type of probability distribution function (Normal distribution), and the other using a different type (Beta distribution). We see that the payoff gained by our proposed RCWG decreases slightly, while the payoffs of other approaches (which do not involve the parameter distribution learning process) do not change. Nonetheless, the payoff of RCWG is still significantly higher than all the baselines. Similarly, we can see from Fig. 13 that the equilibrium strategy only changes slightly under the two different fitting methods.

VI. DiscX System

In this section, we describe the DiscX system that we have been building to better support organizations such as the U.S. Government's ERB in making their decisions with the insights provided by a rigorous scientific analysis.

Suppose we work for a government agency that has discovered a new vulnerability. The user first logs into the DiscX system and sees two options as shown in Fig. 14(a). He/she can either look at vulnerabilities that have been previously reported (either by him/her or by others) or enter a new vulnerability that he/she may have discovered. In the screenshot shown in Fig. 14(a), the user wishes to see the vulnerability assigned the ID CVE-2018-15982.[12]

[12]In real world usage, the vulnerability may not be assigned a CVE number by Mitre/NIST—rather it might be assigned an internal ID by the discovering organization.

Fig. 14(b) shows the resulting screen after this choice was made by the user. The screen shows some information about the vulnerability. For instance, the security researcher who found the vulnerability might have input some information about it, e.g., he expects the severity of the vulnerability to be 9.8 on a 0–10 point scale. He also may have input additional information, e.g., that the t_{dev} parameter is one month, meaning that it would take about one month to develop and test an exploit based on this vulnerability as shown in Fig. 14(c). Fig. 14(d) additionally allows the user currently viewing the vulnerability to update some of these assessments as also shown in Fig. 14(e). Additionally, the DiscX system allows the security analyst to post some comments on a discussion board [see Fig. 14(f)], which can be read by other users. A video of the use of the DiscX system is available online.[13]

VII. Related Work

This section talks about the line of related works, including *optimal disclosure policy*, *vulnerability life-cycle analysis*, and *economics, operation research, and game theory in cybersecurity*.

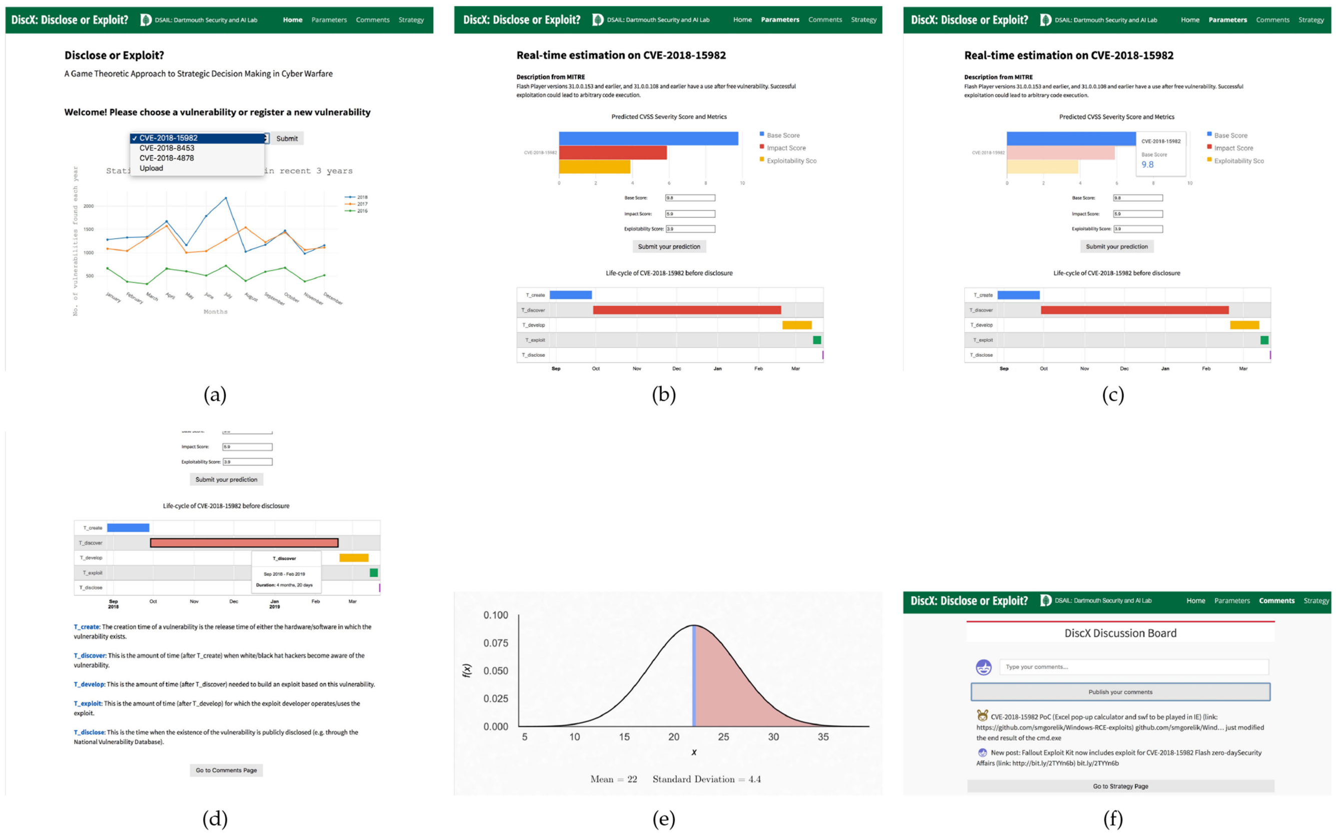
A. Optimal Disclosure Policy

Arora *et al.* [2] showed that full and public disclosure of vulnerabilities allows vendors to respond faster in fixing vulnerabilities, but there is a corresponding increase in the number of attacks when this strategy is adopted. Choi *et al.* [7] examined the effect of both a mandatory disclosure policy and bug bounty programs and found that mandatory disclosures are not beneficial to companies, while bug bounty programs are more beneficial. Cavusoglu *et al.* [6] stated that an optimal disclosure policy depends on the risk of vulnerabilities before and after disclosure, the cost structure of the vulnerable software's user population, and the motivation of the vendor to develop patches for the disclosed vulnerability. They also show that although early discovery always improves social well-being, early warning systems do not necessarily improve social well-being. Caulfield *et al.* [5] presented a set of criteria that governments should use to find the optimal strategy for disclosing a vulnerability, and provided insights on how to achieve a repeatable decision-making process. To the best of our knowledge, there are no previous works that employ game theoretic models to help decision making of the vulnerability disclosure process.

B. Vulnerability Lifecycle Analysis

Frei [11] studied the zero-day vulnerability lifecycle by dividing it into stages. Each stage reflects a specific state of the vulnerability and has varying impacts on the targeted ecosystem (users plus enterprise). The phases include a creation, discovery, exploit, disclosure, patch availability, and patch installation phase. Ablon and Bogart's [1] detailed and excellent study concluded that exploits in past zero-day took a median of 22 days for development. Bilge and Dumitraş [3] found that on average,

[13]https://drive.google.com/open?id=1pE5RKVwiJfnmp7vOAgXbWq4z4OX-Bx7K



Description from MITRE
Flash Player versions 31.0.0.153 and earlier, and 31.0.0.108 and earlier have a use after free vulnerability. Successful exploitation could lead to arbitrary code execution.

T_create: The creation time of a vulnerability is the release time of either the hardware/software in which the vulnerability exists.

T_discover: This is the amount of time (after T_create) when white/black hat hackers become aware of the vulnerability.

T_develop: This is the amount of time (after T_discover) needed to build an exploit based on this vulnerability.

T_exploit: This is the amount of time (after T_develop) for which the exploit developer operates/uses the exploit.

T_disclose: This is the time when the existence of the vulnerability is publicly disclosed (e.g. through the National Vulnerability Database).

CVE-2018-15982 PoC (Excel pop-up calculator and swf to be played in IE) (link: https://github.com/smgorelik/Windows-RCE-exploits) github.com/smgorelik/Wind... just modified the end result of the cmd.exe

New post: Fallout Exploit Kit now includes exploit for CVE-2018-15982 Flash zero-daySecurity Affairs (link: http://bit.ly/2TYYn6b) bit.ly/2TYYn6b

Fig. 14. Selected screenshots of the DiscX system.

a typical zero-day exploit lasts for 312 days in the wild before detection. Frei *et al.* [12] pointed out that from 2001 to 2005, 95% of the exploits are available within a month of disclosure of the vulnerability, a figure that is consistent with that in [1].

C. *Economics, Operation Research, and Game Theory in Cybersecurity*

There has been much work on the use of game theory in cybersecurity—we only discuss a relatively small number as most of these efforts, although excellent, do not bear directly on the problem studied here. Kannan and Telang [15] showed that private companies, which make profits by disclosing vulnerabilities without proper safeguards in place may sometimes lead to higher risks of vulnerability exploits and, thus, are suboptimal. Hewett *et al.* [13] presented a game-theoretic approach to analyze Smart Grid SCADA systems by establishing a sequential, nonzero sum, two player game. Serra *et al.* [20] developed Stackelberg game models, which enable a decision maker to reason about the adversary's maximal expected damage strategy and find the optimal mix of deactivation of buggy software and patching of vulnerabilities in order to minimize the maximal damage caused by an intelligent adversary. Chung *et al.* [8] proposed a game-theoretic model based on expert knowledge to simulate the decision-making process involved in making response to cybersecurity incidents, and determine the optimal strategy to minimize damage caused by an incident. Edwards *et al.* [10] proposed a game-theoretic model showing that the best strategy for the victim in a cyberattack depends on the kind of vulnerability, the knowledge level of the victim, the different payoffs, and the belief to their opponents. Huang and Zhu [14] developed a multistage Bayesian game framework to protect cyber-physical systems proactively and holistically from stealthy advanced persistent threat (APT) attackers under diverse information structures.

To the best of our knowledge, none of the existing works study whether and when to disclose a cyber-vulnerability from the perspective of game theory. However, taking into account the strategic behavior of the decision makers is key to making optimal vulnerability disclosure strategies. Motivated by this, we propose a simple, and yet intuitive RCWG model to help decision makers in the vulnerability disclosure process.

VIII. Real-World Usage and Future Work

The parameters used by the RCWG framework tend to be classified. But the good news is that the parameters are in fact likely to be known to or reasonably estimated by the governments (security/intelligence agencies) that are making the "disclose or exploit" decision. In the rest of this section, we assume that player 1 is the player who is making the disclose or exploit decision. Clearly, a government will know when it discovered a vulnerability t_{disco}. It will also have estimates from its cybersecurity experts on the time t_{avail} by when they will be able to finish developing an exploit based on the vulnerability, as well as expected costs c_{dev} for developing the exploit code (e.g., in terms of person-months, any equipment, software, or travel needed).

However, the government (player 1) will need to come up with a model to assess its reward for exploiting versus disclosing. This would depend very much on the nature of the exploit. For instance, the 2015 hack of the U.S. Office of Personnel Management (attributed to China) led to the theft of over 20 M records. Estimating the payoff of these data to an adversary poses a challenge. On the other hand, a distributed denial-of-service (DDoS) attack such as the one on Wall Street banks in the 2011–2013 time frame (attributed to Iran) cost the banks tens of millions of dollars according to the American Banker magazine.[14] Estimating the reward gained by player 1 if they make a decision to exploit, and estimating the reward gained by the player if they choose to disclose therefore pose some challenges that we are leaving for future work. We note that a start has been made by several authors [4], [9], [17] and so methods for estimating these rewards should build upon these efforts.

In the real world, we note that there are many players involved in the disclose or exploit process. In this article, we have chosen to model this situation as a 2-player game involving the two main combatants. However, in the real world, there can be other players (e.g., security companies and independent hackers). An important future work would be to develop a more complex multiplayer model. One challenge in building such a model is in gathering data about the additional new players and in understanding their incentive/payoff structures.

IX. Conclusion

In this article, we presented a simple, yet insightful two-player RCWG model to formulate the cyber-competition of two entities over a series of cyber-vulnerabilities. The RCWG is decomposed into several one-stage games, where in each game the pure strategy Nash equilibrium is defined to describe the equilibrium status of the one-stage cyber-competition. To solve the formulated RCWG, we proposed a learning and competing framework, where the learning process keeps updating the believed parameter distributions of the vulnerabilities, and the competing process computes whether and when to disclose a newly found vulnerability based on the Nash equilibrium strategy. Our experiment shows the following:

1) though theoretically not guaranteed, our algorithm always converges in 220 iterations (on average) in our experiments—moreover, the convergence occurs in a matter of 20 s on average;
2) the use of our RCWG framework for deciding whether to disclose or exploit a vulnerability yields a significantly higher payoff for the player making the decision compared with a set of three simple baseline strategies;
3) our simulation results show that countries, which develop strengths in discovering vulnerabilities quickly and have a lower cost for developing exploits, would gain substantial payoffs by exploiting newly discovered vulnerabilities. In contrast, countries, which are weaker in discovering vulnerabilities, should tend to disclose the vulnerability for higher payoffs—assuming they have a software industry that would be affected by the exploitation of the vulnerabilities (India is an example that comes to mind).

In addition, we have developed DiscX, the first system (to the best of our knowledge) that can be used by government agencies to make the "disclose or exploit" decision. DiscX is intended to augment the current decision-making procedure for "exploiting versus disclosing" with a rigorous tool that uses agency experts' inputs to help agencies such as the U.S. Government's ERB arrive at an optimal solution.

[14]https://www.americanbanker.com/news/us-charges-iranian-hackers-in-wall-street-cyber-attacks

References

[1] L. Ablon and A. Bogart, *Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits*. Santa Monica, CA, USA: Rand Corporation, 2017.

[2] A. Arora, R. Telang, and H. Xu, "Optimal policy for software vulnerability disclosure," *Manage. Sci.*, vol. 54, no. 4, pp. 642–656, Mar. 2008.

[3] L. Bilge and T. Dumitraş, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 833–844.

[4] B. Cashell, W. D. Jackson, M. Jickling, and B. Webel, The economic impact of cyber-attacks Congressional Res. Service Documents, CRS RL32331 Washington, D.C., USA, 2004.

[5] T. Caulfield, C. Ioannidis, and D. Pym, "The U.S. vulnerabilities equities process: An economic perspective," in *Proc. Int. Conf. Decis. Game Theory Secur.*, 2017, pp. 131–150.

[6] H. Cavusoglu, H. Cavusoglu, and S. Raghunathan, "Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge," *IEEE Trans. Softw. Eng.*, vol. 33, no. 3, pp. 171–185, Mar. 2007.

[7] J. P. Choi, C. Fershtman, and N. Gandal, "Network security: Vulnerabilities and disclosure policy," *J. Ind. Econ.*, vol. 58, no. 4, pp. 868–894, Dec. 2010.

[8] K. Chung, C. A. Kamhoua, K. A. Kwiat, Z. T. Kalbarczyk, and R. K. Iyer, "Game theory with learning for cyber security monitoring," in *Proc. IEEE 17th Int. Symp. High Assurance Syst. Eng.*, 2016, pp. 1–8.

[9] T. Dubendorfer, A. Wagner, and B. Plattner, "An economic damage model for large-scale internet attacks," in *Proc. 13th IEEE Int. Workshops Enabling Technologies: Infrastructure Collaborative Enterprises*, 2004, pp. 223–228.

[10] B. Edwards, A. Furnas, S. Forrest, and R. Axelrod, "Strategic aspects of cyberattack, attribution, and blame," *Proc. Nat. Acad. Sci.*, vol. 114, no. 11, pp. 2825–2830, 2017.

[11] S. Frei, *Security Econometrics: The Dynamics of (in) Security*, vol. 93. Zurich, Switzerland: ETH Zurich, 2009.

[12] S. Frei, M. May, U. Fiedler, and B. Plattner, "Large-scale vulnerability analysis," in *Proc. SIGCOMM Workshop Large-Scale Attack Defense*, 2006, pp. 131–138.

[13] R. Hewett, S. Rudrapattana, and P. Kijsanayothin, "Cyber-security analysis of smart grid SCADA systems with game models," in *Proc. 9th Annu. Cyber Inf. Security Res. Conf.*, 2014, pp. 109–112.

[14] L. Huang and Q. Zhu, "Analysis and computation of adaptive defense strategies against advanced persistent threats for cyber-physical systems," in *Proc. Int. Conf. Decis. Game Theory Secur.*, 2018, pp. 205–226.

[15] K. Kannan and R. Telang, "Economic analysis of market for software vulnerabilities," in *Proc. 3rd Workshop Econ. Inf. Secur.*, 2004.

[16] D. Kushner, "The real story of stuxnet," *IEEE Spectr.*, vol. 50, no. 3, pp. 48–53, Mar. 2013.

[17] R. Layton and P. A. Watters, "A methodology for estimating the tangible cost of data breaches," *J. Inf. Secur. Appl.*, vol. 19, no. 6, pp. 321–330, 2014.

[18] Mitre Corporation. Browse vulnerabilities by date. 2019. [Online]. Available: https://www.cvedetails.com/browse-by-date.php

[19] D. E. Sanger, *Confront and Conceal: Obama's Secret Wars and Surprising Use of American Power*. New York, NY, USA; Crown, 2012.

[20] E. Serra, S. Jajodia, A. Pugliese, A. Rullo, and V. Subrahmanian, "Pareto-optimal adversarial defense of enterprise systems," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 3, 2015, Art. no. 11.