Draining the Water Hole: Mitigating Social Engineering Attacks with CyberTWEAK

Zheyuan Ryan Shi,[1] **Aaron Schlenker,**[2] **Brian Hay,**[3]
Daniel Bittleston,[1] **Siyu Gao,**[1] **Emily Peterson,**[1] **John Trezza,**[1] **Fei Fang**[1]
[1]Carnegie Mellon University, [2]Facebook, Inc., [3]Security Works
ryanshi@cmu.edu, aschlenker@fb.com, bhay@securityworks.com
{dbittles, siyug, emilypet, jtrezza}@andrew.cmu.edu, feif@cs.cmu.edu

Abstract

Cyber adversaries have increasingly leveraged social engineering attacks to breach large organizations and threaten the well-being of today's online users. One clever technique, the "watering hole" attack, compromises a legitimate website to execute drive-by download attacks by redirecting users to another malicious domain. We introduce a game-theoretic model that captures the salient aspects for an organization protecting itself from a watering hole attack by altering the environment information in web traffic so as to deceive the attackers. Our main contributions are (1) a novel Social Engineering Deception (SED) game model that features a continuous action set for the attacker, (2) an in-depth analysis of the SED model to identify computationally feasible real-world cases, and (3) the CYBERTWEAK algorithm which solves for the optimal protection policy. To illustrate the potential use of our framework, we built a browser extension based on our algorithms which is now publicly available online. The CYBERTWEAK extension will be vital to the continued development and deployment of countermeasures for social engineering.

1 Introduction

Social engineering attacks are a scourge for the well-being of today's online user and the current threat landscape only continues to become more dangerous (Mitnick and Simon 2011). Social engineering attacks manipulate people to give up confidential information through the use of phishing campaigns, spear phishing whaling or watering hole attacks. For example, in watering hole attacks, the attacker compromises a legitimate website and redirects visitors to a malicious domain where the attacker can intrude the user's network. The number of social engineering attacks is growing at a catastrophic rate. In a recent survey, 60% organizations were or may have been victim of at least one attack (Agari 2016). Such cybercrime poses an enormous threat to the security at all levels – national, business, and individual.

To mitigate these attacks, organizations take countermeasures from employee awareness training to technology-based defenses. Unfortunately, existing defenses are inadequate. Watering hole attackers typically use zero-day exploits, rendering patching and updating almost useless (Sutton 2014). Sand-boxing potential attacks by VM requires high-end hardware, which hinders its wide adoption (Farquhar 2017). White/blacklisting websites is of limited use, since the adversary is strategically infecting trustworthy websites.

We propose a game-theoretic deception framework to mitigate social engineering attacks, and, in particular, the watering hole attacks. Deception is to delay and misdirect an adversary by incorporating ambiguity. Watering hole attackers rely on the identification of a visitor's system environment to deliver the correct malware to compromise a victim. Towards this end, the defender can manipulate the identifying information in the network packets, such as the user-agent string, IP address, and time-to-live. Consequently, the attacker might receive false or confusing information about the environment and send incompatible exploits. Thus, deceptively manipulating employees' network packets provides a promising countermeasure to social engineering attacks.

Our Contributions We provide the first game-theoretic framework for autonomous countermeasures to social engineering attacks. We propose the Social Engineering Deception (SED) game, in which an organization (defender) strategically alters its network packets. The attacker selects websites to compromise, and captures the organization's traffic to launch an attack. We model it as a zero-sum game and consider the minimax strategy for the defender.

Second, we analyze the structure and properties of the SED game, based on which we identify real-world scenarios where the optimal protection policy can be found efficiently.

Third, we propose the CYBERTWEAK (Thwart WatEring hole AttacK) algorithm to solve the SED game. CYBERTWEAK exploits theoretical properties of SED, linear program relaxation of the attacker's best response problem, and the column generation method, and is enhanced with dominated website elimination. We show that our algorithm can handle corporate-scale instances involving over 10^5 websites.

Finally, we have developed a browser extension based on our algorithm. The software is now publicly available on the Chrome Web Store.[1] The extension is able to manipulate the user-agent string in the network packets. We take additional steps to improve the its usability and explain the output of CYBERTWEAK intuitively. We believe it will be vital to the

[1] http://bit.ly/CyberTWEAK

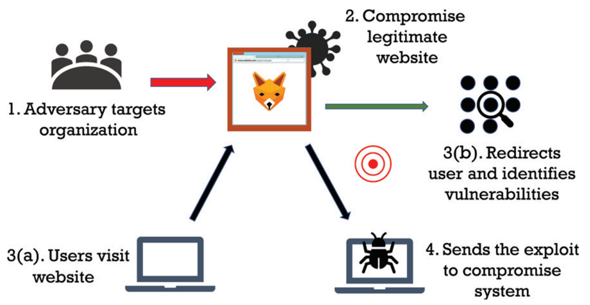

1. Adversary targets organization
2. Compromise legitimate website
3(b). Redirects user and identifies vulnerabilities
3(a). Users visit website
4. Sends the exploit to compromise system

Figure 1: Anatomy of a watering hole attack.

continued development of social engineering defenses.

Related Work Deception is one of the most effective ways to thwart cyberattacks. Recent papers have considered deception techniques for protecting an enterprise network from an attack by sending altered system environment information in response to scans performed during the reconnaissance phase of an attack (Albanese, Battista, and Jajodia 2016; Jajodia et al. 2017). There is a rising interest in building game-theoretic models for deception (Schlenker et al. 2018), in particular in the use of honeypots (Durkota et al. 2015; Píbil et al. 2012) in the enterprise network.

However, there is a fundamental difference between enterprise network defense and social engineering defense. In the former, an adversary targets an organization by compromising computers in the network while in watering hole attacks the attacker targets the user and compromises external websites. A website in SED cannot be properly modeled as a honeypot target, because the defender has no control over it. Neither can the user, because the attack depends on an external task – compromising a website. Instead of actively querying the network, watering hole attackers passively monitor the users' traffic. This necessitates the continuous action space for the attacker in SED, which is also different from most previous works on enterprise network defense.

Laszka, Vorobeychik, and Koutsoukos (2015) study spear phishing, another form of social engineering attacks. The nature of watering hole attacks leads to additional complications. For example, watering hole attackers need to compromise a website and then scan the traffic. Thus, in SED the attacker has two layers of decision making: one continuous and one discrete. This leads to a different problem formulation and solution techniques than those in spear phishing.

2 Watering Hole Attacks

Watering hole attacks are a prominent type of social engineering used by sophisticated attackers. Before we describe our modeling decisions, it is useful to highlight the primary steps in executing a watering hole attack, as illustrated in Fig. 1. In step 1, the attacker identifies a target organization. They use surveys and external information like specialized technical sites to understand the browsing habits of its employees. This allows the adversary to determine the most lucrative websites to compromise for maximum exposure to employees from the targeted organization. In step 2, the adversary compromises a set of legitimate websites. Not only do these websites need to be lucrative, but the attacker also has to be strategic in this choice. For example, compromising Google.com is nearly impossible while the Polish Financial Authority, victim of the 2017 Ratankba malware attacks (Symantec 2017), cannot invest the same security resources. Indeed, in previous attacks the attacker was not observed to compromise all websites (Parliament 2018). In step 3, employees visit the compromised website and are redirected to a malicious website which scans their system environment and the present vulnerabilities. To gather this information, attackers use techniques such as analyzing the user-agent string, operating system fingerprinting, etc. In Step 4, the attacker delivers an exploit for an identified vulnerability. After these steps, the attacker can navigate the target network and access the sensitive information.

Our algorithm and browser extension introduce uncertainty in step 3 of a watering hole attack. Identifying the vulnerabilities in a visitor relies on the information gathered from reconnaissance. The extension modifies the network packets so that the attacker gets false information about the visitor. Deception is not free, though. Altering the network packet can degrade the webpage rendered, e.g., displaying for Android on a Windows desktop. Thus, the defender needs to carefully trade-off security and the quality of service.

In reality, sophisticated attackers typically do not send all exploits without tailoring to the packet information, as defense would become easier after seeing more such unknown exploits. Also, sending all exploits would be flagged as suspicious and get blocked. The attacker would need to get a new zero-day – a costly proposition. Thus, the attacker prefers scanning the system environment of the incoming traffic.

3 Social Engineering Deception Game

We model the strategic interaction between the organization (defender) and an adversary as a two-player zero-sum game, where the defender chooses an alteration policy and the adversary chooses which websites to compromise and decides the effort spent on scanning traffic. In everyday activities employees of a target organization O visit a set of websites W which includes legitimate sites and potential watering holes set up by an adversary. Let t_w^{all} denote the total amount of traffic to $w \in W$ from all visitors and t_w the total traffic to w from O. The defender's alteration policy is represented by $x \in [0,1]^{|W|}$ where x_w is the proportion of O's traffic to website $w \in W$ for which the network packet will be altered. We assume a drive-by download attack will be unsuccessful if, and only if an employee's packet is altered. However, it is easy to account for different levels of adversary and defender sophistication by adding an additional factor in Eq. (1) below. We consider a cost c_w to alter a single unit of traffic to w. The defender is limited to a budget B_d on the allowable cost.

The adversary first chooses which websites to compromise, represented by a binary vector $y \in \{0,1\}^{|W|}$. If $y_w = 1$, i.e., they turn website w into a watering hole, they must pay a cost π_w. The attacker has a budget B_a for compromising websites (w.l.o.g. we assume $\pi_w \leq B_a \; \forall w \in W$). The adversary then decides the scanning effort for each compromised website which can enable them to send exploits tailored to the packet information. We use e_w to denote how much traffic the attacker decides to scan per week for w, and refer to e as the *effort vector*. The discreet attacker has a budget B_e for scanning the incoming traffic. In the special case where the scanning effort is negligible ($B_e = \infty$), all our complexity and algorithmic results to be introduced still hold.

We consider an attacker who aims to maximize the expected amount of unaltered flow from target organization O that is scanned by them, as each unit of scanned unaltered flow can lead to a potential success in the social engineering attack, i.e., compromise an employee and discover critical information about O. We model it as a zero-sum game, and therefore the defender's goal is to minimize this amount.

Social engineering is a complex domain which we cannot fully model. However, we build our model and assumptions so that we can formally reason about deception, and even when our assumptions are not met, our work provides a sensible solution. For example, cyber attackers may have tools to circumvent existing deception techniques. Nonetheless, our solution increases the attacker's uncertainty about the environment as they cannot easily obtain or trust the information in the network packets. In appendix, we provide a detailed discussion of the generality and limitations of our work.

4 Computing Optimal Defender Strategy

In this section, we present complexity analysis and algorithms for finding the optimal defender strategy x^* in this game, which is essentially the minimax strategy, i.e., a strategy that minmizes the attacker's maximum possible expected amount of scanned unaltered flow. x^* should be the solution of the following bi-level optimization problem $\mathcal{P}_1$.

$$\mathcal{P}_1 : \min_x \max_{y,e} \quad \sum\nolimits_{w\in W} \kappa_w(1-x_w)e_w \qquad (1)$$

$$\text{s.t.} \quad \sum\nolimits_{w\in W} e_w \le B_e \qquad (2)$$

$$\sum\nolimits_{w\in W} \pi_w y_w \le B_a \qquad (3)$$

$$e_w \le t_w^{all} \cdot y_w, \forall w \in W \qquad (4)$$

$$y_w \in \{0,1\}, \forall w \in W \qquad (5)$$

$$e_w \in [0,\infty), \forall w \in W \qquad (6)$$

$$\sum\nolimits_{w\in W} c_w t_w x_w \le B_d \qquad (7)$$

$$x_w \in [0,1], \forall w \in W \qquad (8)$$

In objective function 1, $\kappa_w = t_w/t_w^{all}$. Since $t_w(1-x_w)$ is the total amount of unaltered flow from the defender organization O and e_w/t_w^{all} is the percentage of incoming traffic that will be scanned, $\kappa_w(1-x_w)e_w$ is the total scanned unaltered traffic to w. Constraint 2-3 describes the budget constraint for the attacker, and Constraint 4 requires that the attacker can only scan traffic for the compromised websites. Constraint 7 is the budget constraint for the defender.

Unfortunately, solving $\mathcal{P}_1$ is challenging. It cannot be solved using any of the existing solvers directly due to the bi-level optimization structure, the mix of real-valued and binary variables and the bilinear terms in the objective function ($x_w e_w$). In fact, even the adversary's best response problem $\mathcal{P}_2(x)$, represented as a mixed integer linear program (MILP) below, is NP-hard as stated in Thm 1. Due to space limit, we defer all the proofs to appendix.[2]

$$\mathcal{P}_2(x) : \quad \max_{y,e} \quad \sum\nolimits_{w\in W} \kappa_w(1-x_w)e_w \qquad (9)$$

$$\text{s.t.} \quad \text{Constraints (2)} \sim \text{(6)} \qquad (10)$$

Theorem 1. *Finding adversary's best response is NP-hard.*

Therefore, we exploit the structure and properties of SED and $\mathcal{P}_1$ and design several novel algorithms to solve it. We first identify two tractable special classes of SED games which can be solved in polynomial time and discuss their real world implications. Then we present CYBERTWEAK, our algorithm for general SED games.

4.1 Tractable Classes

The first tractable class is identified based on the key observation stated in Thm 2: the optimal solutions of SED games exhibit a greedy allocation of the attacker's effort budget. That is, for at most one website w will the attacker spend scanning effort neither zero nor t_w^{all}.

Theorem 2. *Let (x^*, y^*, e^*) be an optimal solution to $\mathcal{P}_1$, $W_F = \{w : e_w^* = t_w^{all}\}, W_Z = \{w : e_w^* = 0\}, W_B = \{w : e_w^* \in (0, t_w^{all})\}$. There is an optimal solution with $|W_B| \le 1$.*

As a result, if the attacker's scanning budget is so limited that he cannot even scan through the traffic of any website, he will use all the scanning effort on one website in the optimal solution. Thus, the optimal defender strategy can be found by enumerating the websites.

Corollary 1. *(Small Effort Budget) If $0 < B_e \le t_w^{all}, \forall w$, the optimal solution can be found in polynomial time.*

The second tractable class roots in the fact that if the scanning effort is negligible (or equivalently, $B_e = \infty$) the attacker only needs to reason about which websites to compromise. Further, if the attacker has a systematic way of compromising a website which makes the cost π_w uniform across websites, then the attacker only needs to greedily choose the websites with the highest unaltered incoming traffic and the defender can greedily alter traffic in the top websites. We provide details about these algorithms in the appendix.

Theorem 3. *(Uniform Cost + Unlimited Effort) If $\pi_w = 1, \forall w \in W$ and $B_e = \infty$, the defender's optimal strategy can be found in polynomial time.*

4.2 CyberTWEAK

For the general SED games, we propose a novel algorithm CYBERTWEAK (Alg 1). It first computes an upper bound for $\mathcal{P}_1$ leveraging the dual problem of the linear program (LP) relaxation of $\mathcal{P}_2(x)$. As a byproduct, the computation provides a heuristic defender strategy $\hat{x}^*$ (Line 2). It then runs an optimality check (Line 3) to see if $\hat{x}^*$ is optimal for $\mathcal{P}_1$. When optimality cannot be verified, it solves the original problem $\mathcal{P}_1$ by converting $\mathcal{P}_1$ to an equivalent LP and applying column generation (Gilmore and Gomory 1961), an iterative approach to compute the optimal strategy (Line 5-8). We further improve the scalability by identifying and eliminating dominated website as pre-processing (Line 1). Next we provide details about these steps.

[2] https://arxiv.org/abs/1901.00586

Algorithm 1: CyberTWEAK

```
Remove $D \leftarrow$ Find-dominated-websites() from $W$.
Get heuristic defender strategy $\hat{x}^*$ by solving $\hat{\mathcal{P}}_1$.
if $OPT(\mathcal{P}_2(\hat{x}^*)) \leq OPT(\tilde{\mathcal{P}}_3(\hat{x}^*))$ then return $\hat{x}^*$
Initialize max effort vector set $e^{\mathcal{A}} = e^{\mathcal{P}_2(\hat{x}^*)}$.
while new max effort vector was added to $e^{\mathcal{A}}$ do
    $x \leftarrow$ solution of $P_1^{\text{LP}}(e^{\mathcal{A}})$.
    $e \leftarrow$ solution of $\mathcal{P}_2(x)$.
    Add $e$ to $e^{\mathcal{A}}$.
```

Upper Bound for $\mathcal{P}_1$ Let $\hat{\mathcal{P}}_2(x)$ be the LP relaxation of $\mathcal{P}_2(x)$ and denote the dual variables of the (relaxed) constraints (2) $\sim$ (5) as $\lambda_1, \lambda_2, \nu, \eta$. We then include the variable x for the defender strategy along with the dual problem, and obtain the minimization problem $\hat{\mathcal{P}}_1$.

$$\hat{\mathcal{P}}_1 : \min_{x,\lambda,\nu,\eta} B_e\lambda_1 + B_a\lambda_2 + \sum\nolimits_{w\in W} \eta_w \tag{11}$$

$$\text{s.t. } \kappa_w(1 - x_w) \leq \lambda_1 + \nu_w, \quad \forall w \in W \tag{12}$$

$$\pi_w\lambda_2 - t_w^{all}\nu_w + \eta_w \geq 0, \qquad \forall w \in W \tag{13}$$

$$\sum\nolimits_{w\in W} c_w t_w x_w \leq B_d \tag{14}$$

$$x_w \in [0, 1],\ \lambda_1, \lambda_2, \nu_w, \eta_w \geq 0,\ \forall w \in W \tag{15}$$

$\hat{\mathcal{P}}_1$ is an LP which can be solved efficiently. In addition, $\hat{x}^*$ in the optimal solution for $\hat{\mathcal{P}}_1$ is a feasible defender strategy in the original problem $\mathcal{P}_1$. Therefore, solving $\hat{\mathcal{P}}_1$ leads to a heuristic defender strategy as well as bounds for the optimal value of $\mathcal{P}_1$. Denote the optimal value of a problem $\mathcal{P}$ as OPT($\mathcal{P}$). We formalize the bounds below.

Theorem 4. *If $B_e \geq \max_w t_w^{all}$, $OPT(\hat{\mathcal{P}}_1) \leq 3OPT(\mathcal{P}_1)$.*

Theorem 5. *Let x^*, $\hat{x}^*$ be an optimal solution to $\mathcal{P}_1$, $\hat{\mathcal{P}}_1$.*

$$\mathit{OPT}(\mathcal{P}_1) \leq \mathit{OPT}(\mathcal{P}_2(\hat{x}^*)) \leq \mathit{OPT}(\hat{\mathcal{P}}_1) \leq \mathit{OPT}(\hat{\mathcal{P}}_2(x^*)).$$

Optimality Conditions for $\hat{x}^*$ We present a sufficient condition for optimality, which leverages the solution of the following LP $\tilde{\mathcal{P}}_3(\hat{x}^*)$.

$$\tilde{\mathcal{P}}_3(\hat{x}^*) : \min_{x,v} \quad v \tag{16}$$

$$\text{s.t.} \quad v \geq \sum\nolimits_{w\in W} \kappa_w(1 - x_w)e_w,\ \forall e \in e^{\mathcal{P}_2(\hat{x}^*)} \tag{17}$$

$$\sum\nolimits_{w\in W} |x_w - \hat{x}^*| \leq \epsilon \tag{18}$$

Constraints (7) $\sim$ (8)

ϵ is an arbitrary positive number and $e^{\mathcal{P}_2(\hat{x}^*)}$ denotes the set of optimal effort vectors in $\mathcal{P}_2(\hat{x}^*)$. The following claim shows the optimality condition.

Claim 1. *Given $\hat{x}^*$, an optimal solution to $\hat{\mathcal{P}}_1$, $\hat{x}^*$ is optimal for $\mathcal{P}_1$ if $OPT(\mathcal{P}_2(\hat{x}^*)) \leq OPT(\tilde{\mathcal{P}}_3(\hat{x}^*))$.*

Clearly, when ϵ is large, $OPT(\tilde{\mathcal{P}}_3(\hat{x}^*))$ is lower and it is harder to satisfy the condition, so in CyberTWEAK, we use a small enough ϵ in $\tilde{\mathcal{P}}_3(\hat{x}^*)$.

Column Generation Define $\hat{e}^{\mathcal{A}}$ as the set of all *max effort vectors* which satisfy $\sum_w e_w = B_e$ and $|W_B| \leq 1$. According to Thm 2, restricting the attacker to only choose strategies from $\hat{e}^{\mathcal{A}}$ will not impact the optimal solution for the defender. As a result, $\mathcal{P}_1$ is equivalent to the following LP, denoted as $\mathcal{P}_1^{\text{LP}}(e^{\mathcal{A}})$, when $e^{\mathcal{A}} = \hat{e}^{\mathcal{A}}$.

$$\mathcal{P}_1^{\text{LP}}(e^{\mathcal{A}}) : \min_{x,v} \quad v \tag{19}$$

$$\text{s.t.} \quad v \geq \sum\nolimits_{w\in W} \kappa_w(1 - x_w)e_w \qquad \forall e \in e^{\mathcal{A}} \tag{20}$$

Constraints (7) $\sim$ (8)

Although existing LP solvers can solve $\mathcal{P}_1^{\text{LP}}(\hat{e}^{\mathcal{A}})$, the order of $\hat{e}^{\mathcal{A}}$ is prohibitively high, leading to poor scalability. Therefore, CyberTWEAK instead uses an iterative algorithm based on the column generation framework to incrementally generate constraints of the LP. Instead of enumerating all of $\hat{e}^{\mathcal{A}}$, we keep a running subset $e^{\mathcal{A}} \subseteq \hat{e}^{\mathcal{A}}$ of max effort vectors and alternate between solving $\mathcal{P}_1^{\text{LP}}(e^{\mathcal{A}})$ (referred to as the master problem) and finding a new max effort vector to be added to $e^{\mathcal{A}}$ (slave problem). In the slave problem, we solve the adversary's best response problem $\mathcal{P}_2(x)$ where x is the latest defender strategy found. This process repeats until no new effort vectors are found for the adversary. Recall that we get $\hat{x}^*$ and $e^{\mathcal{P}_2(\hat{x}^*)}$ when finding upper bound and verifying optimality of $\hat{x}^*$, which can serve as the initial set of strategies for column generation.

Dominated Websites Not all websites are equally valuable for an organization as some are especially lucrative for an adversary to target. In a Polish bank, many employees may visit the Polish Financial Authority website daily, while perhaps a CS conference website is rarely visited by a banker. Intuitively, attackers will not compromise the conference website and thus, the bank may not need to alter traffic to it. Identifying such websites in pre-processing could greatly reduce the size of our problem. A website w is *dominated by another website* u if the attacker would not attack w unless they have used the maximum effort on u, i.e. $e_u = t_u^{all}$, regardless of the defender's strategy. Thm 6 presents sufficient conditions for a website to be dominated and leads to an algorithm (Alg. 6) to find dominated website to be eliminated.

Theorem 6. *Consider websites $u, w \in W$. If the following conditions hold, the website w is dominated by u:*

$$x_u^{max} := B_d/(c_u t_u) \leq 1, \qquad \kappa_w \leq \kappa_u(1 - x_u^{max}),$$
$$\pi_w \geq \pi_u, \qquad t_w^{all} \leq t_u^{all}.$$

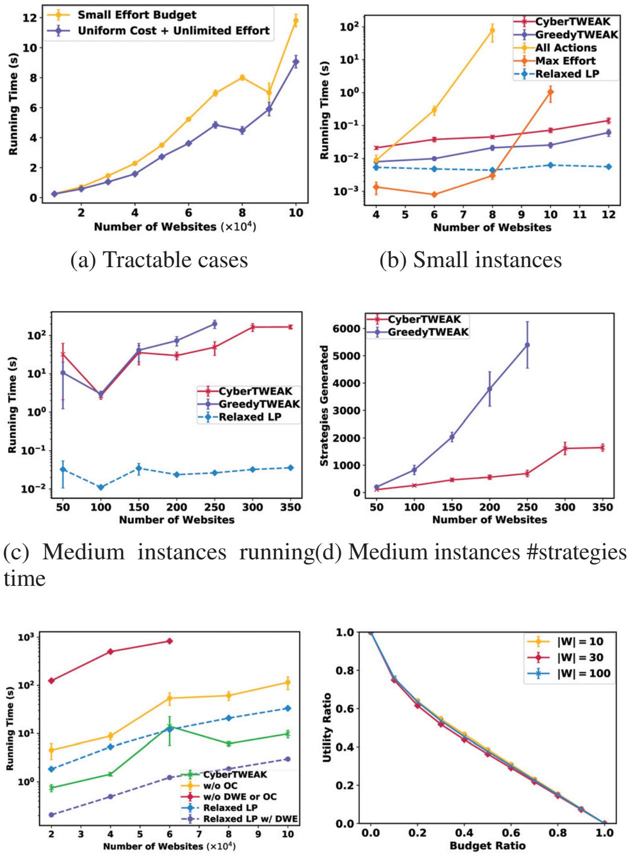
We conclude the section with the following claim.

Claim 2. CyberTWEAK *terminates with optimal solution.*

In light of the hardness of the attacker's best response problem (Thm 1), we also design a variant of CyberTWEAK, which uses a greedy heuristic to find a new max effort vector to be added in each iteration of column generation (denoted as GreedyTWEAK). The algorithm allocates the adversary's budget to websites in decreasing order of $r_w = \kappa_w(1 - x_w)\alpha_w$, where α_w is a tuning parameter. Another variant uses an exact dynamic programming algorithm for the slave problem. Details about these variants can be

Algorithm 2: FIND-DOMINATED-WEBSITES

```
Define $U = \{w \in W : c_w t_w \geq B_d\}$. Let $D = \emptyset$.
Calculate $x_u^{max} = B_d / c_u t_u, \forall u \in U$
foreach website $w \in W$ do
    Set $U_w = \{u \in U : \kappa_w \leq \kappa_u(1 - x_u^{max})\}$
    if exists $U_w^* \subseteq U_w$ such that
        (1) $\sum_{u \in U_w^*} \pi_u \leq \pi_w$, (2) $\sum_{u \in U_w^*} t_u^{all} \geq t_w^{all}$, and
        (3) $\sum_{u \in U_w^*} t_u^{all} \geq B_e$ then $D = D \cup \{w\}$
return set of dominated websites $D$
```



(a) Tractable cases

(b) Small instances

(c) Medium instances running time

(d) Medium instances #strategies

(e) Large instances

(f) Trade-off

Figure 2: Experiment results

found in the appendix. Also, we note that the SED problem is related to the recent work on bi-level knapsack with interdiction (Caprara et al. 2016). However, our outer problem of $\mathcal{P}_1$ is continuous rather than discrete, and the added dimension of adversary's effort makes the inner problem $\mathcal{P}_2(x)$ more complicated than that being studied in this work.

5 Experiments

We developed and tested CyberTWEAK to match the scalability required of large-scale deployment. Unless otherwise noted, problem parameters are described in details in the appendix. All results are averaged over 20 instances; error bars represent standard deviations of the mean.

First, we run experiments on the polynomial time tractable cases (Corollary 1 and Theorem 3). Fig. 2a shows that in both cases, our solution can easily handle 10^5 websites, applicable to real-world corporate-scale problems.

Moving on to the general SED games, we test 3 algorithms (CyberTWEAK, GreedyTWEAK, and RelaxedLP) with two other baselines, MaxEffort and AllActions. RelaxedLP refers to solving $\hat{\mathcal{P}}_1$. MaxEffort solves $\mathcal{P}_1^{\text{LP}}(\hat{e}^{\mathcal{A}})$ directly without column generation. AllActions decomposes SED into subproblems, each assuming some adversary's effort vector is a best response. Its details can be found in the appendix. We test the algorithms with different problem scales. In small and medium sized instances, we skip dominated website elimenateion (DWE) step (Line 6) and optimality check (OC) step (Line 3) in Alg. 1 as the problem size is small enough, making these steps unnecessary. We use solid lines to represent methods with optimality guarantee and dotted lines for others (RelaxedLP based methods).

For small instances (Fig. 2b), both baselines become impractical even on problems with less than 12 websites. However, CyberTWEAK is able to find the optimal solutions rather efficiently. GreedyTWEAK slightly improves over CyberTWEAK. RelaxedLP yields the fastest running time, despite a solution gap above 6% as shown in Table 1.

For medium-sized instances (Fig. 2c), baseline algorithms cannot run and GreedyTWEAK stops being helpful, mainly because the "better" effort vectors generated in GreedyTWEAK far outnumbers the "best" effort vectors in CyberTWEAK (Fig. 2d) despite the saved time in each iteration. Relaxed LP has negligible running time and often solves the problem optimally (Table 1).

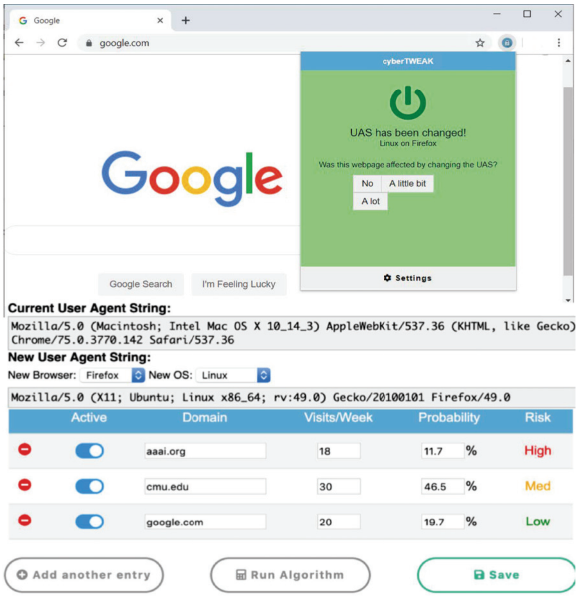
For large instances (Fig. 2e), CyberTWEAK with both DWE and OC steps is able to handles 10^5 websites in 10 seconds. When we remove (denoted as "w\o") DWE and/or OC step, runtime increases significantly, showing the efficacy of these steps.[3] Compared to RelaxedLP or RelaxedLP enhanced with DWE step, which can also efficiently handles 10^5 websites, CyberTWEAK has optimality guarantee.

Finally, we consider the trade-off between the risk exposure and degradation in rendering websites, represented by the objective $OPT(\mathcal{P}_1)$ and defender's budget B_d, respectively. With budget $\bar{B}_d = \sum_{w \in W} c_w t_w$, the attacker would have zero utility. With zero defender budget, the attacker would get maximum utility $\bar{U}$. Fig. 2f shows how the utility ratio $OPT(\mathcal{P}_1)/\bar{U}$ changes with the budget ratio $B_d/\bar{B}_d$. As the organization increases the tolerance for service degradation, its risk exposure drops at a decreasing rate.

[3] The impact of DWE varies significantly across instances and relies heavily on the distribution of traffic. In less than 4 of the 20 instances DWE did not reduce the problem size by much. We report in Fig. 2e the majority group where DWE eliminated a significant number of websites. We provide further discussion in the appendix.

$\lvert W\rvert$	Gap	# Exact	$\lvert W\rvert$	Gap	# Exact
4	13.19%	2/20	150	7e-8	16/20
8	8.11%	5/20	200	8e-10	19/20
12	6.63%	8/20	250	0	20/20
50	2e-6	18/20	300	2e-3	17/20
100	8e-9	19/20	350	2e-8	18/20

Table 1: Solution quality of RELAXEDLP, with the number of instances where RELAXEDLP solves the problem exactly.



UAS has been changed!
Linux on Firefox

Was this webpage affected by changing the UAS?

No | A little bit | A lot

Current User Agent String:

```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/75.0.3770.142 Safari/537.36
```

New User Agent String:

New Browser: Firefox New OS: Linux

```
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox/49.0
```

Active	Domain	Visits/Week	Probability	Risk
●	aaai.org	18	11.7 %	High
●	cmu.edu	30	46.5 %	Med
●	google.com	20	19.7 %	Low

Figure 3: Screenshots of the browser extension

6 Deployment

Based on CYBERTWEAK, we developed a browser extension (available on the Google Chrome Web Store[1]). It can modify the user-agent string sent to websites automatically during browsing which contains information such as the operating system, browser, and services running on the user's machine. The extension receives from the user the websites visited W, number of visits per week t_w, the cost to alter the user-agent string c_w and budget B_d. The total traffic t_w^{all} and attack cost π_w are estimated from the Cisco Umbrella 1 Million list (Cisco 2019). The attacker's budgets are set in scale with the previously mentioned parameters. The extension runs CYBERTWEAK to set the probability of altering the user-agent string for each website. Note that it is the relative magnitudes, rather than the exact values, that matter.

The extension takes additional steps to make our algorithm more usable and interpretable. First, some users may find it hard to specify the cost of altering user-agent string c_w and budget B_d. Our extension will adjust the values based on the qualitative feedback provided by users about whether the degradation of the website's rendering is acceptable when they visit a website using the modified user-agent, as shown in Fig. 3. Second, in addition to showing the computed altering probabilities, the extension also displays a personalized "risk level" for each website, to help the user understand the algorithm's output. Less popular websites frequented more often by the user have higher risk, as shown in Fig. 3.

As mentioned in Section 3, advanced cyber attackers might sometimes circumvent the existing deception methods. Future versions of the extension will leverage the latest advances in anti-fingerprinting techniques, which entail manipulating more than the user-agent string.

We believe this CYBERTWEAK extension is vital to the continued study and development of the countermeasure we develop for this domain and large scale deployments.

Acknowledgments

Co-authors Z. R. Shi and F. Fang are supported in part by the U.S. Army Combat Capabilities Development Command Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA).

References

Agari. 2016. *Email Security: Social Engineering Report*.

Albanese, M.; Battista, E.; and Jajodia, S. 2016. Deceiving attackers by creating a virtual attack surface. In *Cyber Deception*. Springer. 167–199.

Caprara, A.; Carvalho, M.; Lodi, A.; and Woeginger, G. J. 2016. Bilevel knapsack with interdiction constraints. *INFORMS Journal on Computing*.

Cisco. 2019. *Cisco Umbrella Popularity List*.

Durkota, K.; Lisỳ, V.; Bosanskỳ, B.; and Kiekintveld, C. 2015. Optimal network security hardening using attack graph games. In *IJCAI*.

Farquhar, D. 2017. *Watering hole attack prevention*.

Gilmore, P. C., and Gomory, R. E. 1961. A linear programming approach to the cutting-stock problem. *Operations research* 9(6):849–859.

Jajodia, S.; Park, N.; Pierazzi, F.; Pugliese, A.; Serra, E.; Simari, G. I.; and Subrahmanian, V. 2017. A probabilistic logic of cyber deception. *IEEE Transactions on Information Forensics and Security* 12(11):2532–2544.

Laszka, A.; Vorobeychik, Y.; and Koutsoukos, X. 2015. Optimal personalized filtering against spear-phishing attacks. In *AAAI*.

Mitnick, K. D., and Simon, W. L. 2011. *The art of deception*. John Wiley & Sons.

Parliament. 2018. *Watering Hole Attacks*.

Píbil, R.; Lisỳ, V.; Kiekintveld, C.; Bošanskỳ, B.; and Pěchouček, M. 2012. Game theoretic model of strategic honeypot selection in computer networks. In *GameSec*. Springer.

Schlenker, A.; Thakoor, O.; Xu, H.; Tambe, M.; Vayanos, P.; Fang, F.; Tran-Thanh, L.; and Vorobeychik, Y. 2018. Deceiving cyber adversaries: A game theoretic approach. In *AAMAS*.

Sutton, M. 2014. *How to protect against watering hole attacks*.

Symantec. 2017. *Attackers target dozens of global banks with new malware*.