The AAAI-17 Workshop on
Artificial Intelligence for Cyber Security
WS-17-04

A Stackelberg Game Model for Botnet Traffic Exfiltration

Thanh H. Nguyen, Michael P. Wellman, Satinder Singh
University of Michigan, Ann Arbor
{thanhhng,wellman,baveja}@umich.edu

Abstract

Cyber-criminals can distribute malware to control computers on a networked system and leverage these compromised computers (i.e., botnets) to perform their malicious activities inside the network. Botnet-detection mechanisms, based on a detailed analysis of network traffic characteristics, provide a basis for defense against botnet attacks. In this work, we formulate the botnet defense problem as a Stackelberg security game, allocating detection resources to deter botnet attacks taking into account the strategic response of cyber-criminals. Based on the new game model, we propose a game-theoretic algorithm, **ORANI**, to compute an optimal detection resource allocation strategy in zero-sum game settings. Our algorithm employs the double-oracle method to deal with an exponential number of players' actions. Furthermore, we provide greedy heuristics to approximately compute an equilibrium of these botnet defense games. Finally, we conduct extensive experiments based on both simulated and real-world network topologies to demonstrate advantages of our game-theoretic solution compared to previously proposed defense policies.

Introduction

Cyber-criminals intent on denial-of-service, spam dissemination, data theft, or other information security breaches often pursue their attacks with the assistance of *botnets*: collections of compromised computers subject to their control (Holz, Engelberth, and Freiling 2009; Peng, Leckie, and Ramamohanarao 2007; Stone-Gross et al. 2009). In 2014 testimony, the US Federal Bureau of Investigation cited over nine billion dollars of US losses and $110 billion losses globally (Demarest 2014). The estimated 500 million computers infected globally each year by botnet activities amounts to 18 victims per second.

The threat of botnets has drawn significant attention from network security researchers (Bacher et al. 2005; Choi et al. 2007; Cooke, Jahanian, and McPherson 2005; Feily, Shahrestani, and Ramadass 2009; Gu et al. 2008; 2007; Gu, Zhang, and Lee 2008; Strayer et al. 2008). Much existing work focuses on detection mechanisms to identify compromised computers based on network traffic characteristics. For example, BotSniffer (Gu, Zhang, and Lee 2008) searches for spatial-temporal patterns in network traffic characteristic of coordinated botnet behavior. Given some underlying detection capability, it remains to design defenses against botnet attacks that effectively deploys detection resources. For example, Venkatesan et al. consider the problem of allocating a limited number of localized detection resources on a network in order to maximally disrupt botnet data exfiltration attacks, where the botnet aims to transfer stolen information out of the network. Their first solution (Venkatesan, Albanese, and Jajodia 2015) allocated resources statically, which could effectively disrupt one-time attacks but is vulnerable to adaptive attackers. They extended this method to randomize detector placement dynamically to improve robustness against adaptation (Venkatesan et al. 2016). In a related work, Mc Carthy et al. (2016) address the additional challenge of imperfect botnet detection.

Our work extends these prior efforts by formulating the botnet defense problem as a Stackelberg security game, thus accounting for the strategic response of attackers to deployed defenses. In our botnet defense game, the defender attempts to protect traffic traversing within a computer network by allocating detection resources (detectors). The attacker eavesdrops on network traffic, and attempts to exfiltrate the stolen information by transferring it outside the defender's network. We assume that the source bot routes the stolen information along a single path designated by the attacker, and term this the *uni-exfiltration* setting.

Given our game model, we propose a new algorithm to compute optimal defense strategies, named **ORANI** (**O**ptimal **R**esource **A**llocation for u**N**i-exfiltration **I**nterception). We adopt the double-oracle methodology (McMahan, Gordon, and Blum 2003) to overcome the computational challenges of exponential strategy spaces for both players. Our main algorithmic contributions lie in defining mixed-integer linear programs (MILPs) for the defender's and attacker's best-response oracles. In addition, we introduce greedy heuristics to approximately implement these oracles. Finally, we conduct experiments based on both simulated and real-world network topologies to evaluate solution quality as well as runtime performance of our new game-theoretic algorithms, demonstrating significant improvements over previous defense strategies.

Related Work

Recent work has introduced game-theoretic models and/or corresponding defense solutions for various botnet detection and/or prevention problems (Bensoussan, Kantarcioglu, and Hoe 2010; Kolokoltsov and Bensoussan 2015; Soper and Musacchio 2014; Soper 2015). In these game models, cyber criminals intrude by compromising computers in a network. Users or owners of computers in the network defend by patching or replacing their computers based on alerts of potential security threats.

Stackelberg security games have been successfully applied for solving many real-world physical security problems (Fang et al. 2016; Basilico, Gatti, and Amigoni 2009; Letchford and Vorobeychik 2011; Shieh et al. 2012; Tambe 2011). Jain et al. (2011) address a problem in urban network security with some analog to uni-exfiltration, as the attacker follows a single path to attack its best target in an urban road network. Vaněk et al. (2012) tackles a problem of malicious packet prevention in computer networks, assuming the attacker only determines which entry point to access to a network to attack a specific target in that network while the corresponding traversing path is fixed. In our botnet defense problem, cyber criminals can not only determine which computers to compromise but can also create an overlaying network over these compromised computers to exfiltrate traffic from multiple targets in the network. The action complexities of cyber criminals in our game model lead to a totally different and difficult security problem to solve.

Botnet Defense Game Model

Our game model is based upon the botnet model presented by Venkatesan et al. (2015). Specifically, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ represents a computer network where $\mathbf{V}$ is the set of nodes which refer to network elements such as routers and end hosts,and $\mathbf{E}$ is the set of edges connecting these nodes. Let $\mathbf{V^h} \subseteq \mathbf{V}$ be a set of mission-critical end hosts and $\mathbf{P^h}$ be a set of pairs of these end hosts that need to exchange data via the network. Traffic between any two nodes is routed by a routing algorithm fixed by the network system. For each pair of nodes (u, v) in the network, we denote by $\mathbf{P}(u, v)$ the *routing path* between u and v.

Definition 1 (Mission-critical Nodes)**.** *Given a network* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *and a set of pairs of mission-critical end hosts* $\mathbf{P^h}$ *that need to exchange data, the set of* mission-critical nodes*, denoted by* $\mathbf{V^c}$*, consists of all the nodes in the network that lie on routing paths between any pair of nodes in* $\mathbf{P^h}$*:*

$$\mathbf{V^c} = \bigcup\nolimits_{(u,v)\in\mathbf{P^h}} \mathbf{P}(u, v) \tag{1}$$

We model the botnet defense problem as a *Stackelberg security game* (SSG). In such a game, the defender commits to a mixed (randomized) strategy to allocate limited security resources to protect important targets. The attacker then optimizes its attack action with respect to the distribution of defender allocations. In our context, the defender is the security controller of a computer network, with a limited set of available detection resources. The defender aims at deploying these resources in the most effective way to detect and thereby impede the attack chosen in response to the defender's deployment strategy.

The attacker in this game is a cyber criminal who eavesdrops on network traffic. Compromising a node on the routing path $\mathbf{P}(u, v)$ enables the attacker to eavesdrop on traffic between u and v. Compromising other nodes in the network helps the attacker to relay the eavesdropped traffic to a server he controls, S^a. The attacker evades the defender's detectors to the best of his ability through compromised nodes. Note that the attacker can determine a sequence of compromised nodes to relay eavesdropped traffic, yet the traffic exchange between any pair of consecutive bots in the chain must still follow fixed routing paths specified by the network. The attacker successfully exfiltrates traffic from an eavesdropping bot b to S^a if and only if (iff) the defender has no detectors on routing paths between consecutive compromised nodes on the chain used for relaying traffic. We call this chain of ordered bots and nodes on routing paths between consecutive bots an *exfiltration path*, denoted by $\pi(b, S^a)$.

Definition 2 (Exfiltration Prevention)**.** *Given a network* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *and a set of mission-critical nodes* $\mathbf{V^c}$*, for each eavesdropping bot* b*, traffic exfiltration from* b *is impeded by the defender iff there is a detector on the exfiltration path* $\pi(b, S^a)$*.*

In our game model, the attacker's remote server S^a is located outside the network. The defender is aware of which nodes in the network can potentially communicate with the attacker's server. The defender randomizes the allocation of detection resources so that locations of detectors become unpredictable to the attacker. Given the defender strategy, the attacker chooses an optimal response consisting of a set of compromised nodes and a set of exfiltration paths over these compromised nodes.

Definition 3 (Strategy Space)**.** *In our game model, the strategy spaces of the players are defined as follows:*

Defender's strategy space: *The defender can deploy up to* $K^d < |\mathbf{V}|$ *detection resources. We denote by* $\mathbf{D} = \{\mathbf{D}_i \mid \mathbf{D}_i \subseteq \mathbf{V}, |\mathbf{D}_i| \leq K^d\}$ *the set of all pure strategies of the defender, each is an allocation of detectors over the nodes. Let* $\mathbf{x} = \{x_i\}$ *be a mixed strategy of the defender where* x_i *is the probability that the defender plays* $\mathbf{D}_i$ *such that* $\sum_i x_i = 1$ *and forall* i*,* $x_i \in [0, 1]$*.*

Attacker's strategy space: *The attacker can compromise up to* $K^a < |\mathbf{V}|$ *nodes in the network. We denote by* $\mathbf{A} = \{\mathbf{A}_j = (\mathbf{B}_j, \mathbf{\Pi}_j) \mid \mathbf{B}_j \subseteq \mathbf{V}, |\mathbf{B}_j| \leq K^a, \mathbf{\Pi}_j = \{\pi_j(b, S^a),\ b \in \mathbf{B}_j \cap \mathbf{V^c}\}\}$ *the set of all pure strategies of the attacker. In particular, each pure strategy* $\mathbf{A}_j$ *consists of* $\mathbf{B}_j$ *a set of compromised nodes and* $\mathbf{\Pi}_j$ *a set of all exfiltration paths over* $\mathbf{B}_j$*.*

A simple example of a botnet defense game is shown in Figure 1. Our work mainly focuses on computing an optimal mixed strategy of the defender in zero-sum game settings.

Definition 4 (Payoff in Zero-sum Game)**.** *Traffic of every pair* $(u, v) \in \mathbf{P^h}$ *is associated with a value, denoted by* $r(u, v) > 0$*, which implies the importance of the traffic. If the attacker successfully exfiltrates traffic of* $(u, v) \in \mathbf{P^h}$*, he obtains a reward* $r(u, v)$ *while the defender receives a*

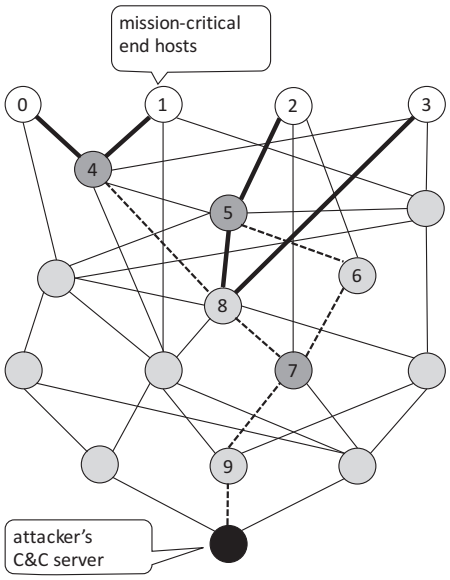


Figure 1: In this example, two pairs of mission-critical end hosts which need to exchange data are $\mathbf{P^h} = \{(0,1),(2,3)\}$. These two pairs of hosts has to exchange data via fixed routing paths $(0 \rightarrow 4 \rightarrow 1)$ and $(2 \rightarrow 5 \rightarrow 8 \rightarrow 3)$ respectively. Thus, the set of mission-critical nodes is $\mathbf{V^c} = \{0,4,1,2,5,8,3\}$. Suppose the attacker can compromise three nodes at most, then a pure-strategy example of the attacker $\mathbf{A}_j$ can be: (i) a set of compromised nodes $\mathbf{B}_j = \{4,5,7\}$; and (ii) a set of exfiltration paths $\mathbf{\Pi}_j = \{\pi_j(4), \pi_j(5)\}$ to exfiltrate traffic from eavesdropping bots 4 and 5 to the attacker's server S^a. These exfiltration paths $\pi_j(4) = \mathbf{P}(4,7) \cup \mathbf{P}(7,S^a)$ and $\pi_j(5) = \mathbf{P}(5,7) \cup \mathbf{P}(7,S^a)$ relay eavesdropped traffic via relaying bot 7, where $\mathbf{P}(4,7) = (4 \rightarrow 8 \rightarrow 7)$, $\mathbf{P}(5,7) = (5 \rightarrow 6 \rightarrow 7)$ and $\mathbf{P}(7,S^a) = (7 \rightarrow 9 \rightarrow S^a)$ are routing paths predetermined by the network system. Suppose the defender has only one detection resource, then if the defender allocates this detector on node 8, the attacker fails at exfiltrating traffic from node 4 since $8 \in \pi_j(4)$ while succeeding at node 5 since $8 \notin \pi_j(5)$.

penalty $-r(u,v)$. Conversely, if the defender deploys a detector on the exfiltration path $\pi(b,S^a)$ of the attacker's eavesdropping bot b, both players receive a payoff of zero.

We can now determine the payoff of the players if the players play $(\mathbf{D}_i, \mathbf{A}_j)$. In particular, the defender receives a payoff $U^d(\mathbf{D}_i, \mathbf{A}_j)$ and the attacker receives a payoff $U^a(\mathbf{D}_i, \mathbf{A}_j)$ which are determined as follows:

$$U^d(\mathbf{D}_i, \mathbf{A}_j) = -\sum_{(u,v)\in\mathbf{P^h}} r(u,v)I(u,v) \tag{2}$$

$$U^a(\mathbf{D}_i, \mathbf{A}_j) = \sum_{(u,v)\in\mathbf{P^h}} r(u,v)I(u,v) \tag{3}$$

where $I(u,v)$ implies whether the attacker successfully exfiltrates the traffic of the pair of the mission-critical end hosts

Algorithm 1: ORANI Algorithm Overview

```
1 Initialize the sets of pure strategies: A = {A_j} and
    D = {D_i} for some j and i;
2 repeat
3 |   (x*, a*) = MaximinCore(D, A);
4 |   D_o = DefenderOracle(a*);
5 |   A_o = AttackerOracle(x*);
6 |   A = A ∪ {A_o}, D = D ∪ {D_o}
7 until converge;
```

$(u,v) \in \mathbf{P^h}$ or not, which is determined as follows:

$$I(u,v) = \begin{cases} 1 & \text{if } \exists b \in \mathbf{B}_j \text{ s.t. } b \in \mathbf{P}(u,v) \text{ and } \mathbf{D}_i \cap \pi_j(b,S^a) = \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The expected utility the players receive when the defender plays $\mathbf{x}$ while the attacker plays $\mathbf{A}_j$ is computed as follows:

$$U^d(\mathbf{x}, \mathbf{A}_j) = \sum_i x_i U^d(\mathbf{D}_i, \mathbf{A}_j) \tag{5}$$

$$U^a(\mathbf{x}, \mathbf{A}_j) = \sum_i x_i U^a(\mathbf{D}_i, \mathbf{A}_j) \tag{6}$$

In botnet defense games, the defender chooses a mixed strategy $\mathbf{x}$, and the attacker chooses a best response against $\mathbf{x}$. A defender mixed strategy that maximizes her utility given the attacker plays a best response and breaks tie in favor of the defender constitutes a *Strong Stackelberg Equilibrium* (SSE) of the game. Given the new game model, our second main contribution lies on providing a new algorithm, **ORANI**, for computing an optimal defense strategy in zero-sum games.

Overview of ORANI Algorithm

In zero-sum games, any defender's SSE strategy is also a Maximin strategy (Yin et al. 2010). Therefore, finding an optimal defense mixed strategy is formulated as follows:

$$\max_{\mathbf{x}} U_*^d \tag{7}$$

$$\text{s.t. } U_*^d \le U^d(\mathbf{x}, \mathbf{A}_j),\ \forall j \tag{8}$$

$$\sum\nolimits_i x_i = 1,\ x_i \in [0,1],\ \forall i. \tag{9}$$

where U_*^d is the defender's utility for playing a mixed strategy $\mathbf{x}$ that we aim to maximize. Constraint (8) ensures the attacker will choose an optimal action against $\mathbf{x}$, leading to the lowest utility for the defender. Essentially, solving (7–9) is computationally expensive due to an exponential number of pure strategies of the defender and the attacker involved. To overcome this computational challenge, **ORANI** applies the double oracle method – a commonly-used standard method to solve massive zero-sum games (Jain et al. 2011; McMahan, Gordon, and Blum 2003). The general overview of **ORANI** is sketched in Algorithm 1.

Essentially, our **ORANI** algorithm starts by solving a maximin sub-game of (7–9) by considering only a small subset of pure strategies for the defender $\mathbf{D}$ and the attacker $\mathbf{A}$ (Line 3). By solving this sub-game, we obtain a solution

of the defender and attacker's mixed strategies $(\mathbf{x}^*, \mathbf{a}^*)$ w.r.t the current $(\mathbf{D}, \mathbf{A})$. Then **ORANI** iteratively adds new best pure strategies $\mathbf{D}_o$ and $\mathbf{A}_o$ to the current strategy sets in the defender and the attacker's oracles respectively (Lines (4–6)). These strategies $\mathbf{D}_o$ and $\mathbf{A}_o$ are chosen such that maximizing the defender's and the attacker's utility against the attacker and the defender's current mixed strategy $\mathbf{a}^*$ and $\mathbf{x}^*$ respectively. This iteration process continues until the solution converges (i.e., no new pure strategy can be added to improve the defender and the attacker's utilities). Previous work showed that once the double oracle converges, the current solution of the algorithm must be an equilibrium of the game (McMahan, Gordon, and Blum 2003). Following this general algorithmic framework, our main contributions of **ORANI** lies on providing new MILPs to solve the attacker and the defender oracle in botnet defense games.

ORANI: Attacker Oracle

The attacker oracle attempts to find a new pure strategy for the attacker that maximizes his utility against the current mixed strategy of the defender $\mathbf{x}^*$ returned by Maximin-Core. We first represent a MILP to solve the attacker oracle problem and then show that the problem is NP-hard.

Attacker Strategy Representation

We first parameterize each pure strategy of the attacker as follows: (i) we represent a set of compromised nodes of the attacker using *compromising* variables $\mathbf{z} = \{z_w\}$ where $w \in \mathbf{V}$, indicating if the attacker compromises that node ($z_w = 1$) or not ($z_w = 0$); and (ii) we represent exfiltration paths via *path-exfiltration* variables $\mathbf{q} = \{q_c(u,v)\}$ where $u \in \mathbf{V}$ and $v \in \mathbf{V} \cup \{S^a\}$ which refer to the exfiltration path from the mission-critical node $c \in \mathbf{V^c}$ to the attacker server S^a. In particular, when $z_c = z_u = z_v = 1$, $q_c(u,v) = 1$ if the exfiltration path from the eavesdropping bot $c \in \mathbf{V^c}$ includes the routing path $\mathbf{P}(u,v)$ between two bots (u,v). Otherwise, $q_c(u,v) = 0$. Note that $q_c(u,v) = 0$ for all (u,v,c) where either c or u or v is not compromised.

We now represent required constraints on the strategy variables $(\mathbf{z}, \mathbf{q})$ as shown in (10–15) where constraints (10–12) enforce that there is only a single exfiltration path for exfiltrating traffic from the mission-critical node $c \in \mathbf{V^c}$ to the attacker's server S^a if node c is compromised ($z_c = 1$). In particular, when $z_c = 1$, constraint (10) indicates that there is a single out-exfiltration path from node c and constraint (11) imposes that there is only a single in-exfiltration path to the attacker's server S^a. Otherwise, when c is not compromised $z_c = 0$, there is no exfiltration path from c. Constraint (12) ensures the flow reservation condition to hold w.r.t to the mission-critical node c. In other words, the total number of in-exfiltration paths to a node v must be equal to the total number of out-exfiltration paths from that node. Furthermore, constraints (13–14) guarantee that exfiltration paths are determined upon compromised nodes only (i.e., if either $z_u = 0$ or $z_v = 0$, then $q_c(u,v) = 0$). Finally, constraint (15) ensures that the total number of compromised nodes does not exceed the attacker's resource limit, K^a.

$$\sum_{u \in \mathbf{V} \cup \{S^a\} \setminus \{c\}} q_c(c,u) = z_c, \ \forall c \in \mathbf{V^c} \tag{10}$$

$$\sum\nolimits_{u \in \mathbf{V}} q_c(u, S^a) = z_c, \ \forall c \in \mathbf{V^c} \tag{11}$$

$$\sum_{u \in \mathbf{V} \setminus \{v\}} q_c(u,v) = \sum_{w \in \mathbf{V} \cup \{S^a\} \setminus \{v\}} q_c(v,w), \tag{12}$$
$$\forall v \in \mathbf{V} \setminus \{c\}, \ \forall c \in \mathbf{V^c}$$

$$q_c(u,v) \leq z_u, \forall u \in \mathbf{V}, v \in \mathbf{V} \cup \{S^a\} \setminus \{c\}, c \in \mathbf{V^c} \tag{13}$$

$$q_c(u,v) \leq z_v, \forall u \in \mathbf{V}, v \in \mathbf{V} \setminus \{c\}, c \in \mathbf{V^c} \tag{14}$$

$$\sum\nolimits_{z \in \mathbf{V}} z_w \leq K^a, \ \forall w \in \mathbf{V} \tag{15}$$

Traffic Exfiltration Representation

Given the attacker's pure strategy $(\mathbf{z}, \mathbf{q})$, we then introduce *traffic-exfiltration* variables $\mathbf{h} = \{h_i(c)\}$, implying if the attacker can successfully exfiltrate traffic from the mission-critical node $c \in \mathbf{V^c}$ (i.e., $h_i(c) = 1$) or not (i.e., $h_i(c) = 0$) given the defender's strategy $\mathbf{D}_i \in \mathbf{D}$. Essentially, given an eavesdropping bot $c \in \mathbf{V^c}$ (i.e., $z_c = 1$), then $h_i(c) = 1$ only when the defender does not deploys a detector on the exfiltration path from node c to the attacker's server S^a. Otherwise, $h_i(c) = 0$. Therefore, the required constraints for $\mathbf{h} = \{h_i(c)\}$ can be represented as:

$$h_i(c) \leq 1 - q_c(u,v), \ \forall c \in \mathbf{V^c}, u \in \mathbf{V}, v \in \mathbf{V} \cup \{S^a\} \setminus \{u,c\},$$
$$\text{and } \forall i \text{ s.t. } \mathbf{P}(u,v) \cap \mathbf{D}_i \neq \emptyset \tag{16}$$

$$h_i(c) \leq z_c, \forall c \in \mathbf{V^c}, \forall i. \tag{17}$$

where constraint (16) enforces that the attacker can not successfully exfiltrate traffic from the mission-critical node $c \in \mathbf{V^c}$ (i.e., $h_i(c) = 0$) if the defender's pure strategy $\mathbf{D}_i$ deploys a detector on the exfiltration path of c (i.e., $q_c(u,v) = 1$ and $\mathbf{P}(u,v) \cap \mathbf{D}_i \neq \emptyset$ for some (u,v)). Furthermore, constraint (17) implies that the traffic exfiltration of node c is considered only when that node is compromised. In other words, $h_i(c)$ is forced to be zero if $z_c = 0$.

Attacker Reward Representation

Finally, we introduce *reward* variables $\mathbf{I} = \{I_i(u,v)\}$ where $(u,v) \in \mathbf{P^h}$, implying if the attacker successfully exfiltrates the traffic between two mission-critical end hosts $(u,v) \in \mathbf{P^h}$ (i.e., $I_i(u,v) = 1$) or not ($I_i(u,v) = 0$) given that the defender plays the pure strategy $\mathbf{D}_i$ and the attacker plays $(\mathbf{z}, \mathbf{q})$. In particular, given the exfiltration variables $\mathbf{h} = \{h_i(c)\}$, the reward variables $\mathbf{I} = \{I_i(u,v)\}$ with $(u,v) \in \mathbf{P^h}$ are determined as follows:

$$I_i(u,v) = \begin{cases} 1 & \text{if } \exists c \in \mathbf{P}(u,v) \text{ s.t. } h_i(c) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Therefore, we have the following required constraints for the attacker reward variables:

$$I_i(u,v) \leq \sum_{c \in \mathbf{P}(u,v)} h_i(c), \ \forall (u,v) \in \mathbf{P^h}, \ \forall i \tag{19}$$

which implies that the attacker can not successfully exfiltrate traffic of the pair $(u,v) \in \mathbf{P^h}$ if he fails at exfiltrating from any eavesdropping bots on the routing path $\mathbf{P}(u,v)$. Specifically, $I_i(u,v) = 0$ when $h_i(c) = 0 \ \forall c \in \mathbf{P}(u,v)$.

Algorithm 2: Attacker Greedy Heuristic

Initialize set of compromised nodes $\mathbf{B^c} = \emptyset$, set of exfiltration paths $\mathbf{\Pi} = \emptyset$, attacker utility greedyU $= 0$;
repeat
  for $u \in \mathbf{V} \setminus \mathbf{B^c}$ **do**
    $(\mathbf{\Pi}(u), \text{utility}(u)) = \text{optExPaths}(\mathbf{x}^*, \mathbf{B^c} \cup \{u\})$;
    if *utility*$(u) >$ *greedyU* **then**
      greedyU $= \text{utility}(u)$;
      $\mathbf{\Pi} = \mathbf{\Pi}(u)$;
      $u^* = u$;
  $\mathbf{B^c} = \mathbf{B^c} \cup \{u^*\}$;
until $|\mathbf{B^c}| = K^a$;

MILP Representation

Considering the reward variables $\mathbf{I} = \{I_i(u,v)\}$, given that the defender plays $\mathbf{x}^*$ and the attacker plays $(\mathbf{z}, \mathbf{q})$, the attacker obtains a utility which is computed as follows:

$$U^a(\mathbf{x}^*, (\mathbf{z}, \mathbf{q})) = \sum_{\mathbf{D}_i \in \mathbf{D}} x_i \sum_{(u,v) \in \mathbf{P^h}} r(u,v) I_i(u,v) \quad (20)$$

The problem of computing an optimal pure strategy for the attacker can be now formulated as the following MILP:

$$\max_{\mathbf{z},\mathbf{q},\mathbf{h},\mathbf{I}} U^a(\mathbf{x}^*, (\mathbf{z}, \mathbf{q})) \quad (21)$$

$$\text{s.t. } \textit{Attacker strategy constraints (10–15)} \quad (22)$$

$$\textit{Traffic exfiltration constraints (16–17)} \quad (23)$$

$$\textit{Attacker reward constraints (19)} \quad (24)$$

$$q_c(u,v) \in [0,1], I_i(u,v) \in [0,1], \forall u, v, c, i \quad (25)$$

$$z_w \in \{0,1\}, h_i(c) \in \{0,1\}, \forall w, c, i. \quad (26)$$

where only $\mathbf{z} = \{z_w\}$ and $\mathbf{h} = \{h_i(c)\}$ are required to be binary (constraint (26)).

Theorem 1. *The MILP (21–26) returns an optimal pure strategy for the attacker against* $\mathbf{x}^*$.[1]

Finally, we obtain Proposition 1. The detail of the proof is in the Online Appendix C. Next, we introduce a new greedy heuristic to approximately solve the attacker oracle.

Proposition 1. *The attacker oracle problem is NP-hard.*

Attacker Greedy Heuristic

The general idea of the attacker greedy heuristic is outlined in Algorithm 2. Essentially, the heuristic iteratively adds the next best node to compromise to the current set of compromised nodes until the number of compromised nodes reaches the resource limit K^a. In particular, at each iteration, given the current set of compromised nodes $\mathbf{B^c}$, the greedy heuristic searches over all uncompromised nodes $u \in \mathbf{V} \setminus \mathbf{B^c}$ of the network to find the best next node for the attacker to compromise such that his utility is maximized. The core part of the greedy algorithm is the optExPaths$(\mathbf{x}^*, \mathbf{B^c} \cup \{u\})$ method which determines the optimal exfiltration paths given the compromised set $\mathbf{B^c} \cup \{u\}$ and the defender strategy $\mathbf{x}^*$.

Overall, the problem of finding an optimal set of exfiltration paths for the attacker given a set of compromised nodes $\mathbf{B^c} \cup \{u\}$ and the defender's strategy $\mathbf{x}^*$ can be represented as a MILP which is a simplification of (21–26) in which the compromising variables $\mathbf{z} = \{z_w\}$ are no longer needed. Furthermore, the path-exfiltration variables, traffic-exfiltration variables, and reward variables are now defined only upon the current set of compromised nodes $\mathbf{B^c} \cup \{u\}$ instead of the whole node set $\mathbf{V}$. As a result, the total number of variables and constraints involved in the resulting MILP is reduced significantly. Finally, although the attacker greedy heuristic helps in reducing the computational time of the attacker oracle, the core part of the heuristic, optExPaths$(\mathbf{x}^*, \mathbf{B^c} \cup \{u\})$, remains NP-hard.

ORANI: Defender Oracle

The defender oracle attempts to find a new defense pure strategy which maximizes the defender utility against the current mixed strategy of the attacker $\mathbf{a}^* = \{a_j^*\}$ returned by MaximinCore. Here, a_j^* is the probability that the attacker will follow the pure strategy $\mathbf{A}_j$ such that $\sum_j a_j^* = 1, a_j^* \in [0,1]$. We first present a new MILP to solve this defender oracle and then show that the problem is NP-hard.

MILP Representation. We first parameterize a pure strategy of the defender using *detection* variables $\mathbf{z} = \{z_w\}$ where $w \in \mathbf{V}$. In particular, $z_w = 1$ if the defender deploys a detector on node w. Otherwise, $z_w = 0$. In addition, given that the attacker plays $\mathbf{A}_j$ and the defender plays $\mathbf{z}$, we introduce *reward* variables $\mathbf{I} = \{I_j(u,v)\}$ where $(u,v) \in \mathbf{P^h}$, implying whether the attacker successfully exfiltrates the traffic of (u,v) (i.e., $I_j(u,v) = 1$) or not ($I_j(u,v) = 0$). Then the following condition is required:

$$I_j(u,v) = \begin{cases} 1 & \text{if } \exists c \in \mathbf{B}_j \text{ s.t. } c \in \mathbf{P}(u,v) \text{ and} \\ & z_w = 0 \text{ for all } w \in \pi_j(c, S^a) \\ 0 & \text{otherwise} \end{cases} \quad (27)$$

Considering the reward variables $\mathbf{I} = \{I_j(u,v)\}$, given that the attacker plays $\mathbf{a}^*$ and the defender plays $\mathbf{z}$, the defender obtains a utility which is computed as follows:

$$U^d(\mathbf{z}, \mathbf{a}^*) = -\sum_{\mathbf{A}_j \in \mathbf{A}} a_j^* \sum_{(u,v) \in \mathbf{P^h}} r(u,v) I_j(u,v) \quad (28)$$

Then the problem of finding an optimal pure defense strategy that maximizes the defender's utility against the attacker's strategy $\mathbf{a}^*$ can be formulated as the MILP (29–32).

$$\max_{\mathbf{z},\mathbf{I}} U^d(\mathbf{z}, \mathbf{a}^*) \quad (29)$$

$$\text{s.t. } I_j(u,v) \geq 1 - \sum\nolimits_{w \in \pi_j(c, S^a)} z_w, \quad (30)$$

$$\forall c \in \mathbf{P}(u,v) \cap \mathbf{B}_j, \ \forall (u,v) \in \mathbf{P^h}, \ \forall j$$

$$\sum\nolimits_{w \in \mathbf{V}} z_w \leq K^d, \ z_w \in \{0,1\}, \ \forall w \in \mathbf{V} \quad (31)$$

$$I_j(u,v) \in [0,1], \ \forall j, \ \forall (u,v) \in \mathbf{P^h}. \quad (32)$$

[1] The proof of the theorem is in the Online Appendix B: https://www.dropbox.com/s/s5lpoxt4mwn822p/appendix.pdf?dl=0

In (29–32), only $\mathbf{z} = \{z_w\}$ are required to be binary. In particular, constraint (30) ensures that $I_j(u, v) = 1$ when the attacker successfully exfiltrates from an eavesdropping bot $c \in \mathbf{P}(u, v) \cap \mathbf{B}_j$ (i.e., the defender does not deploy a detector on the exfiltration path of that bot). On the other hand, since the MILP attempts to maximize the defender's utility (Equation 29) which is a monotonically decreasing function of $I_j(u, v)$, then any MILP solver will automatically force $I_j(u, v) = 0$ if possible given constraint (32). Finally, constraints (31) guarantee that the number of resources for the defender does not exceed the limit K^d.

We provide Proposition 2 w.r.t the complexity of the defender oracle. Its proof is in the Online Appendix D.

Proposition 2. *The defender oracle problem is NP-hard.*

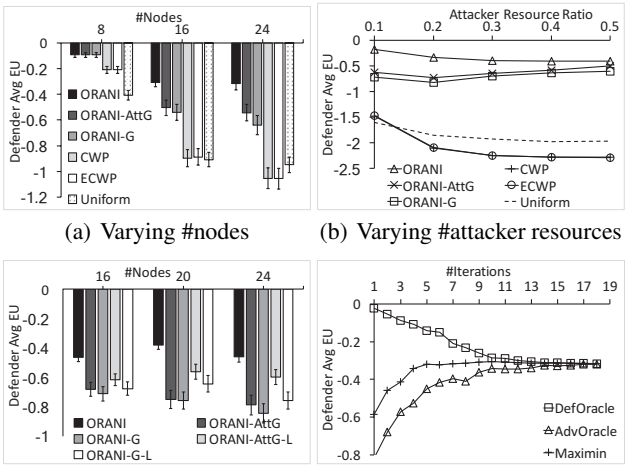
Defender Greedy Heuristic. We introduce a new defender greedy heuristic to approximately solve the defender oracle in polynomial time. The overview of this greedy heuristic is similar to Algorithm 2. Essentially, given the attacker's mixed strategy $\mathbf{a}^*$ and an initially empty set of monitored nodes $\mathbf{D^c}$, the greedy heuristic iteratively adds the next best node to monitor to the set $\mathbf{D^c}$ until $|\mathbf{D^c}| = K^d$. At each iteration, given the current set of monitored nodes $\mathbf{D^c}$, the greedy heuristic searches over all unmonitored nodes $u \in \mathbf{V} \backslash \mathbf{D^c}$ of the network to find the best next node to monitor such that the defender's utility is maximized. Since computing the defender's utility given a set of monitored nodes and the attacker's strategy $\mathbf{a}^*$ is polynomial (Equations 2 and 4), our defender greedy heuristic is of polynomial time.

Experiments

We aim at evaluating both solution quality and runtime performance of our algorithms. We conduct experiments based on two different datasets: (i) synthetic network topology — we use JGraphT (Naveh and Contributors 2009), a free Java graph library, to randomly generate scale-free graphs used in our experiments since many real-world network topologies exhibit the power-law property (Faloutsos, Faloutsos, and Faloutsos 1999); and (ii) real-world network topology — we derive different network topologies from the Rocketfuel network topology dataset (Rocketfuel 2002). Each data point in our results is averaged over 50 different samples of network topologies. We compare six different algorithms: 1) **ORANI** – both exact oracles; 2) **ORANI-AttG** – exact defender oracle and greedy attacker oracle; 3) **ORANI-G** – both greedy oracles; 4& 5) CWP & ECWP – heuristics proposed by (Venkatesan et al. 2016) to generate a defense mixed strategy based on the centrality values of nodes in the network; and 6) Uniform – generating a uniform defense mixed strategy. We consider CWP, ECWP, and Uniform as the three baseline algorithms.

Synthetic Network Topology

In the first experiment (Figure 2(a)), we examine solution quality of the algorithms with varying graph size. In particular, in Figure 2(a), the x-axis is the number of nodes in each graph. The y-axis is the averaged expected utility of the defender obtained by the evaluated algorithms. The number of defender resources, K^d, and of attacker resources, K^a,



(a) Varying #nodes (b) Varying #attacker resources

(c) Greedy enhancement (d) Iteration convergence

Figure 2: Evaluations on random scale-free graphs

are chosen to be 10% and 15% of the graph nodes respectively. The number of pairs of end hosts to exchange data is $10\% \times |\mathbf{V}|$ and the traffic value associated with each pair is generated uniformly at random within $[0, 1]$. Intuitively, the higher averaged expected utility an algorithm gets, the better the solution quality of the algorithm is. Figure 2(a) shows that all of our algorithms, **ORANI**, **ORANI-AttG**, **ORANI-G** defeat the baseline algorithms in obtaining a significant higher utility for the defender. Figure 2(a) also shows that **ORANI-AttG** and **ORANI-G** obtain a lower defender's utility compared to **ORANI** as expected. Nevertheless, we show later that the greedy heuristics help in reducing the solving time of double oracle.

In our second experiment (Figure 2(b)), we fix $|\mathbf{V}| = 20$ and $K^d = 20\% \times |\mathbf{V}|$ while varying the number of attacker resources, K^a. We aim at examining the solution quality of our algorithms in terms of the defender's expected utility when K^a increases. In Figure 2(b), the y-axis is the ratio of the attacker resources to the graph size. Figure 2(b) shows that when K^a increases, the decrease in defender's expected utility obtained by **ORANI** is small. In addition, Figure 2(b) shows the gap in the defender' utility between **ORANI** and **ORANI-AttG** is getting smaller when K^a increases. This result implies that the attacker greedy heuristic can provide a solution for the attacker oracle closer to the optimal one w.r.t the increase in K^a. As a result, **ORANI-AttG** reaches to a closer optimal defense strategy.

Based on this finding, we anticipate that the solution quality of **ORANI-AttG** and **ORANI-G** can be enhanced for any game instance by conservatively assuming a larger K^a than the true K^a. Indeed, our third experimental result (Figure 2(c)) confirms our anticipation. In this experiment, we set $K^a = 20\% \times |\mathbf{V}|$. In addition to **ORANI**, **ORANI-AttG**, **ORANI-G**, we evaluate the other two algorithms, called **ORANI-AttG-L** and **ORANI-G-L**, which find a defense strategy assuming a larger $K^a = 40\% \times |\mathbf{V}|$. In Figure 2(c), the x-axis is the number of nodes in the network and the y-axis is the defender's expected utility on av-

erage against the attacker's best response with $K^a = 20\% \times |\mathbf{V}|$. Figure 2(c) clearly shows that both **ORANI-AttG-L** and **ORANI-G-L** obtain a higher utility for the defender compared to **ORANI-AttG** and **ORANI-G** respectively.

In our fourth experiment (Figure 2(d)), we examine the convergence of the double oracle used in **ORANI**. The x-axis is the number of iterations of adding new strategies for both players until convergence. In addition, the y-axis is the average of the defender's expected utility at each iteration w.r.t the defender oracle, the attacker oracle, and the Maximin core. Figure 2(d) shows that **ORANI** converges quickly after approximately 19 iterations. This result shows there is only a small set of pure strategies of players involved in the game equilibrium even though there is an exponential number of strategies in total. In addition, **ORANI** can find this set of pure strategies after a small number of iterations.

Lastly, in our fifth experiment (Figure 3), we investigate the runtime performance. In Figure 3, the x-axis is the number of nodes in the graphs and the y-axis is the runtime on average in hundred seconds. As expected, **ORANI**'s runtime grows exponentially when the number of nodes increases. By using the greedy heuristics, **ORANI-AttG** and **ORANI-G** run faster compared to **ORANI**.

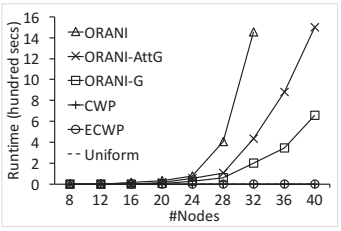


Figure 3: Runtime performance

Real-world Network Topology

Our second set of experiments is conducted on real-world network topologies from the Rocket-fuel dataset (Rocketfuel 2002). Overall, the dataset provides router-level topologies of 10 different ISP networks: Telstra, Sprintlink, Ebone, Verio, Tiscali, Level3, Exodus, VSNL, Abovenet, and AT&T. In this set of experiments, we mainly focus on evaluating the solution quality of our algorithms compared with the three baseline algorithms. For each of our experiments, we randomly sample 50 24-node sub-graphs from every network topology using random walk. In addition, we assume that all external routers located outside the ISP can potentially route traffic to the attacker's server. Each data point in our experimental results is averaged over 50 different graph samples. The defender's averaged expected utility obtained by the evaluated algorithms is shown in Table 1.

Table 1 shows that all of our algorithms obtain a significantly higher expected utility for the defender than the three baseline algorithms. Moreover, our greedy-based algorithms: **ORANI-AttG** and **ORANI-G** are shown to consistently perform well on all the ISP network topologies compared to the optimal one: **ORANI**. For example, the defender's utility obtained by **ORANI-G** is only $\approx 7\%$ lower than **ORANI** on average over the 10 network topologies.

Dataset	ORANI	ORANI -AttG	ORANI -G	CWP	ECWP	Uniform
Telstra	-0.59	-0.66	-0.68	-1.08	-1.10	-1.08
Sprintlink	-0.55	-0.55	-0.58	-1.06	-1.06	-1.09
Ebone	-0.71	-0.75	-0.77	-0.96	-0.97	-0.89
Verio	-0.48	-0.49	-0.51	-0.92	-0.93	-0.95
Tiscali	-0.75	-0.79	-0.80	-1.10	-1.10	-1.05
Level3	-0.75	-0.76	-0.82	-1.01	-1.01	-0.94
Exodus	-0.83	-0.85	-0.88	-1.06	-1.07	-0.99
VSNL	-0.86	-0.88	-0.90	-1.11	-1.11	-1.02
Abovenet	-0.82	-0.85	-0.89	-1.14	-1.14	-1.06
AT&T	-0.40	-0.41	-0.42	-0.96	-0.94	-0.97

Table 1: Evaluations on real-world network topologies

Conclusion

Many computer networks have been suffered from botnet data exfiltration attacks, leading to a significant research emphasis on botnet defense. In this work, we introduce a new Stackelberg game model for the botnet defense problem, taking into account the strategic response of cyber criminals to deployed defenses. Given our game model, we propose a new double oracle based algorithm, **ORANI**, to compute an optimal defense strategy to allocate limited detection resources on computers in a network to deter botnet traffic exfiltration. We also provide new greedy heuristics to approximately solve the defender and the attacker best-response oracles of **ORANI**. We conduct extensive experiments based on both random scale-free graphs and 10 real-world ISP network topologies, demonstrating the significant advances of our game theoretic solution compared to existing work.

Acknowledgements: This work was supported in part by MURI grant W911NF-13-1-0421 from the US Army Research Office.

References

Bacher, P.; Holz, T.; Kotter, M.; and Wicherski, G. 2005. Know your enemy: Tracking botnets. Technical report.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 57–64.

Bensoussan, A.; Kantarcioglu, M.; and Hoe, S. C. 2010. A game-theoretical approach for finding optimal strategies in a botnet defense model. In *GameSec*, 135–148. Springer.

Choi, H.; Lee, H.; Lee, H.; and Kim, H. 2007. Botnet detection by monitoring group activities in dns traffic. In *CIT*, 715–720. IEEE.

Cooke, E.; Jahanian, F.; and McPherson, D. 2005. The zombie roundup: Understanding, detecting, and disrupting botnets. *SRUTI* 5:6–6.

Demarest, J. 2014. Taking down botnets. Statement before the Senate Judiciary Committee, Subcommittee on Crime and Terrorism.

Faloutsos, M.; Faloutsos, P.; and Faloutsos, C. 1999. On power-law relationships of the internet topology. In *ACM SIGCOMM CCR*, volume 29, 251–262. ACM.

Fang, F.; Nguyen, T. H.; Pickles, R.; Lam, W. Y.; Clements, G. R.; An, B.; Singh, A.; Tambe, M.; and Lemieux, A. 2016.

Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *IAAI*.

Feily, M.; Shahrestani, A.; and Ramadass, S. 2009. A survey of botnet and botnet detection. In *SECURWARE*, 268–273. IEEE.

Gu, G.; Porras, P. A.; Yegneswaran, V.; Fong, M. W.; and Lee, W. 2007. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Usenix Security*, volume 7, 1–16.

Gu, G.; Perdisci, R.; Zhang, J.; Lee, W.; et al. 2008. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *USENIX Security Symposium*, volume 5, 139–154.

Gu, G.; Zhang, J.; and Lee, W. 2008. Botsniffer: Detecting botnet command and control channels in network traffic.

Holz, T.; Engelberth, M.; and Freiling, F. 2009. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *ESORICS*, 1–18. Springer.

Jain, M.; Korzhyk, D.; Vaněk, O.; Conitzer, V.; Pěchouček, M.; and Tambe, M. 2011. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, 327–334.

Kolokoltsov, V., and Bensoussan, A. 2015. Mean-field-game model for botnet defense in cyber-security. *arXiv preprint arXiv:1511.06642*.

Letchford, J., and Vorobeychik, Y. 2011. Computing randomized security strategies in networked domains. *Applied Adversarial Reasoning and Risk Modeling* 11:06.

Mc Carthy, S. M.; Sinha, A.; Tambe, M.; and Manadhata, P. 2016. Data exfiltration detection and prevention: Virtually distributed POMDPs for practically safer networks. In *Seventh Conference on Decision and Game Theory for Security*.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *ICML*, 536–543.

Naveh, B., and Contributors. 2009. JGraphT – a free java graph library.

Peng, T.; Leckie, C.; and Ramamohanarao, K. 2007. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Computing Surveys (CSUR)* 39(1):3.

Rocketfuel. 2002. Rocketfuel: an ISP topology mapping engine.

Shieh, E.; An, B.; Yang, R.; Tambe, M.; Baldwin, C.; DiRenzo, J.; Maule, B.; and Meyer, G. 2012. PROTECT: A deployed game theoretic system to protect the ports of the United States. In *AAMAS*.

Soper, B., and Musacchio, J. 2014. A botnet detection game. In *Allerton*, 294–303. IEEE.

Soper, B. C. 2015. Non-zero-sum, adversarial detection games in network security.

Stone-Gross, B.; Cova, M.; Cavallaro, L.; Gilbert, B.; Szydlowski, M.; Kemmerer, R.; Kruegel, C.; and Vigna, G. 2009. Your botnet is my botnet: Analysis of a botnet takeover. In *ACM CCS*, 635–647.

Strayer, W. T.; Lapsely, D.; Walsh, R.; and Livadas, C. 2008. Botnet detection based on network behavior. In *Botnet Detection*. Springer. 1–24.

Tambe, M., ed. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

Vaněk, O.; Yin, Z.; Jain, M.; Bošanskỳ, B.; Tambe, M.; and Pěchouček, M. 2012. Game-theoretic resource allocation for malicious packet detection in computer networks. In *AAMAS*, 905–912.

Venkatesan, S.; Albanese, M.; and Jajodia, S. 2015. Disrupting stealthy botnets through strategic placement of detectors. In *IEEE CNS*, 95–103.

Venkatesan, S.; Albanese, M.; Cybenko, G.; and Jajodia, S. 2016. A moving target defense approach to disrupting stealthy botnets. In *ACM*.

Yin, Z.; Korzhyk, D.; Kiekintveld, C.; Conitzer, V.; and Tambe, M. 2010. Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS*, 1139–1146.