A Stackelberg Game Model for Botnet Data Exfiltration

Thanh 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 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. We formulate the botnet defense problem as a zero-sum Stackelberg security game, allocating detection resources to deter botnet attacks taking into account the strategic response of cyber-criminals. We model two different botnet data-exfiltration scenarios, representing exfiltration on single or multiple paths. Based on the game model, we propose algorithms to compute an optimal detection resource allocation strategy with respect to these formulations. Our algorithms employ the double-oracle method to deal with the exponential action spaces for attacker and defender. Furthermore, we provide greedy heuristics to approximately compute an equilibrium of these botnet defense games. Finally, we conduct experiments based on both synthetic and real-world network topologies to demonstrate advantages of our game-theoretic solution compared to previously proposed defense policies.

1 Introduction

Cyber-criminals intent on denial-of-service, spam dissemination, data theft, or other information security breaches often pursue their attacks with *botnets*: collections of compromised computers (bots) subject to their control [14, 23, 30, 31, 33]. In 2014 testimony, the US Federal Bureau of Investigation cited over $9 billion of US losses and $110 billion losses globally due to botnet activities [7]. 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 [1, 5, 6, 10, 11, 12, 13, 32]. Much existing work focuses on detection mechanisms to identify compromised computers based on network traffic characteristics. For example, BotSniffer [13] searches for spatial-temporal patterns in network traffic characteristic of coordinated botnet behavior. Given some underlying detection capability, the defender faces the problem of how to effectively deploy its detection resources against potential botnet attacks. 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 *data exfiltration* attacks, where the botnet aims to transfer stolen information out of the network [38]. Their first solution 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 [37]. In a related work, Mc Carthy et al. address the additional challenge of imperfect botnet detection [20].

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 data within a computer network by allocating detection resources (*detectors*). The attacker compromises computers in the network to steal data, and attempts to exfiltrate the stolen data by transferring it outside the defender's network. We consider two formulations of data exfiltration: (i) *uni-exfiltration*, where the source bot routes the stolen data along a single path designated by the attacker; and (ii) *broad-exfiltration*, where each bot propagates the received stolen data to all other bots in the network.

We propose algorithms to compute defense strategies for these data exfiltration formulations: **ORANI** (**O**ptimal **R**esource **A**llocation for u**N**i-exfiltration **I**nterception) and **ORABI** (**O**ptimal **R**esource **A**llocation for **B**road-exfiltration **I**nterception). Both **ORANI** and **ORABI** employ the double-oracle method [21] to control exploration of the exponential strategy spaces available to attacker and defender. Our main algorithmic contributions lie in defining mixed-integer linear programs (MILPs) for the defender and attacker's best-response oracles. In addition, we introduce greedy heuristics to approximately implement these oracles. Finally, we conduct experiments based on both synthetic and real-world network topologies to evaluate solution quality as well as runtime performance of our game-theoretic algorithms, demonstrating significant improvements over previous defense strategies.

2 Related Work

Prior studies of botnet security tend to focus on designing botnet detection mechanisms [1, 5, 6, 10, 11, 12, 13, 32] or advanced botnet designs against these detection mechanisms [29, 39]. Some studies provide empirical and statistical analysis on related cybersecurity implications such as the role of Internet service providers in botnet mitigation [35] or contagion in cyber attacks [2].

Recent work has introduced game-theoretic models and corresponding defense solutions for various botnet detection and prevention problems [4, 17, 27, 28]. In these 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 to many real-world physical security problems [3, 9, 19, 26, 34]. Jain et al. address a problem in urban network security partially analogous to uni-exfiltration, as the attacker follows a single path to attack its best target in an urban road network [15]. Vaněk et al. tackle a problem of malicious packet prevention, where the attacker determines which entry point to access a network to attack a specific target assuming the corresponding traversing path is fixed [36]. In our botnet defense problem, cyber-criminals decide not only which computers to compromise but also create an overlay network over these bots to exfiltrate data from multiple targets in the network. The additional complexity of considering the exfiltration plan leads to a distinct and difficult security problem.

3 Game Model: Uni-Exfiltration

Our game model for uni-exfiltration is built on the botnet model introduced by Venkatesan et al. [38]. Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ represent a computer network where the set of nodes $\mathbf{V}$ comprises network elements such as routers and end hosts, and edges in $\mathbf{E}$ connect these nodes. We denote by $\mathbf{V^c}$ a set of *mission-critical nodes* in the network which contain sensitive data. Data exchange is governed by a routing algorithm fixed by the network system. For each pair of nodes (u, v), we denote by $\mathbf{P}(u, v)$ the *routing path* between u and v. In our experiments, we assume that routing is via the shortest path.

We model the botnet defense problem as a *Stackelberg security game* (SSG) [16]. In such a game, the defender commits to a mixed (randomized) strategy to allocate limited security resources to protect important targets. The attacker then optimally chooses targets with respect to the distribution of defender allocations. In our context, the defender is the security controller of a computer network, with limited detection resources. The defender attempts to deploy its detectors in the most effective way to impede the attack chosen in response.

The attacker in the botnet exfiltration game is a cyber-criminal who attempts to steal sensitive network data. Compromising a mission-critical node $c \in \mathbf{V^c}$ enables the attacker to steal data owned by c. Compromising other nodes in the network helps the attacker to relay the stolen data to a server S^a outside the network, which he controls. The attacker specifies a sequence of compromised nodes (bots) to relay stolen data. Routing between consecutive bots in the sequence follows fixed paths out of the attacker's control. We call this chain of ordered bots and nodes on routing paths between consecutive bots an *exfiltration path*, denoted by $\pi(c, S^a)$.

Definition 1 (Exfiltration Prevention). *Given a network $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and a set of mission-critical nodes $\mathbf{V^c}$, data exfiltration from $c \in \mathbf{V^c}$ is* prevented *by the defender iff there is a detector on the exfiltration path $\pi(c, S^a)$.*

Though the attacker's remote server S^a is located outside the network, we assume the defender is aware of which nodes in the network can relay data to S^a.

In our Stackelberg game model, the defender moves first by allocating detection resources, and the attacker responds with a plan for compromise and exfiltration to evade detection. The defender placement of detectors is randomized, so any attack plan succeeds with some probability.

Definition 2 (Strategy Space). *The strategy spaces of the players are as follows:*

Defender: *The defender has $K^d < |\mathbf{V}|$ detection resources available for deployment on network nodes. 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 defense strategies *of the defender. Let $\mathbf{x} = \{x_i\}$ be a* mixed strategy *of the defender where $x_i \in [0, 1]$ is the probability that the defender plays $\mathbf{D}_i$, and $\sum_i x_i = 1$.*

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

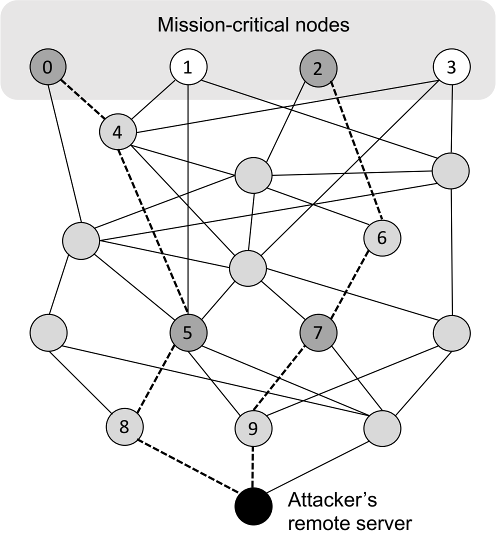


Fig. 1. An example scenario of the botnet exfiltration game. There are four mission-critical nodes, $\mathbf{V^c} = \{0, 1, 2, 3\}$. If $K^a = 4$, then a possible pure strategy of the attacker $\mathbf{A}_j$ can be: (i) a set of compromised nodes $\mathbf{B}_j = \{0, 2, 5, 7\}$; and (ii) a set of exfiltration paths $\mathbf{\Pi}_j = \{\pi_j(0), \pi_j(2)\}$ to exfiltrate data from stealing bots 0 and 2 to the attacker's server S^a. These exfiltration paths $\pi_j(0) = \mathbf{P}(0, 5) \cup \mathbf{P}(5, S^a)$ and $\pi_j(2) = \mathbf{P}(2, 7) \cup \mathbf{P}(7, S^a)$ relay stolen data via relaying bots 5 and 7 respectively, where $\mathbf{P}(0, 5) = (0 \to 4 \to 5)$, $\mathbf{P}(5, S^a) = (5 \to 8 \to S^a)$, $\mathbf{P}(2, 7) = (2 \to 6 \to 7)$ and $\mathbf{P}(7, S^a) = (7 \to 9 \to S^a)$ are routing paths fixed by the network system. Suppose $K^d = 1$. If the defender allocates its one detector to node 9, the attacker fails at exfiltrating data from node 2 since $9 \in \pi_j(2)$ but succeeds from node 0 since $9 \notin \pi_j(0)$.

A simple scenario of the botnet defense game is shown in Figure 1. The model specification is completed by defining the payoff structure, which we take to be zero-sum.

Definition 3 (Game Payoff). *Each mission-critical node $c \in \mathbf{V^c}$ is associated with a value, $r(c) > 0$, representing the importance of data stored at that node. Successfully exfiltrating data from c yields the attacker a payoff $r(c)$, and the defender receives a payoff $-r(c)$. For prevented exfiltrations, both players receive zero.*

Note that the maximum achievable payoff for a defender is zero, obtained by preventing all exfiltration attempts. In general terms, let $U^a(\mathbf{D}_i, \mathbf{A}_j)$ denote the payoff to the attacker if the defender plays $\mathbf{D}_i$ and the attacker plays $\mathbf{A}_j$. Since the game is zero-sum, the defender payoff $U^d(\mathbf{D}_i, \mathbf{A}_j) \equiv -U^a(\mathbf{D}_i, \mathbf{A}_j)$. The payoff can be decomposed across mission-critical nodes,

$$U^a(\mathbf{D}_i, \mathbf{A}_j) \equiv \sum_{c \in \mathbf{V^c}} r(c)h(c), \tag{1}$$

where $h(c)$ indicates whether the attacker successfully exfiltrates the data of the mission-critical node $c \in \mathbf{V^c}$. This is determined as follows:

$$h(c) = \begin{cases} 1 & \text{if } c \in \mathbf{B}_j \text{ and } \mathbf{D}_i \cap \pi_j(c, S^a) = \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The expected utility for the attacker when the defender plays mixed-strategy $\mathbf{x}$ is

$$U^a(\mathbf{x}, \mathbf{A}_j) = \sum\nolimits_i x_i U^a(\mathbf{D}_i, \mathbf{A}_j),$$

which is negated to obtain the expected defender payoff $U^d(\mathbf{x}, \mathbf{A}_j)$. A defender mixed strategy that maximizes $U^d(\mathbf{x}, \mathbf{A}_j)$ given the attacker plays a best response and breaks ties in favor of the defender constitutes a *Strong Stackelberg Equilibrium* (SSE) of the game.

4 ORANI: An Algorithm for Uni-Exfiltration Games

In zero-sum games, the first mover's SSE strategy is also a maximin strategy [18]. Therefore, finding an optimal mixed defense strategy can be formulated as follows:

$$\max\nolimits_{\mathbf{x}} U^d_* \tag{3}$$

$$\text{s.t. } U^d_* \leq U^d(\mathbf{x}, \mathbf{A}_j), \ \forall j \tag{4}$$

$$\sum\nolimits_i x_i = 1, \ x_i \geq 0, \ \forall i, \tag{5}$$

where U^d_* is the defender's utility for playing mixed strategy $\mathbf{x}$ when the attacker best-responds. Constraint (4) ensures the attacker chooses an optimal action against $\mathbf{x}$, that is, $U^d_* = \min_j U^d(\mathbf{x}, \mathbf{A}_j) = \max_j U^a(\mathbf{x}, \mathbf{A}_j)$. Solving (3–5) is computationally expensive due to the exponential number of pure strategies of the defender and the attacker. To overcome this computational challenge, **ORANI** applies the *double-oracle* method [15, 21]. Algorithm 1 presents a sketch of **ORANI**.

ORANI starts by solving a maximin sub-game of (3–5) by considering only small seed subsets $\mathbf{D}$ and $\mathbf{A}$ of pure strategies for the defender and attacker (Line 3). Solving this sub-game yields a solution $(\mathbf{x}^*, \mathbf{a}^*)$ with respect to the strategy subsets. **ORANI** iteratively adds new best pure strategies $\mathbf{D}_o$ and $\mathbf{A}_o$ to the current strategy sets $\mathbf{D}$ and $\mathbf{A}$ (Lines 4–6). These strategies $\mathbf{D}_o$ and $\mathbf{A}_o$ are chosen by the oracles to maximize the defender and attacker utility, respectively, against the current (in iteration) counterpart solution strategies $\mathbf{a}^*$ and $\mathbf{x}^*$. This iterative process continues until the solution *converges*: when no new pure strategy can be added to improve the defender and the attacker's utilities. At convergence, the latest solution $(\mathbf{x}^*, \mathbf{a}^*)$ an equilibrium of the game [21]. Following this general methodology, the specific contribution of **ORANI** is in defining MILPs representing the attacker and the defender oracle problems in botnet exfiltration games.

Algorithm 1: ORANI Algorithm Overview

```
Initialize the sets of pure strategies: $\mathbf{A} = \{\mathbf{A}_j\}$ and $\mathbf{D} = \{\mathbf{D}_i\}$ for some $j$ and $i$;
repeat
    $(\mathbf{x}^*, \mathbf{a}^*) = \text{MaximinCore}(\mathbf{D}, \mathbf{A})$;
    $\mathbf{D}_o = \text{DefenderOracle}(\mathbf{a}^*)$;
    $\mathbf{A}_o = \text{AttackerOracle}(\mathbf{x}^*)$;
    $\mathbf{A} = \mathbf{A} \cup \{\mathbf{A}_o\}, \mathbf{D} = \mathbf{D} \cup \{\mathbf{D}_o\}$
until converge;
```

4.1 ORANI Attacker Oracle

The attacker oracle returns a pure strategy for the attacker maximizing utility against a given defender mixed strategy $\mathbf{x}^*$. Below, we present a MILP exactly representing the attacker oracle and show that the problem is NP-hard. We then provide a greedy heuristic to approximately solve the attacker oracle problem.

MILP Representation We parameterize each pure strategy of the attacker as follows:

1. *bot* variables $\mathbf{z} = \{z_w \mid w \in \mathbf{V}\}$, indicate whether the attacker compromises node w ($z_w = 1$) or not ($z_w = 0$), and
2. *bot-chain* variables $\mathbf{q} = \{q_c(u, v) \mid c \in \mathbf{V^c},\, u \in \mathbf{V},\, v \in \mathbf{V} \cup \{S^a\} \setminus \{c, u\}\}$, represent exfiltration paths.

For each stealing bot c, $\{q_c(u, v)\}$ represents the bot chain to exfiltrate data from c to S^a. Note that the bot-chain variables employ compromised nodes only. This means that $q_c(u, v) = 0$ for all (c, u, v) such that $z_c = 0$ or $z_u = 0$ or $z_v = 0$. Conversely, when $z_c = z_u = z_v = 1$, $q_c(u, v) = 1$ iff (u, v) are consecutive bots in the bot chain for c. This entails that the exfiltration path $\pi(c, S^a)$ includes the routing path $\mathbf{P}(u, v)$.

Given the attacker's pure strategy $(\mathbf{z}, \mathbf{q})$, we introduce *data-exfiltration* variables $\mathbf{h} = \{h_i(c)\}$ to describe the outcome of the attack. For stealing bot $c \in \mathbf{V^c}$ with $z_c = 1$, $h_i(c)$ indicates whether the attacker successfully exfiltrates from c when the defender plays $\mathbf{D}_i \in \mathbf{D}$. Specifically, $h_i(c) = 0$ if $\mathbf{D}_i$ includes a detector on the exfiltration path from node c to S^a. Otherwise, $h_i(c) = 1$. The attacker utility can be computed based on $\mathbf{h} = \{h_i(c)\}$,

$$U^a(\mathbf{x}^*, (\mathbf{z}, \mathbf{q})) = \sum_{\mathbf{D}_i \in \mathbf{D}} x_i \sum_{c \in \mathbf{V^c}} r(c) h_i(c).$$

The optimization problem for the attacker can now be formulated as a MILP (6–15). Variables $\mathbf{z}$ and $\mathbf{h}$ are constrained to be binary. Constraints (7–9) enforce that there is only a single exfiltration path from each mission-critical node $c \in \mathbf{V^c}$ to S^a if node c is compromised ($z_c = 1$). In particular, when $z_c = 1$, constraint (7) indicates that there is a single out-exfiltration path from node c and constraint (8) 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 (9) ensures, for each $c \in \mathbf{V^c}$, that the total number of in-exfiltration paths to any node v equals the total number of out-exfiltration paths from that node. Constraints (10) and (11) guarantee that exfiltration paths are determined using compromised nodes only (i.e., if either $z_u = 0$ or $z_v = 0$, then $q_c(u, v) = 0$). Constraint (12) ensures that the number of compromised nodes does not exceed the attacker's resource limit, K^a. Finally, constraint (13) enforces $h_i(c) = 0$ when $\mathbf{P}(u, v) \cap \mathbf{D}_i \neq \emptyset$ for some pair of consecutive bots (u, v) on the exfiltration path from c (i.e., such that $q_c(u, v) = 1$). Constraint (14) ensures $h_i(c) = 0$ when c is not compromised.

$$\max_{\mathbf{z},\mathbf{q},\mathbf{h}} U^a(\mathbf{x}^*, (\mathbf{z}, \mathbf{q})) \tag{6}$$

$$\text{s.t.} \sum_{u \in \mathbf{V} \cup \{S^a\} \setminus \{c\}} q_c(c, u) = z_c, \forall c \in \mathbf{V^c} \tag{7}$$

$$\sum_{u \in \mathbf{V}} q_c(u, S^a) = z_c, \forall c \in \mathbf{V^c} \tag{8}$$

$$\sum_{u \in \mathbf{V} \setminus \{v\}} q_c(u, v) = \sum_{w \in \mathbf{V} \cup \{S^a\} \setminus \{v, c\}} q_c(v, w), \forall c \in \mathbf{V^c}, v \in \mathbf{V} \setminus \{c\} \tag{9}$$

$$q_c(u, v) \leq z_u, \forall c \in \mathbf{V^c}, u \in \mathbf{V}, v \in \mathbf{V} \cup \{S^a\} \setminus \{c, u\} \tag{10}$$

$$q_c(u, v) \leq z_v, \forall c \in \mathbf{V^c}, u \in \mathbf{V}, v \in \mathbf{V} \setminus \{c, u\} \tag{11}$$

$$\sum_{w \in \mathbf{V}} z_w \leq K^a, z_w \in \{0, 1\}, \forall w \in \mathbf{V} \tag{12}$$

$$h_i(c) \leq 1 - q_c(u, v), \forall c \in \mathbf{V^c}, u \in \mathbf{V}, v \in \mathbf{V} \cup \{S^a\} \setminus \{u, c\}, \text{ and } \forall \mathbf{D}_i \in \mathbf{D} \text{ such that } \mathbf{P}(u, v) \cap \mathbf{D}_i \neq \emptyset \tag{13}$$

$$h_i(c) \leq z_c, \forall c \in \mathbf{V^c}, \mathbf{D}_i \in \mathbf{D} \tag{14}$$

$$q_c(u, v) \in [0, 1], h_i(c) \in \{0, 1\}, \forall c, u, v, i \tag{15}$$

Theorem 1. *A solution to MILP (6–15) is an optimal pure strategy for the attacker against defender mixed strategy* $\mathbf{x}^*$.

Proof. Given a solution of (6–15), consider each mission-critical node $c \in \mathbf{V^c}$ such that $h_i(c) = 1$ for some i. This means that the attacker successfully exfiltrates data from c given defender pure strategy $\mathbf{D}_i$. There must exist a positive exfiltration path, $\pi^+(c)$, from c to S^a. That is $q_c(u, v) > 0$ for all consecutive bots (u, v) on $\pi^+(c)$. This conclusion results from the attacker strategy constraints in (7–9). Then an optimal pure strategy for the attacker consists of: (i) the set of compromised nodes u with $z_u = 1$; and (ii) the set of positive exfiltration paths $\{\pi^+(c)\}$ for any c which satisfies $h_i(c) = 1$ with some i.

Solving this MILP may take exponential time. In fact, the problem is NP-hard.

Proposition 1. *The attacker oracle problem for data uni-exfiltration is NP-hard.*

The proof is presented in Online Appendix B.[1] We introduce a greedy heuristic to approximately solve the problem.

[1] Link: http://hdl.handle.net/2027.42/137970

Attacker Greedy Heuristic Our greedy heuristic iteratively adds nodes to compromise until the resource limit K^a is reached. At each iteration, given the current set of compromised nodes $\mathbf{B^c}$ (which is initially empty), the greedy heuristic selects among uncompromised nodes $u \in \mathbf{V} \setminus \mathbf{B^c}$ the best next node for the attacker to compromise. A key step of the algorithm is to determine 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 (6–15). In this MILP simplification, the bot variables $\mathbf{z} = \{z_w\}$ are no longer needed. Furthermore, the bot-chain and data-exfiltration variables can be limited to the current set of compromised nodes $\mathbf{B^c} \cup \{u\}$, rather than the whole node set $\mathbf{V}$. As a result, the total number of variables and constraints involved is reduced significantly.

4.2 ORANI Defender Oracle

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

MILP Representation. We parameterize each 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 *data-exfiltration* variables $\mathbf{h} = \{h_j(c)\}$ where $c \in \mathbf{V^c} \cap \mathbf{B}_j$, implying whether the attacker successfully exfiltrates the data of c (i.e., $h_j(c) = 1$) or not ($h_j(c) = 0$). Given that the attacker plays $\mathbf{a}^*$ and the defender plays $\mathbf{z}$, the defender's utility can be now computed based on $\mathbf{h}$ as follows:

$$U^d(\mathbf{z}, \mathbf{a}^*) = -\sum_{\mathbf{A}_j \in \mathbf{A}} a_j^* \sum_{c \in \mathbf{V^c} \cap \mathbf{B}_j} r(c) h_j(c) \tag{16}$$

The problem of finding an optimal pure defense strategy which maximizes the defender's utility against the attacker's strategy $\mathbf{a}^*$ can be now formulated as the following MILP (17–20).

$$\max_{\mathbf{z},\mathbf{I}} U^d(\mathbf{z}, \mathbf{a}^*) \tag{17}$$

$$\text{s.t. } h_j(c) \geq 1 - \sum_{w \in \pi_j(c, S^a)} z_w, \forall c \in \mathbf{V^c} \cap \mathbf{B}_j, \forall j \tag{18}$$

$$\sum_{w \in \mathbf{V}} z_w \leq K^d \tag{19}$$

$$z_w \in \{0,1\}, h_j(c) \in [0,1], \ \forall w \in \mathbf{V}, c \in \mathbf{V^c} \cap \mathbf{B}_j, \forall j \tag{20}$$

In (17–20), only $\mathbf{z} = \{z_w\}$ are required to be binary. Constraint (18) ensures that $h_j(c) = 1$ when the attacker successfully exfiltrates from an stealing bot $c \in \mathbf{V^c} \cap \mathbf{B}_j$

(i.e., the defender does not deploy a detector on the exfiltration path of that bot: $z_w = 0$ for all $w \in \pi_j(c, S^a)$). On the other hand, since the MILP attempts to maximize the defender's utility (Equation 16) which is a monotonically decreasing function of $h_j(c)$, then any MILP solver will automatically force $h_j(c) = 0$ if possible given the bound constraint (20). Constraint (19) guarantees that the number of detection resources deployed does not exceed the limit K^d.

Finally, Proposition 2 shows the complexity of the defender oracle problem. Its proof is in Online Appendix C.

Proposition 2. *The defender oracle problem corresponding to data uni-exfiltration is NP-hard.*

Defender Greedy Heuristic. We introduce a greedy heuristic to approximately solve the defender oracle problem in polynomial time. 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 $u \in \mathbf{V} \setminus \mathbf{D^c}$ to find the best next node to monitor such that the defender's utility is maximized. Computing the defender's utility given a set of monitored nodes and the attacker's strategy $\mathbf{a}^*$ is possible in polynomial time (Equations 1 and 2), thus our defender greedy heuristic runs in polynomial time.

5 Data Broad-Exfiltration

In the botnet defense game model with respect to uni-exfiltration (Section 3), for each stealing bot, the attacker is assumed to only select a single exfiltration path from that bot to exfiltrate data. In this section, we study the botnet defense game model with respect to the alternative data broad-exfiltration. In particular, for every stealing bot, the attacker is able to broadcast the data stolen by this bot to all other compromised nodes via corresponding routing paths. Once receiving the stolen data, each compromised node continues to broadcast the data to all other compromised nodes, and so on. The game model for broad-exfiltration is motivated by the botnet models studied by Rossow et al. [25]. Overall, there is a higher chance that the attacker can successfully exfiltrate network data with broad-exfiltration compared to uni-exfiltration. In the following, we briefly describe the botnet defense game model with data broad-exfiltration and the corresponding algorithm, **ORABI**, to compute an optimal mixed defense strategy.

5.1 Game Model

In the botnet defense game model with data broad-exfiltration, the strategy space of the defender remains the same as shown in Section 3. On the other hand, since the attacker now can broadcast the data, we can abstractly represent each pure strategy of the attacker as a set of compromised nodes $\mathbf{A}_j \equiv \mathbf{B}_j$ only. Given a pair of pure strategies $(\mathbf{D}_i, \mathbf{B}_j)$, we need to determine payoffs the players receive. Note that in the case of broad-exfiltration, given $(\mathbf{D}_i, \mathbf{B}_j)$, the attacker succeeds in exfiltrating the stolen data from a stealing bot if there is an exfiltration path among all the possible exfiltration paths over the compromised set $\mathbf{B}_j$ from this bot to S^a which is not blocked by $\mathbf{D}_i$. Therefore, the players receive a payoff computed as in (1) where the binary indicator $h(c)$ for each mission-critical node $c \in \mathbf{V^c}$ is now determined as:

$$h(c) = \begin{cases} 1 & \text{if } \exists c \in \mathbf{B}_j \ \& \ \exists \pi_j(c, S^a) \\ & \ \text{s.t. } \mathbf{D}_i \cap \pi_j(c, S^a) = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

In fact, when players plays $(\mathbf{D}_i, \mathbf{B}_j)$, we can determine if there is an exfiltration path from a stealing bot $c \in \mathbf{B}_j \cap \mathbf{V^c}$ which is not blocked by $\mathbf{D}_i$ by using depth or breath-first search over the compromised set $\mathbf{B}_j$, which runs in polynomial time. We next aim at computing an SSE of the botnet defense games with data broad-exfiltration. Based upon the double oracle methodology, we introduce a new algorithm, **ORABI**, which consists of new MILPs to exactly solve the resulting attacker and the defender's oracle problems. We also provide greedy heuristics to approximately solve these oracle problems in polynomial time. In the following, we briefly explain our MILPs in **ORABI**.

5.2 ORABI Attacker Oracle

MILP Representation. In solving the attacker oracle problem with respect to data broad-exfiltration, we can extend the MILP (6–15) for data uni-exfiltration as follows. First, each pure strategy of the attacker is now parameterized using only bot variables $\mathbf{z} = \{z_w\}$ for $w \in \mathbf{V}$. Second, although bot-chain variables $\{q_c(u, v)\}$ are not parts of the attacker's pure strategies anymore, we extend these variables $\mathbf{q} = \{q_{i,c}(u, v)\}$ for each pure strategy of the defender $\mathbf{D}_i$. For each mission-critical node $c \in \mathbf{V^c}$ and for each $\mathbf{D}_i \in \mathbf{D}$, $\{q_{i,c}(u, v)\}$ will determine if there is an exfiltration path which successfully exfiltrates stolen data from c given the attacker's pure strategy $\mathbf{z}$. Third, the path-exfiltration constraints in (7–11) and the data-exfiltration constraint in (13) are extended accordingly. Finally, the data-exfiltration and all other constraints and the objective are kept unchanged. Given the extended bot-chain variables $\mathbf{q} = \{q_{i,c}(u, v)\}$ and corresponding extended constraints, the resulting extension of (6–15) will search over all possible exfiltration paths with respect to the attacker's strategy $\mathbf{z}$ to find exfiltration paths which are not blocked by each $\mathbf{D}_i \in \mathbf{D}$. Thus, the extended MILP of (6–15) returns an optimal set of compromised nodes u with $z_u = 1$ for the attacker.

Finally, the attacker oracle problem with broad-exfiltration is NP-hard (Proposition 3 with proof is in the Online Appendix D). The resulting MILP involves a larger number of variables and constraints compared to the uni-exfiltration case due to the extension of bot-chain variables $\mathbf{q} = \{q_{i,c}(u, v)\}$ with respect to the defender's pure strategies $\{\mathbf{D}_i\}$. In the following, we apply the greedy approach for solving the attacker oracle problem in polynomial time.

Proposition 3. *The attacker oracle problem corresponding to data broad-exfiltration is NP-hard.*

Attacker Greedy Heuristic. The attacker greedy heuristic with respect to data broad-exfiltration is similar to the uni-exfiltration case. Nevertheless, given a mixed defense strategy $\mathbf{x}^*$ and a set of compromised nodes $\mathbf{B^c} \cup \{u\}$, we no longer need to find an optimal set of exfiltration paths as in the uni-exfiltration case. As shown in Section 5.1, we can compute the players' utility given $\mathbf{x}^*$ and $\mathbf{B^c} \cup \{u\}$ in polynomial time using depth or breadth-first search.

In addition to this heuristic, we propose a modification of the greedy approach which iteratively adds multiple new pure strategies as follows. Instead of starting the greedy search with an initial empty compromised set $\mathbf{B^c} = \emptyset$, we create $|\mathbf{V^c}|$ different compromised sets $\mathbf{B^c}$, each consists of a mission-critical node $c \in \mathbf{V^c}$ as a compromised seed node. Then for each initial compromised set $\mathbf{B^c}$ with one seed node, we run the greedy search. As a result, we obtain $|\mathbf{V^c}|$ different compromised sets or pure strategies for the attacker. In other words, we add $|\mathbf{V^c}|$ new pure strategies for the attacker at each iteration. We call this modified greedy approach as **greedy-multi** heuristic. Intuitively, by adding multiple new pure strategies, we expect **ORABI** with the greedy-multi heuristic for solving the attacker oracle problem would potentially converge to a solution close to the optimal one. Indeed, our experimental results confirm our conjecture.

5.3 ORABI Defender Oracle

MILP Representation. Although we can also extend the MILP (17–20) for uni-exfiltration to represent the defender oracle problem with broad-exfiltration, solving this extended MILP is impractical. Specifically, in the constraint (18) of the MILP (17–20), we need to iterate over all exfiltration paths to find if the defender's pure strategy $\mathbf{z}$ can block these exfiltration paths or not. Since each pure strategy of the attacker with uni-exfiltration only consists of a small set of exfiltration paths, it is straightforward to iterate over these exfiltration paths. On the other hand, in the broad-exfiltration case, given a pure strategy of the attacker which is now a set of compromised nodes, there is an exponential number of exfiltration paths over these nodes to relay the stolen data. Iterating over all these exfiltration paths is thus impractical.

Given this computational challenge, **ORABI** introduces a new MILP to solve the defender oracle problem. First, we continue to use *detection* variables $\mathbf{z} = \{z_w\}$ to represent a pure strategy of the defender in which $z_w = 1$ if the defender deploys a detector on node w. Otherwise, $z_w = 0$. Second, for each pure strategy of the attacker $\mathbf{B}_j$ and the defender's pure strategy $\mathbf{z}$, we introduce *relaying* variables $\mathbf{l} = \{l_j(u,v)\}$ where $u, v \in \mathbf{B}_j$ are two compromised nodes, indicating whether the attacker can successfully relay data via the routing path $\mathbf{P}(u,v)$. Specifically, the attacker successfully relays data from u to v (i.e., $l_j(u,v) = 1$) if the defender does not deploy a detector on the routing path $\mathbf{P}(u,v)$. Otherwise, $l_j(u,v) = 0$. Third, we introduce variables $\mathbf{s} = \{s^c_j(w)\}$ where $c \in \mathbf{V^c} \cap \mathbf{B}_j$ and $w \in \mathbf{B}_j$. By an abuse of variable name, we also call these new variables as *data-exfiltration* variables. In particular, for each stolen bot $c \in \mathbf{V^c} \cap \mathbf{B}_j$ and $w \in \mathbf{B}_j$, $s^c_j(w)$ indicates if the attacker successfully exfiltrates data of c to the compromised node w ($s^c_j(w) = 1$) or not ($s^c_j(w) = 0$). In other words, $s^c_j(w) = 1$ only when there is an exfiltration path from the stealing bot $c \in \mathbf{V^c} \cap \mathbf{B}_j$ to the compromised node w which is not blocked by the defender. Given $\mathbf{s}$, the defender's utility is computed as follows:

$$U^d(\mathbf{z}, \mathbf{a}^*) = -\sum_{\mathbf{B}_j} a^*_j \sum_{c \in \mathbf{V^c} \cap \mathbf{B}_j} s^c_j(S^a) r(c) \tag{21}$$

where $s^c_j(S^a) = 1$ indicates that the attacker successfully exfiltrates data of $c \in \mathbf{V^c} \cap \mathbf{B}_j$ to S^a. Otherwise, $s^c_j(S^a) = 0$. We now can formulate the defender oracle problem as the following MILP:

$$\max_{\mathbf{z},\mathbf{l},\mathbf{s}} U^d(\mathbf{z}, \mathbf{a}^*) \tag{22}$$

$$\text{s.t. } l_j(u,v) \geq 1 - \sum_{w \in \mathbf{P}(u,v)} z_w, \forall u, v \in \mathbf{B}_j, u \neq v, \forall j \tag{23}$$

$$s^c_j(w) \geq s^c_j(w') + l_j(w', w) - 1, \tag{24}$$

$$\forall c \in \mathbf{V^c} \cap \mathbf{B}_j, w \in \mathbf{B}_j \cup \{S^a\} \setminus \{c\}, w' \in \mathbf{B}_j, w' \neq w, \forall j$$

$$s^c_j(c) \geq 1 - z_c, \forall c \in \mathbf{B}_j \cap \mathbf{V^c}, \forall j \tag{25}$$

$$\sum_{w \in \mathbf{V}} z_w \leq K^d, z_w \in \{0,1\}, \forall w \in \mathbf{V} \tag{26}$$

$$l_j(u,v), s^c_j(w) \in [0,1], \forall c \in \mathbf{V^c}, u, v, w \in \mathbf{B}_j, u \neq v, \forall j \tag{27}$$

which maximizes the defender's utility in Equation 21. Constraint (23) ensures that the attacker can successfully relay data from compromised node u to compromised node v ($l_j(u,v) = 1$) if there is no detector of the defender on the routing path, i.e., $z_w = 0, \forall w \in \mathbf{P}(u,v)$. Constraint (24) guarantees that if the defender does not block the routing path $\mathbf{P}(w', w)$ (i.e., $l_j(w', w) = 1$), node w receives data broadcasted by node w' (i.e., $s^c_j(w) \geq s^c_j(w')$). Furthermore, constraint (25) implies that if the defender does not deploy a detector on a stealing bot $c \in \mathbf{B}_j \cap \mathbf{V^c}$, then the attacker can steal the data of c. In other words, $s^c_j(c) = 1$ if $z_c = 0$ for all $c \in \mathbf{B}_j \cap \mathbf{V^c}$. Finally, constraint (26) imposes the requirement of detection resource limit for the defender.

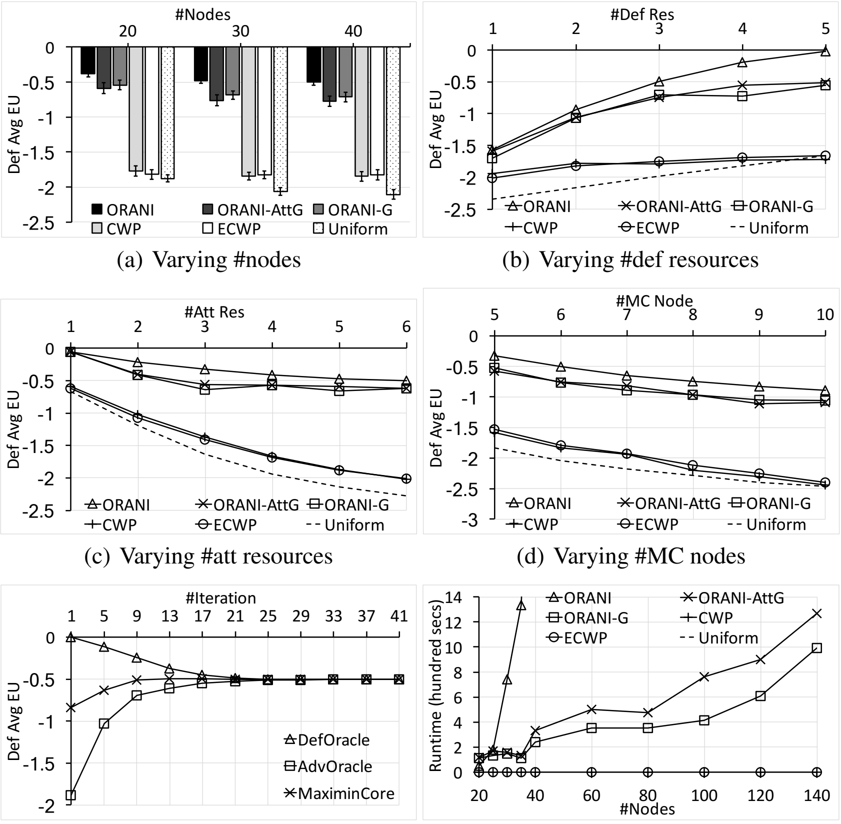
In our MILP (22–27), only the detection variables $\mathbf{z} = \{z_w\}$ are required to be binary. The relaying variables and the data-exfiltration variables will be forced to be equal to one by constraints (23–25) if the attacker can successfully exfiltrate the data. Otherwise, since the defender utility in (21) is monotonically decreasing with respect to the data-exfiltration variables, (22–27) will force these variables to be zero whenever possible. Thus, all the variables are either zero or one in the optimal solution of (22–27). Finally, the defender oracle problem with respect to broad-exfiltration is NP-hard (Proposition 4 with proof is in the Online Appendix E).

Proposition 4. *The defender oracle problem corresponding to data broad-exfiltration is NP-hard.*

Defender Greedy Heuristic. We also apply the greedy approach to solve the defender oracle problem in polynomial time. The idea is similar to the attacker greedy heuristic.

6 Experiments

We evaluate both solution quality and runtime performance of our algorithms compared with previously proposed defense policies. We conduct experiments based on two dif-



(a) Varying #nodes

(b) Varying #def resources

(c) Varying #att resources

(d) Varying #MC nodes

(e) Iteration convergence

(f) Runtime performance

Fig. 2. Uni-Exfiltration: Random scale-free graphs

ferent datasets: (i) synthetic network topology — we use JGraphT [22], a free Java graph library, to randomly generate scale-free graphs since many real-world network topologies exhibit the power-law property [8]; and (ii) real-world network topology — we derive different network topologies from the Rocket-fuel dataset [24]. Each data point in our results is averaged over 50 different samples of network topologies.

6.1 Synthetic Network Topology

Data Uni-Exfiltration We compare six different algorithms: (i) **ORANI** – both exact oracles; (ii) **ORANI-AttG** – exact defender oracle and greedy attacker oracle; (iii) **ORANI-G** – both greedy oracles; (iv&v) CWP & ECWP – heuristics proposed in [37] to generate a mixed defense strategy based on the centrality values of nodes in the network; and (vi) Uniform – generating a uniformly-mixed defense strategy. We consider CWP, ECWP, and Uniform as the three baseline algorithms.

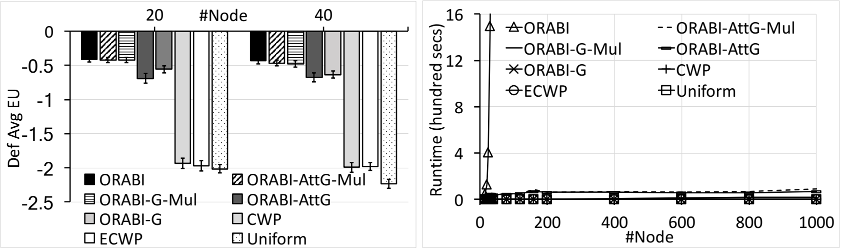
In the first four experiments (Figures 2(a),(b),(c),(d)), we examine solution quality of the algorithms with varying number of nodes, of defender resources, of attacker resources, and of mission-critical nodes respectively. In Figures 2(a),(b),(c),(d), the x-axis is the number of nodes, of defender resources, of attacker resources, and of mission-critical nodes in each graph respectively. In the later three figures, the number of nodes is 30. The y-axis is the averaged expected utility of the defender obtained by the evaluated algorithms. The data value associated with each mission-critical node 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. Figures 2(a), (b), (c), (d) show that all of our algorithms, **ORANI**, **ORANI-AttG**, **ORANI-G** defeat the baseline algorithms in obtaining a much higher utility for the defender. Moreover, when the number of defender resources increases, the defender's expected utility on average increases quickly and reaches the defender's highest utility of zero with just five defender resources. On the other hand, when the number of attacker resources increases, there is only a small decrease in the defender's expected utility on average. Finally, both **ORANI-AttG** and **ORANI-G** obtain a lower average utility of the defender compared to **ORANI** as expected. Yet, we show that the greedy heuristics help in significantly reducing the time of solving the double oracle problem.

In our fifth experiment (Figure 2(e)), 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 with respect to the defender oracle, the attacker oracle, and the Maximin core. The number of nodes in the graph is set to 40. Figure 2(e) shows that **ORANI** converges quickly, i.e., after approximately 25 iterations. This result implies that there is only a small set of pure strategies of players involved in the game equilibrium despite an exponential number of strategies in total. In addition, **ORANI** can find this set of pure strategies after a small number of iterations.

In our sixth experiment (Figure 2(f)), we investigate runtime performance. In Figure 2(f), the x-axis is the number of nodes in the graphs and the y-axis is the runtime on average in hundreds of seconds. As expected, the runtime of **ORANI** grows exponentially when $|\mathbf{V}|$ increases. In addition, by using the greedy heuristics, **ORANI-AttG** and **ORANI-G** run significantly faster than **ORANI**. For example, **ORANI** reaches 1333 seconds on average when $|\mathbf{V}| = 35$ while **ORANI-AttG** and **ORANI-G** reach 1266 and 990 seconds respectively when $|\mathbf{V}| = 140$.

Data Broad-Exfiltration In the case of data broad-exfiltration, we compare eight algorithms: i) **ORABI** – both exact oracles; ii) **ORABI-AttG** – exact defender oracle and greedy attacker oracle; iii) **ORABI-G** – both greedy oracles; iv) **ORABI-AttG-Mul** – exact defender oracle and greedy-multi attacker oracle; v) **ORABI-G-Mul** – both greedy-multi oracles; and vi, vii, viii) CWP, ECWP, and Uniform. Our experiment settings for broad-exfiltration are similar to uni-exfiltration. In the following, we only highlight some key findings.

First, our experimental result on solution quality is shown in Figure 3(a). Figure 3(a) shows that all of our five evaluated algorithms, **ORABI**, **ORABI-AttG-Mul**, **ORABI-G-Mul**, **ORABI-AttG**, and **ORABI-G** obtain a much higher averaged ex-



(a) Solution quality

(b) Runtime performance

Fig. 3. Broad-Exfiltration: Random scale-free graphs

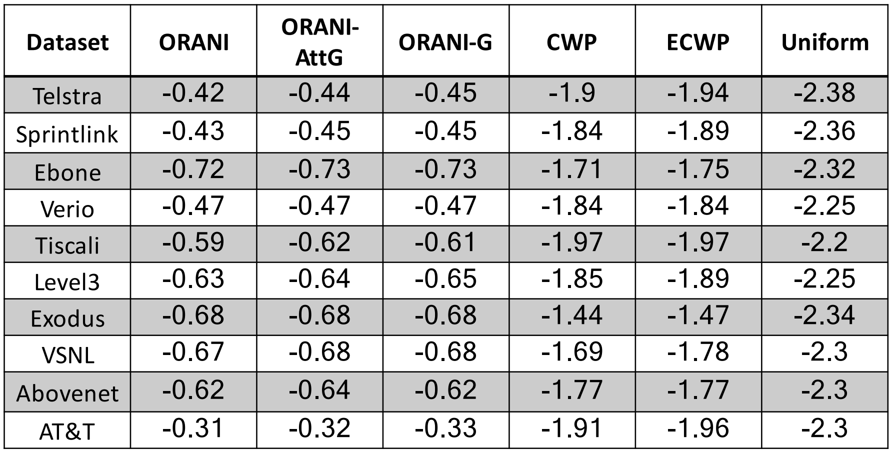
pected utility for the defender compared to the baseline algorithms. Furthermore, by adding multple new strategies at each iteration, both our algorithms **ORABI-AttG-Mul** and **ORABI-G-Mul** perform approximately as well as **ORABI** while outperforming **ORABI-AttG**, and **ORABI-G**.

Furthermore, in the experimental result on runtime performance (Figure 3(b)), our algorithms with greedy heuristics can scale up to large graphs. For example, when $|\mathbf{V}| = 1000$, the runtime of **ORABI-AttG-Mul**, **ORABI-G-Mul**, **ORABI-AttG**, and **ORABI-G** reaches 89, 20, 71, and 2 seconds respectively. We conclude that **ORABI** is the best algorithm for small graphs while **ORABI-AttG-Mul** and **ORABI-G-Mul** are proper choices for large-scale graphs.

Finally, we investigate the benefit to the attacker from broad-exfiltration compared to uni-exfiltration. We run **ORANI** and **ORABI** on the same set of 50 scale-free graph samples generated by uniformly at random with 20, 30, 40 nodes in each graph respectively. Among all the samples, there are only 58%, 72%, and 52% of the 20-node, 30-node, and 40-node graphs respectively for which the attacker obtains a strictly higher utility by using broad-exfiltration. This result shows that the attacker does not always benefit from broad-exfiltration. Indeed, despite broad-exfiltration, the data exchange between any pairs of compromised nodes must follow fixed routing paths specified by the network system, thus constraining the data exfiltration possibilities.

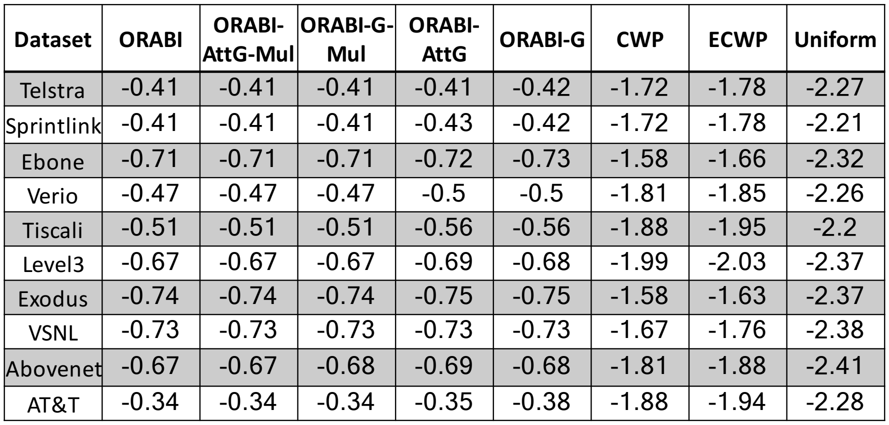
6.2 Real-world Network Topology

Our third set of experiments is conducted on real-world network topologies from the Rocket-fuel dataset [24]. 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 fifty 40-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 data 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 Figures 4&5 with respect to data uni-exfiltration and broad-exfiltration respectively.



Dataset	ORANI	ORANI-AttG	ORANI-G	CWP	ECWP	Uniform
Telstra	-0.42	-0.44	-0.45	-1.9	-1.94	-2.38
Sprintlink	-0.43	-0.45	-0.45	-1.84	-1.89	-2.36
Ebone	-0.72	-0.73	-0.73	-1.71	-1.75	-2.32
Verio	-0.47	-0.47	-0.47	-1.84	-1.84	-2.25
Tiscali	-0.59	-0.62	-0.61	-1.97	-1.97	-2.2
Level3	-0.63	-0.64	-0.65	-1.85	-1.89	-2.25
Exodus	-0.68	-0.68	-0.68	-1.44	-1.47	-2.34
VSNL	-0.67	-0.68	-0.68	-1.69	-1.78	-2.3
Abovenet	-0.62	-0.64	-0.62	-1.77	-1.77	-2.3
AT&T	-0.31	-0.32	-0.33	-1.91	-1.96	-2.3

Fig. 4. Uni-Exfiltration: Defender's average utility



Dataset	ORABI	ORABI-AttG-Mul	ORABI-G-Mul	ORABI-AttG	ORABI-G	CWP	ECWP	Uniform
Telstra	-0.41	-0.41	-0.41	-0.41	-0.42	-1.72	-1.78	-2.27
Sprintlink	-0.41	-0.41	-0.41	-0.43	-0.42	-1.72	-1.78	-2.21
Ebone	-0.71	-0.71	-0.71	-0.72	-0.73	-1.58	-1.66	-2.32
Verio	-0.47	-0.47	-0.47	-0.5	-0.5	-1.81	-1.85	-2.26
Tiscali	-0.51	-0.51	-0.51	-0.56	-0.56	-1.88	-1.95	-2.2
Level3	-0.67	-0.67	-0.67	-0.69	-0.68	-1.99	-2.03	-2.37
Exodus	-0.74	-0.74	-0.74	-0.75	-0.75	-1.58	-1.63	-2.37
VSNL	-0.73	-0.73	-0.73	-0.73	-0.73	-1.67	-1.76	-2.38
Abovenet	-0.67	-0.67	-0.68	-0.69	-0.68	-1.81	-1.88	-2.41
AT&T	-0.34	-0.34	-0.34	-0.35	-0.38	-1.88	-1.94	-2.28

Fig. 5. Broad-Exfiltration: Defender's average utility

Figures 4 and 5 show that all of our algorithms obtain higher defender expected utility than the three baseline algorithms. Further, the greedy algorithms—**ORANI-AttG**, **ORANI-G**, and **ORABI-AttG**, **ORABI-G**—are shown to consistently perform well on all the ISP network topologies compared to the optimal ones—**ORANI** and **ORABI** respectively. In particular, the average expected defender utility obtained by **ORANI-G** is only $\approx 3\%$ lower than **ORANI** on average over the 10 network topologies.

7 Summary

Many computer networks have suffered from botnet data exfiltration attacks, leading to a significant research emphasis on botnet defense. Our Stackelberg game model for the botnet defense problem accounts for the strategic response of cyber-criminals to deployed defenses. We propose two double-oracle based algorithms, **ORANI** and **ORABI**, to compute optimal defense strategies with respect to data uni-exfiltration and broad-exfiltration formulations, respectively. We also provide greedy heuristics to approximate the defender and the attacker best-response oracles. We conduct experiments based on both random scale-free graphs and 10 real-world ISP network topologies, demonstrating advantages of our game-theoretic solution compared to previous strategies.

Acknowledgment

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

Bibliography

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

[2] A. Baldwin, I. Gheyas, C. Ioannidis, D. Pym, and J. Williams. Contagion in cyber security attacks. *Journal of the Operational Research Society*, 68:780–791, 2017.

[3] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *8th International Conference on Autonomous Agents and Multiagent Systems*, pages 57–64, 2009.

[4] A. Bensoussan, M. Kantarcioglu, and S. C. Hoe. A game-theoretical approach for finding optimal strategies in a botnet defense model. In *1st Conference on Decision and Game Theory for Security*, pages 135–148, 2010.

[5] H. Choi, H. Lee, H. Lee, and H. Kim. Botnet detection by monitoring group activities in DNS traffic. In *7th IEEE International Conference on Computer and Information Technology*, pages 715–720. IEEE, 2007.

[6] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, pages 39–44, 2005.

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

[8] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *ACM SIGCOMM Computer Communication Review*, 29(4):251–262, 1999.

[9] F. Fang, T. H. Nguyen, R. Pickles, W. Y. Lam, G. R. Clements, B. An, A. Singh, M. Tambe, and A. Lemieux. Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *28th Conference on Innovative Applications of Artificial Intelligence*, pages 3966–3973, 2016.

[10] M. Feily, A. Shahrestani, and S. Ramadass. A survey of botnet and botnet detection. In *3rd International Conference on Emerging Security Information, Systems, and Technologies*, pages 268–273, 2009.

[11] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al. BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *17th USENIX Security Symposium*, pages 139–154, 2008.

[12] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In *16th USENIX Security Symposium*, pages 167–182, 2007.

[13] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *15th Annual Network and Distributed System Security Symposium*, 2008.

[14] T. Holz, M. Engelberth, and F. Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *European Symposium on Research in Computer Security*, pages 1–18, 2009.

[15] M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *10th Inter-*

national Conference on Autonomous Agents and MultiAgent Systems, pages 327–334, 2011.

[16] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordó/ nez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *8th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 689–696, 2009.

[17] V. Kolokoltsov and A. Bensoussan. Mean-field-game model for botnet defense in cyber-security. *Applied Mathematics and Optimization*, 74:669–692, 2016.

[18] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

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

[20] S. M. Mc Carthy, A. Sinha, M. Tambe, and P. Manadhata. Data exfiltration detection and prevention: Virtually distributed POMDPs for practically safer networks. In *7th Conference on Decision and Game Theory for Security*, pages 69–61, 2016.

[21] H. B. McMahan, G. J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *20th International Conference on Machine Learning*, pages 536–543, 2003.

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

[23] T. Peng, C. Leckie, and K. Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys*, 39(1):3, 2007.

[24] Rocketfuel. Rocketfuel: An ISP topology mapping engine, 2002.

[25] C. Rossow, D. Andriesse, T. Werner, B. Stone-Gross, D. Plohmann, C. J. Dietrich, and H. Bos. SoK: P2PWNED — Modeling and evaluating the resilience of peer-to-peer botnets. In *IEEE Symposium on Security and Privacy*, pages 97–111, 2013.

[26] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. PROTECT: A deployed game theoretic system to protect the ports of the United States. In *11th International Conference on Autonomous Agents and Multiagent Systems*, pages 13–20, 2012.

[27] B. Soper and J. Musacchio. A botnet detection game. In *52nd Annual Allerton Conference on Communication Control and Computing*, pages 294–303. IEEE, 2014.

[28] B. C. Soper. *Non-Zero-Sum, Adversarial Detection Games in Network Security*. PhD thesis, University of California, Santa Cruz, 2015.

[29] E. Stinson and J. C. Mitchell. Towards systematic evaluation of the evadability of bot/botnet detection methods. In *2d USENIX Workshop on Offensive Technologies*, 2008.

[30] B. Stone-Gross, R. Abman, R. A. Kemmerer, C. Kruegel, D. G. Steigerwald, and G. Vigna. The underground economy of fake antivirus software. In *10th Workshop on the Economics of Information Security*, 2011.

[31] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *16th ACM Conference on Computer and Communications Security*, pages 635–647, 2009.

[32] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas. Botnet detection based on network behavior. In W. Lee, C. Wang, and D. Dagon, editors, *Botnet Detection: Countering the Largest Security Threat*, pages 1–24. Springer, 2008.

[33] P. J. Sweeney. *Designing Effective and Stealthy Botnets for Cyber Espionage and Interdiction: Finding the Cyber High Ground*. PhD thesis, Sept. 2014.

[34] M. Tambe, editor. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[35] M. Van Eeten, J. M. Bauer, H. Asghari, S. Tabatabaie, and D. Rand. The role of Internet service providers in botnet mitigation an empirical analysis based on spam data. In *9th Workshop on the Economics of Information Security*, 2010.

[36] O. Vaněk, Z. Yin, M. Jain, B. Bošanskỳ, M. Tambe, and M. Pěchouček. Game-theoretic resource allocation for malicious packet detection in computer networks. In *11th International Conference on Autonomous Agents and Multiagent Systems*, pages 905–912, 2012.

[37] S. Venkatesan, M. Albanese, G. Cybenko, and S. Jajodia. A moving target defense approach to disrupting stealthy botnets. In *ACM Workshop on Moving Target Defense*, pages 37–46, 2016.

[38] S. Venkatesan, M. Albanese, and S. Jajodia. Disrupting stealthy botnets through strategic placement of detectors. In *IEEE Conference on Communications and Network Security (CNS)*, pages 95–103, 2015.

[39] P. Wang, S. Sparks, and C. C. Zou. An advanced hybrid peer-to-peer botnet. *IEEE Transactions on Dependable and Secure Computing*, 7(2):113, 2010.