Game-theoretic Resource Allocation for Malicious Packet Detection in Computer Networks

Ondřej Vaněk[†],Zhengyu Yin[*], Manish Jain[*], Branislav Bošanský[†],
Milind Tambe[*],Michal Pěchouček[†]
[†] Faculty of Electrical Engineering, Czech Technical University, Prague. Czech Republic.
{vanek,bosansky,pechoucek}@agents.fel.cvut.cz
[*] Computer Science Department, University of Southern California, Los Angeles, CA. USA.
{zhengyu.yin,manish.jain,tambe}@usc.edu

ABSTRACT

We study the problem of optimal resource allocation for packet selection and inspection to detect potential threats in large computer networks with multiple valuable computers of differing importance. An attacker tries to harm these targets by sending malicious packets from multiple entry points of the network; the defender thus needs to optimally allocate his resources to maximize the probability of malicious packet detection under network latency constraints.

We formulate the problem as a graph-based security game with multiple resources of heterogeneous capabilities and propose a mathematical program for finding optimal solutions. Due to the very limited scalability caused by the large attacker's strategy space and non-linearity of the program, we investigate solutions with approximated utility function and propose GRANDE, a novel polynomial approximate algorithm utilizing submodularity of the problem able to find solutions with a bounded error on problem of a realistic size.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent Systems; C.2.0 [**Computer-Communication Networks**]: Security and Protection

General Terms

Algorithms, Security, Performance

Keywords

computer networks, security, game-theory, approximation algorithm, submodularity

1. INTRODUCTION

The problem of attacks on computer systems and corporate computer networks gets more pressing each year as the sophistication of the attacks increases together with the cost of their prevention. A number of intrusion detection and monitoring systems is being developed in order to increase the security of sensitive information, and many research works seek methods for optimizing the use of available security resources. *Deep packet inspection* method that periodically selects a subset of packets in a computer network for analysis, is one of such countermeasures frequently deployed in computer networks. However, there is a cost associated with the deep packet inspection, as it leads to significant delays in the throughput of the network. Thus, the monitoring system works under a constraint of limited selection of a fraction of all packets which can be inspected.

Game-theoretic methods are appropriate for modeling such problems and the optimal behavior of the involved parties can be found using a well-defined concept of an equilibrium computation. We formulate this problem as a game between two players: the *attacker* (or the *intruder*), and the *defender* (the *detection system*). The intruder wants to gain control over (or to disable) a valuable computer in the network by scanning the network, hacking into a more vulnerable system, and/or gaining access to further devices on the computer network. The actions of the attacker can therefore be seen as sending malicious packets from a controlled computer (termed *source*) to a single or multiple vulnerable computers (termed *targets*). The objective of the defender is to prevent the intruder from succeeding by selecting the packets for inspection, identifying the attacker, and subsequently thwarting the attack. However, packet inspections cause unwanted latency and hence the defender has to decide where and how to inspect network traffic in order to maximize the probability of a successful malicious packet detection.

We build on previous work on game-theoretic approaches to network security [1, 10] and security games [8, 19] to present a novel approach to the challenges of malicious packet detection. In our approach, we follow the deep packet inspection scenario on an arbitrary network topology, and consider following assumptions, that hold in this domain: the attacker can access the computer network through multiple entry points, can attack multiple targets of differing importance in parallel, and the defender has limited resources that can be used for packet analysis. To the best of our knowledge, there is no previous work considering together all of these aspects of the problem.

This paper offers three main contributions: (1) we propose a novel game-theoretic model that can be characterized as a graph-based security game with multiple heterogeneous attacker's and defender's resources; (2) we give a mathematical program for finding the optimal solutions for this

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

problem formulated both as a non-zero sum and zero sum game; and (3) we describe a polynomial approximation algorithm GRANDE (*GReedy Algorithm for Network DEfense*) that scales well with the increasing size of the network at the cost of a known bounded error.

The paper is organized as follows: After reviewing related work we formalize the problem of a packet selection for deep inspection as a two-player general-sum game and define utility functions for both players. Next, we provide a mathematical program for finding optimal solutions for this problem. As program is non-linear, we follow with two simplifications of the model. We reformulate the game as a zero-sum game, analyze the property of submodularity of the problem, and formulate a discretized submodular zero-sum variant of the game. Next, we present the approximate algorithm, GRANDE, that benefits from the submodularity property of the discretized zero-sum variant of the game and finds solutions with bounded error in polynomial time. We experimentally evaluate these three algorithms and show the trade-off in computational time and quality of found solutions for our problem. Using GRANDE, we are able to compute solutions for problems on networks with up to thousands of nodes, hundreds of sources and tens of targets in tens of hours.

2. RELATED WORK

Game theory has been applied to a wide range of security problems, with many deployed applications in transportation networks [9, 19]. In fact, game-theoretic models of real-world security problems are applicable in a wide variety of domains with similar attributes, including (1) intelligent players, (2) varying preferences among targets, (3) and limited resources to completely protect all targets. This has led researchers to model computer network security in game-theoretic frameworks and a large body of work has been created (summarized e.g. in [14]).

Most related is the recent work by Kodialam and Lakshman [10] since they also look at a scenario where the defender conducts inspections on possible paths from a source to a target. However, they look at a zero-sum setting for a single source and a single target. Similarly, Otrok et al. [16] present solutions for a domain with a single target, where the attacker potentially uses multiple packets for an attack. Furthermore, Chen et al. [2] present solutions for heterogeneous targets, with multiple attacker resources. However, they only consider detection at the target nodes.

From the research focused on the security-games models, Korzhyk et al. [11] present a polynomial algorithm for general-sum security games with multiple attacker resources, however, without constraining underlying graph structure. Jain et al. [8] present an algorithm for securing an urban network with many sources and heterogeneous targets. However, this model is zero-sum and the attacker has a single resource. Our approach mainly differs in considering a network-security domain, where the payoffs are not necessarily zero-sum and player's utilities have more complex structure. We also model the attacker with multiple resources used in parallel, so the defender succeeds in preventing an attack, only if all attacker's paths leading to a single target are intercepted.

Our work also exploits the submodular properties of the network security domain. Submodular functions for optimal resource allocation optimization in adversarial environments were first introduced by Freud et al. [6], and further developed by Krause et. al [12]. However, they do not work with continuous defender resources and consider only zero-sum setting.

3. FORMAL MODEL

3.1 Environment

We define the problem of the packet selection for inspection as a two-player game between *the attacker* and *the defender*. The game is played on a graph $G(N, E)$ that represents the topology of a computer network. The set of nodes can be decomposed into three non-empty sets: (1) S is the set of nodes that can be under the control of the attacker; (2) T is the set of targets ($S \cap T = \emptyset$); (3) $A = N \backslash \{S \cup T\}$ is the set of all other nodes in the network. From A, the defender can inspect the traffic only on a subset of *intermediate* nodes $I \subseteq A$ (representing, for example, firewalls, proxy servers, etc.). For our problem, we consider only nodes from S, T, I.

The packages are routed in the network by an underlying *deterministic* routing protocol that is not under the attacker's control; therefore, for each tuple $(s, t) : s \subseteq S, t \subseteq T$, there is either a fixed *single* path through intermediate nodes I, or there is a path without intermediate nodes leading from s to t, or there is no path from s to t in the graph. Thus, the defender does not need to consider allocation of resources to such intermediate nodes which do not lie on any path (given set of sources and targets).

Each target t has an associated value $\tau_t \geq 0$ that represents the importance of the target; the defender loses τ_t if t is attacked successfully and gains 0 if she succeeds in preventing the attack. The attacker gains τ_t if t is attacked successfully. In case of an unsuccessful attack (i.e. a malicious packet was detected by the defender), the attacker pays a detection penalty $\gamma_s \geq 0$ associated with using the source s[1]. The penalty for the attacker models situations when the defender detects that source s is being used to send malicious packets and blocks this source from the network. Due to this penalty, the attacker may choose not to attack any of the targets, we thus define a virtual node in the network — a *dummy target* t_D that is directly connected to all sources, and for which $\tau_{t_D} = 0$. Finally, for each intermediate node n_i we define *flow* f_i that represents the amount of legitimate network traffic going through n_i. We assume this amount to be constant in time[2] and we assume that the amount of malicious packets sent by the attacker is fractional compared to the legitimate network traffic[3].

3.2 Players

Both players have multiple resources that they can use. The resources of the attacker are determined by the size of the set of sources $|S| = k$ and we assume that the attacker attacks from one source only a single target (however, one target can be attacked from multiple sources). The strategy of the attacker is to select k tuples determining which target

[1]We assume standard IP packets with a source IP address, i.e. the source is traceable from the packet header; for *spoofing* attacks, γ_s is set to zero.

[2]If the amount of traffic varies in time (e.g., weekends vs. workdays, days vs. nights), we can compute multiple strategies and switch between them.

[3]For flooding attacks, other detection/prevention countermeasures than deep packet inspection are used.

will be attacked from each of the sources:

$$P = \{(s_i, t_i) : s_i \in S, t_i \in T, i = 1 \dots k, \forall_{j \neq i} s_i \neq s_j\}$$

Since there is at most one path from source s to target t in the graph, we refer to attacker's strategies as paths and denote them as $p_{(s,t)} \subseteq I$ with subscript omitted if the source-target is clear from the context. Hence, we denote $\mathcal{C}_t$ to be a set of such paths that originate in some source and end in a single target t. Due to the structure of the graph, the attacker's resources can be seen as heterogeneous, because not each target can be attacked from each source.

The strategy of the defender is an assignment of a vector of probabilities $X = (x_1, x_2, \dots, x_m)$, where for each intermediate node $n \in I$; $(m = |I|)$, the probability x_i represents the fraction of the traffic going through the node n_i that will be inspected. Therefore, the value x_i also represents the probability of the detection of a malicious packet sent by the attacker through node n_i[4]. The available amount of defender's resources is determined by the maximum amount of inspected traffic B that is limited by the maximum allowed average latency in the computer network. Therefore, the defender is seeking her strategy under the following constraint:

$$L(X) = \sum_{n_i \in I} x_i \cdot f_i \leq B$$

where $x_i \cdot f_i$ represents the expected number of packets that were inspected at node n_i (for the complete network, we use $L(X)$ for brevity). As in the case of the attacker, the heterogeneity of the defender's resources is given by the structure of the graph – different intermediate nodes provide a malicious packet detection for different groups of targets.

Finally, when designing an intrusion detection system, a typical assumption is that the attacker will have a full knowledge of techniques used by the system [18] and together with the full knowledge of the network structure[5] the attacker is able to reconstruct the defender's strategy; the attacker is thus assumed to know the probability with which a packet may be inspected at each of the intermediate nodes. In this paper, we thus assume Stackelberg game formulation; however, the relaxation of these assumptions is subject of further research.

3.3 Utility Functions

The utility functions of both players primarily depend on the probability of detection of sent malicious packets. Since the intermediate nodes inspect packets independently, the probability of a single malicious packet **avoiding detection** along the path p is given by:

$$\pi(X, p) = \prod_{i \in p} (1 - x_i) \tag{1}$$

where X is a strategy of the defender in the form of allocation of detection resources at nodes n_i. The probability of **detecting** a packet on each path for a set of paths $\mathcal{C}$ is computed as:

$$\psi(X, \mathcal{C}) = \prod_{p \in \mathcal{C}} [1 - \pi(X, p)]$$

Now, if P denotes the strategy of the attacker (i.e., paths p_j in the graph), and $\mathcal{C}_t$ is the set of all paths chosen by the attacker leading to a target t, the utility of the defender $U_d(X, P)$ is defined as follows:

$$U_d(X, P) = -\sum_{\forall t \in T} \tau_t \cdot [1 - \psi(X, \mathcal{C}_t)] \tag{2}$$

where the term $(1 - \psi(X, \mathcal{C}_t))$ denotes the probability that at least one malicious packet avoids detection and reaches target t. Therefore, the defender's utility is an expected loss of values of targets that were reached by malicious packets. Analogously, we define the attacker's utility $U_a(X, P)$ as:

$$U_a(X, P) = -U_d(X, P) - \sum_{p_{(s,t)} \in P} \gamma_s \cdot [1 - \pi(X, p)] \tag{3}$$

The attacker's utility equals to the expected gain of values of targets that were reached by malicious packets reduced by the detection penalty γ_s for each path that the attacker uses;recall the attacker needs to pay a penalty when a packet is detected, as discussed in Section 3.1. As such, the game is not necessarily zero-sum.

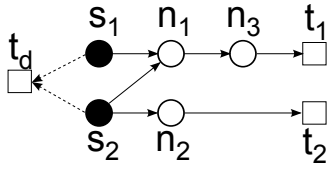


Figure 1: Example graph. Two source nodes s_1 and s_2, three intermediate nodes n_1, n_2 and n_3, two target nodes t_1 and t_2, and a dummy target node t_d.

3.4 Example

The example on Figure 1 depicts a simple graph with two sources s_1, s_2, three intermediate nodes n_1, n_2, n_3 with flows $f_1 = 5, f_2 = 3, f_3 = 5$ and two targets t_1, t_2 with values $\tau_{t_1} = 2, \tau_{t_2} = 6$. The number of adversary resources $k = 2$ and defender's latency budget is set to $B = 6$. The attacker's strategy set is:

$$\begin{aligned}\mathbb{P} = \{ \quad & [(s_1, t_1), (s_2, t_1)], [(s_1, t_1), (s_2, t_2)], \\ & [(s_1, t_D), (s_2, t_1)], [(s_1, t_D), (s_2, t_2)], \\ & [(s_1, t_1), (s_2, t_D)], [(s_1, t_D), (s_2, t_D)]\}.\end{aligned}$$

If, for example, the defender chooses it's strategy to be $X = \{x_1 = 0.0, x_2 = 0.5, x_3 = 0.1\}$, the latency caused is $L(X) = 0.0 \cdot 5 + 0.5 \cdot 3 + 0.1 \cdot 5 = 2$. If the attacker selects a strategy $P = [(s_1, t_1), (s_2, t_1)]$, the defender's utility will be: $U_d(X, P) = -2 \cdot [1 - (1 - (1 - 0.0) \cdot (1 - 0.1)) \cdot (1 - (1 - 0.0) \cdot (1 - 0.1))] = -1.98$. The attacker's utility will be (when setting $\gamma_{s_1} = \gamma_{s_2} = 1$): $U_a(X, P) = -U_d(X, P) - 1 \cdot (1 - (1 - 0.0) \cdot (1 - 0.1)) + (1 - (1 - 0.0) \cdot (1 - 0.1)) = 1.98 - 0.2 = 1.78$. The optimal setting is $X^* = \{x_1 = 0.0, x_2 = 0.857, x_3 = 0.686\}$, forcing the attacker to select $P^* = [(s_1, t_D), (s_2, t_2)]$, giving the defender expected utility $U_d(X^*, P^*) = -0.858$; and the attacker's expected utility is $U_a(X^*, P^*) = 0.001$.

4. SOLUTION APPROACH

First, we look for Strong Stackelberg Equilibrium (SSE) of the full general-sum game. Because the computational limits are reached even for small problems, we also propose a zero-sum game model (Section 4.2). Exploring and then

[4]This assumes having a perfect detector.

[5]Using standard network analysis tools, such as Nmap.

utilizing the submodularity of the problem (Section 4.3), we propose an iterative algorithm GRANDE for finding suboptimal solutions in polynomial time (Section 4.4).

4.1 General-sum Game Model

Given the assumptions stated above, we model the problem as a Stackelberg general-sum game between the defender and the attacker: the defender is the leader, committing to her strategy first, and the attacker is a follower, choosing strategy after the leader's commitment. The SSE gives the optimal strategy for the leader given that the follower acts with the knowledge of this optimal leader strategy. It is found by solving multiple programs [3] as follows:

$$\max_X \quad U_d(X, P^*) \tag{4}$$

$$\text{s.t.} \quad L(X) \leq B \tag{5}$$

$$U_a(X, P^*) \geq U_a(X, P) \qquad \forall P \tag{6}$$

$$x_i \in [0, 1] \tag{7}$$

The inputs of the programs are all possible pure strategies of the attacker P and P^* is assumed to be the current best response for the attacker. We compute the defender strategy X that maximizes the defender's utility $U_d(X, P^*)$ (Equation 4) while adhering to the latency constraint (Equation 5) and ensuring that the assumed best response of the attacker is consistent with the attacker's rational attitude (Equation 6). Note that the program may not always be feasible if some choice of P^* is strictly dominated by others, but it will always return a solution for all non-dominated P^*. The number of programs needed to be solved to find an optimal solution is given by the number of attacker's strategies, which is $|T|^k$, since there are $|T|$ targets and k sources. This approach has two main scalability limitations: first, the non-linear formulations of U_d and U_a prohibit us from using fast linear-program solvers; second, the attacker's strategy space is extremely large (for a graph with 5 sources, 5 targets and one *dummy* target, we get over 7500 (6^5) programs with similar number of non-linear equations), limiting the usability of the non-linear solvers.

An alternative approach, inspired by algorithms computing SSE by solving a single mixed-integer program [17], would introduce into each Equation 6 an integer variable z_i (for each attacker's strategy P_i) and restrict the variables by $\sum z_i = 1$, i.e., only one attacker's strategy can be selected as the best response. However, this program would be very large, having $(|T|^k)^2$ non-linear equations (which is over 56 million for the problem with 5 sources and 5 targets). Hence, we look at the zero-sum game formulation for the problem which allows us to exploit the structure in ways that keep the solution tractable.

4.2 Zero-sum Game Approximation

Finding an optimal solution using the full general-sum game representation is computationally demanding on large problems. We thus propose a zero-sum game formulation which reduces the complexity of the model. Setting the cost of each source to $\gamma_s = 0$, the utility function of the attacker becomes a negation of the utility of the defender ($U_d(X, P) = -U_a(X, P)$), and the game becomes zero-sum. In zero-sum games, the SSE equals to Nash Equilibrium, which can be computed using the minimax theorem. This approximation causes an error quantified in Section 5. SSE of our zero-sum game can be found by solving a single non-linear mathematical program:

$$\max_X \quad \mathcal{V} \tag{8}$$

$$\text{s.t.} \quad U_d(X, P) \geq \mathcal{V} \qquad \forall P \tag{9}$$

$$L(X) \leq B \tag{10}$$

$$x_i \in [0, 1] \tag{11}$$

In this mathematical program, the main scalability limitation persists – as for the general sum model – the non-linear nature of the utility function (Equation 9) and the size of the linear program, depending on the size of the attacker's strategy space (Equation 9). However, in spite of the negative findings, zero-sum games are generally easier to solve optimally (e.g., iterative algorithms can be used as in [7, 8]) or to approximate [13]. We follow the latter approach and investigate approximation algorithms that utilize the property of submodularity and are able to find solutions for zero-sum games with guaranteed bounded error.

4.3 Submodularity

In our problem formulation, there exists the effect of diminishing returns, i.e., as the number of defender's resources is increased, the marginal utility of deploying one extra resource keeps decreasing. This property is formalized by the concept of submodularity [15] which is utilized in many domains (e.g., sensor networks) to design effective algorithms able to solve large problems. A real-valued function F defined on subsets A of a finite set V is called submodular, if for all $A \subset B \subset V$ and for all $s \in V \backslash B$ holds, that $F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$. The constrained optimization of a submodular set function is NP-hard in general, however, a number of approximation algorithms with provable quality guarantees can be used [21].

In our formulation, we have intermediate nodes that detect the activity of the attacker. The value of the detected activity in this problem setting is a probability between $[0, 1]$, as opposed to being binary which is generally assumed in submodularity. Thus, our requirements do not meet the assumptions of most prior work on submodularity, except the work by Vondrak et al. [21], which studied smooth continuous extension of submodular functions by taking expectations, defining sensors making observations independently with probability in range $[0, 1]$. The approach requires the continuous function to be twice partially differentiable and an approximation bound is established by exploiting the up-concavity[6] of the resulting continuous function [5]. Unfortunately, this work is not applicable to our problem since our objective function, $U_d(X, P)$, is not up-concave which can be determined by taking the double derivative of the defender utility function.

4.4 GRANDE Algorithm

We choose a different approach (in contrast to standard submodular approaches) for purchasing an algorithm utilizing the submodularity of the problem: we transform U_d into a submodular function defined over sets by discretizing the sampling rate of each node and we allow nodes to sample only a fixed portions of traffic defined by a discretization step $d \leq 1$; e.g. for $d = 0.1$, the sampling rate at each node can

[6]Up-concavity means that the function is concave along any non-negative direction vector; however, it is not necessarily concave in all directions.

be set only to $0, 10, 20, \ldots, 100\%$. Then, each node n_i can be seen as a set of $1/d$ *sensors* $\mathcal{S}(n_i) = \{n_i^1, n_i^2, \ldots, n_i^{1/d}\}$. A sensor n_i^j can be switched either on (and sample a portion of the traffic) or off which is expressed by a binary variable $x_i^j \in \{0, 1\}$ having value of 1 for a sensor switched on. The defender's strategy is defined using the sensor notation as $X = \{x_i^j\}$. We redefine the Equation 1 defining the probability of a single malicious packet avoiding detection along a path p as:

$$\pi(X, p) = \prod_{n_i \in p} (1 - d \cdot \sum_{\mathcal{S}(n_i)} x_i^j) \tag{12}$$

Having a submodular utility function defined over sets for the defender U_d (which has the same formulation as in Equation 2), we are able to design an iterative greedy algorithm to achieve at least $(1 - 1/e)$-optimal (approximately 63.2%) solution (compared to the zero-sum game SSE) [6] similarly to work of Krause et al. [13]. However, it is also necessary to consider the cost of each sensor, given by the budget constraint $L(X) \leq B$. When inspecting the same ratio of packets at two nodes n_i, n_k with flows $f(n_i) > f(n_k)$, the cost of inspection at the node n_i (and thus switching on a sensor at node n_i) is higher than inspection cost at node n_k. The cost of switching on a sensor is defined as $c(n_i^j) = d \cdot f(n_i)$. As shown in [12], the greedy algorithm has to select a sensor with the highest benefit-cost ratio $x_i^j = \arg\max_{x_i^j} \frac{U_d(X \cup \{x_i^j\}, P) - U_d(X, P)}{c(n_i^j)}$.

Based on this formalization, we introduce GRANDE (*GReedy Algorithm for Network DEfense*), depicted in Algorithm 1. GRANDE iteratively selects sensors with the highest security increment vs. cost ratio to add to the defender strategy, until the whole sampling budget is spent. To find the best sensor to add, we find attacker's optimal strategy $attackerBR$ and test each candidate sensor against this strategy. The complexity of the algorithm is thus given by the number of nodes $n = |I|$, by the discretization step d, by the minimum amount of traffic flow at each node f, by the sampling budget B and by the complexity of the attacker's best response oracle $O(\mathcal{BR})$. The algorithm asymptotic complexity is thus $O(nB/fd) \cdot O(\mathcal{BR})$.

The attacker's optimal strategy $attackerBR$ is a best response to the current defender's strategy (following the original Stackelberg formulation). The algorithm thus needs a fast best response oracle providing best response to the current strategy of the other player. The following section defines such oracle and provides insight into the complexity of this approach.

4.4.1 Attacker's Best Response Oracle

Recall, that the attacker only selects for each source a target to attack and the routing path is automatically assigned. The attacker's best response is thus an optimal assignment of a target to every source, given a fixed defender's strategy X, maximizing the attacker's utility. The attacker's best response can be found using an iterative greedy approach.

Let's assume we have defender's strategy — a mixture of sampling probabilities x_i for each node n_i. We can compute for each source-target pair, what is the likelihood of being detected $\rho_s^t = 1 - \pi(X, p_{(s,t)})$. Let's denote the source-target pairs by $STP = \{(s_1, t_1), \ldots, (s_n, t_m)\}$. We also know that two different source-target pairs (with different sources) can share the same target t. Given input $\{STP, \rho_s^t\}$, the best response is k source-target pairs, $\{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$ such that the attacker's utility is maximized.

Algorithm 1 *GReedy Algorithm for Network DEfense.*

```
budget ← B
I ← nodesOnPaths
repeat
  updated ← false
  bestNode ← null
  bestIncrement ← 0
  attackerBR ← getAttackerBR(graph)
  for node ∈ I do
    increment = getSecurityIncrement(d, attackerBR)
    if bestIncrement < increment then
      if d · flow(node) < Budget then
        bestIncrement ← increment
        bestNode ← node
      end if
    end if
  end for
  if bestNode! = null then
    bestNode.sampling ← bestNode.sampling + d
    budget ← budget − d · flow(node)
    updated ← true
  end if
until not updated
```

Algorithm 2 Attacker's Best Response Oracle

```
H ← {}
K ← attackerResources
Pairs ← enumerateAllPairs()
repeat
  (s*, t*) ← emptyPair
  for (s, t) ∈ Pairs do
    if U((s, t)|H) > U((s*, t*)|H) then
      (s*, t*) ← (s, t)
    end if
  end for
  H ← H ∪ (s*, t*)
until size(H) = K
```

The greedy algorithm (summarized in Algorithm 2) works as the following: we choose one source-target pair at a time that maximizes the attacker's immediate gain in utility. Let's assume some pairs H, have been chosen and we need to choose the next one. Since pairs in H have already been chosen, we know there is some probability of successfully attacking a target t, denoted by q^t, which may or may not be 0. If we choose source s, and target t, the additional utility we will get will be:

$$\begin{aligned} U((s,t)|H) &= [1 - \rho_s^t \cdot (1 - q^t)] \cdot \tau_t - q^t \cdot \tau_t \\ &= (1 - \rho_s^t) \cdot (1 - q^t) \cdot \tau^t \end{aligned} \tag{13}$$

Note if H is empty, all $q^t = 0$ and $U((s,t)|\{\}) = (1 - \rho_s^t) \cdot \tau^t$, is the expected value of attacking t from s, intuitively. The greedy algorithm then chooses (s^*, t^*) such that $U((s^*, t^*)|H)$ is maximized.

THEOREM 1. *The attacker's oracle always returns attacker's best response to any defender's strategy.*

PROOF. Consider at any point of the algorithm, a set of source-target pairs H has been chosen. The greedy algorithm returns (s^*, t^*). We want to show (s^*, t^*) must be in the best solution conditioned on H being included. This will allow us to do induction on the number of pairs chosen. Let's denote the optimal solution by C^*. Then the first pair

chosen must be in C^* because H_1 is empty (no condition required). And if the pairs up to k are all in the optimal solution, implying H_k is in C^*, therefore the $k+1$-th pair must be in the optimal solution.

This implies that we want to show (s^*, t^*) must be in the best solution conditioned on H being included. To show this by contradiction, we consider another candidate best solution C (having H) which does not have (s^*, t^*). Two cases to consider:

1. C contains no pair attacking target t^* other than those in H. Then we find an arbitrary pair (s', t') in C but not in H (such set is denoted as $C \backslash H$) and replace it by (s^*, t^*). We know the attacker gains exactly $U((s^*, t^*)|H)$ (since no other pair in $C \backslash H$ attacks t^*) and loses at most $U((s', t')|H)$ (since there might be another pair in $C \backslash H$). Recall $U((s^*, t^*)|H) \geq U((s', t')|H)$ given how (s^*, t^*) is chosen, the new solution $C + (s^*, t^*) - (s', t')$ must be better than C which also includes H, leading to a contradiction.

2. $C \backslash H$ has at least another pair attacking t^* that is not (s^*, t^*). Let the pair be (s', t^*). We replace it by (s^*, t^*). We know $\rho^{t^*}_{s^*} \geq \rho^{t^*}_{s'}$ because $U((s^*, t^*)|H) \geq U((s', t^*)|H)$. Therefore the total probability of successfully attacking t^* must increase after the replacing given other pairs in C remain fixed. Again this shows, $C + (s^*, t^*) - (s', t^*)$ is a better solution which is the contradiction.

Having reached contradiction in both points, we have shown that (s^*, t^*) must be in the best solution conditioned on H being included, implying validity of the induction step. □

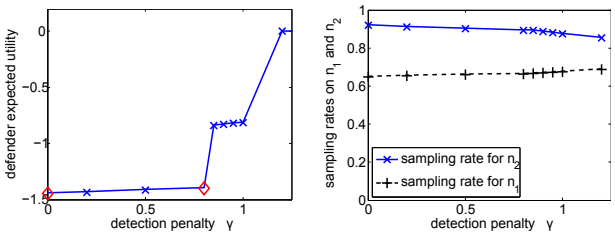
The complexity of the algorithm is $O(STn + S^2T) = O(n^3)$, where the $O(STn)$ is complexity of the initialization and $O(S^2T)$ is complexity of iteration part. S is the number of sources, T is number of targets and $n = |I|$ is the number of nodes.

5. EVALUATION

In the evaluation, we focus on exploring the trade-off between scalability and the quality of the solution. We consider the solution of the general-sum model to be optimal and compare it with the solution of the mathematical program representing the zero-sum game model, and the solution from GRANDE. Additionally, we want to explore finer properties of GRANDE, specifically, the dependency of the solution error on the discretization step of the sampling rate.

Experimental scenarios of the analyzed problem depend on a large set of parameters that affect both the performance of the algorithms, as well as the quality of produced solutions for the approximative ones. The key parameter is the graph on which the game is played; more specifically the number of intermediate nodes $|I|$, number of sources $|S|$, and number of targets $|T|$. Moreover, the degree of overlapping paths also plays an important role in the non-linear models. The detection penalty γ_s has no direct impact on performance, the defender's budget B, traffic flow f_i and discretization step d proportionally influence mainly the GRANDE algorithm.

While we conducted experiments for different graph structures, we present results only on *scale-free graphs* since these graphs are known to be the closest to general computer networks in their structure. We performed experiments with



(a) Defender's exp. utility. Red diamonds denote locally optimal solution.

(b) Distribution of defender resources between n_1 and n_2 (log x axis).

Figure 2: Impact of detection penalty γ on the solution structure. While increasing value of γ, the defender redistributes it's resources between n_1 and n_2 and his exp. utility changes (b), however, it stays equal to zero for $\gamma > 1.2$ (a).

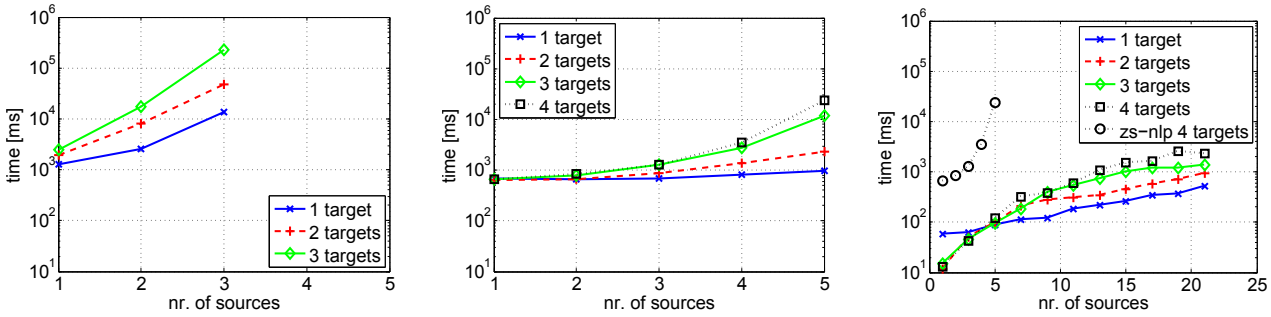
random flows (e.g. the flow at each node is set independently on the flow of the others) as well as with network-flow constrained traffic distribution (the flow at each node is computed from the network-flow equations by randomly selecting traffic sources and sinks in the network) without any direct impact on both the performance and quality of the solution. Without loss of generality, in every experiment, the traffic flow in the graph is set between $[0, 1]$ at each node. We have included the dummy target in each model to keep the graph size constant for all algorithms, even though the zero-sum model as well as the iterative algorithm never consider the attacker to attack the dummy target. The detection penalty γ_s was equal for each source, set to $\gamma_s = 1$.

5.1 Solving Non-linear Constrained Programs

To obtain an optimal solution of the program representing the general-sum game model, we use a non-linear solver to find optimal or locally optimal solution. NEOS server [4] provides on-line solvers for solving a number of programs including non-linear programs. We used LINDOGlobal [20], a non-linear constrained program solver able to find globally optimal solutions for many constrained non-linear programs. The input for the solver is a file describing the program in GAMS format, which is sent by a remote procedure call to the NEOS server using XML remote procedure call API. The solution is computed on the server and the results are sent back to the user.

5.2 General-sum vs. Zero-sum Model

As a first step, we compare the quality of the general-sum and the zero-sum game model. The difference in the solution quality between these two models will be directly affected by the value of the detection penalty γ, as it can be observed from Equations 2 and 3. For the example described in section 3.4, the trend of defender's expected utility while varying γ is depicted on Figure 2a. As γ is increased (i.e. the attacker is penalized more for a detected malicious packet, thus the utility of players is further from zero-sum), the defender's expected utility rises. Two rapid transitions occur for ($\gamma = 0.8$ and $\gamma = 1.2$) which are caused by the switch of attacker's strategies to attack the dummy target t_D. In the interval from $[0, 0.8]$, the attacker attacks from both sources, in the interval from $[0.8, 1.2]$ the attacker attacks only from one source, and from $[1.2, \infty]$, the attacker chooses not to attack at all. Figure 2b shows the distribution of defender's



(a) General-sum NLP (b) Zero-sum NLP (c) GRANDE

Figure 3: Scalability of the three models with respect to the number of sources and targets on a scale-free graph of size 100 (note the different scale of the x axis). For comparison of GRANDE with other two approaches, we have displayed one result of the zero-sum NLP in the Figure (c) denoting performance of the zero-sum model for 4 targets.

resources between nodes n_1 and n_2 as γ is varied. Notice that the defender has to redistribute the resources to discourage the attacker from attacking, while still adhering to the latency constraints, i.e. the node sampling rates multiplied by the flows through the nodes $x_1 \cdot f_1 + x_2 \cdot f_2 \leq B = 6$. The results of the zero-sum model are equal to the results of the general-sum model with $\gamma = 0$.

5.3 Performance

Figure 3 depicts the scalability of the three main algorithms. The NEOS server is limited by the size of uploaded GAMS files, thus we weren't able to compute solution for general-sum NLP beyond 3 sources and 3 targets (as the GAMS files describing the model are over 1MB large). For the zero-sum NLP, the limit was reached at 5 sources and 4 targets. However, even on these problem sizes it is possible to see performance trends: the runtime of the general-sum as well as zero-sum NLP is exponential (even in logarithmic coordinates) in the number of sources (i.e. number of attacker strategies) and time needed to solve a graph with 3 sources and 3 targets is over three minutes in average for the general-sum NLP. Using the zero-sum NLP, we are able to compute solutions on graphs with 5 sources and 4 targets in approximately 30 seconds.

Comparing GRANDE to mathematical program formulations, we can observe its superiority on Figure 3c (performance of the zero-sum NLP on 4 targets is depicted as a single line on the left of the chart). The performance of GRANDE is linearly dependent on the discretization step d (see Section 4.4). The algorithm is able to find solution on graphs with 20 sources and 4 targets in seconds, having the discretization step set to $d = 0.01$ (which is sufficient to compute solutions with an average error under 10%, see Section 5.4). The largest problem tried, with 2000 nodes, 200 sources and 20 targets was solved in approximately 50 hours on a standard PC.

5.4 GRANDE Solution Error

The theoretical error bounds of greedy algorithms optimizing submodular set functions shown in [21] are valid only for zero-sum settings. We explore the error of GRANDE compared to the general-sum game solution, which can be possibly unbounded. It is necessary to set the discretization step of GRANDE to a specific step, which has a direct impact on the quality of solution. To evaluate the error, we have varied both the discretization step as well as attacker loss expressed by γ. The budget constraint of the defender was fixed to $B = 4$.

For every graph, we have computed the defender's resource allocation X^* and attacker's best response P^* using the program of the general-sum NLP, which served as a reference optimal solution (even if only a local optimum was found by the solver, due to lack of other globally optimal techniques). Then we computed the defender's resource allocation X^G using GRANDE. To evaluate the quality of X^G, we have found the attacker's best response P^G to X^G using the general-sum utility formulation. Then, we computed the error of X^G as $err = \frac{U_D(X^*,P^*) - U_D(X^G,P^G)}{\mathcal{T}}$, where $\mathcal{T} = \sum \tau_t$ (maximum achievable error).

Figure 4 quantifies errors of GRANDE from 50 different scale-free graphs with two sources and two targets (limit given by the necessity of computing optimal solution using the general-sum NLP). The graph depicts median error (denoted by the circle) with 25th and 75th percentile (denoted by a thick bar) and maximal and minimal error (denoted by whiskers). As we refine the discretization step from 1 to 0.001 (i.e. the sensors can increase their sampling rate by 0.1% for $d = 0.001$), the quality of solution increases. An average error under 10% is reached when the discretization step is set to $d = 0.01$, however GRANDE is able to compute strategies with discretization step set to 0.001 resulting into errors under 5%. The variance is highest for $d = 0.1$ as the solutions varied from close-to-optimal to 100% ineffective.

6. CONCLUSION

Effectively securing large computer networks without stifling the quality of service is a practical and theoretical challenge of grave impact in the real-world. In this paper, we have outlined the mathematical model of the network security domain. We have provided the mathematical formulation for the two person security game between the defender and the attacker, where the attacker sends malicious packets from some (known) set of sources and the defender uses packet inspections to detect such malicious traffic. While we provide results for this optimal mathematical formulation, the complex non-linear calculations render the model intractable for *large* computer networks. Thus, we also introduced a zero-sum simplification of the original model as well as GRANDE, a novel error-bounded approximation algorithm that relies on the submodularity properties of network

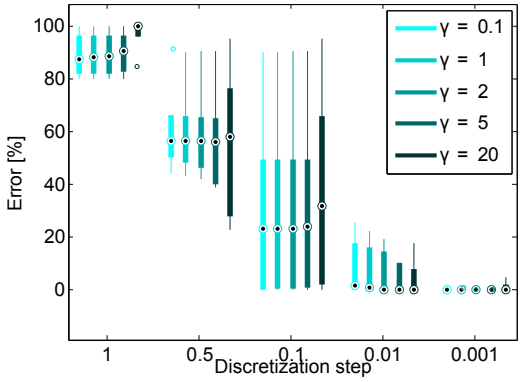


Figure 4: Error of GRANDE evaluated over different scale-free graphs. The errors are grouped according d, and in each group, γ was set to $\{0.1, 1, 2, 5, 20\}$.

security. We validate these algorithms experimentally, and show that GRANDE, on average, produces results with orders of magnitude higher solution quality as projected by theoretical worst case bounds. This work contributes by outlining the challenges present in the network security domain, and by introducing state-of-the-art algorithms to assist defender's by suggesting optimal network security strategies.

7. ACKNOWLEDGMENTS

This research was supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001, by the Czech Ministry of Education, Youth and Sports (grant no. LH11051), by the Czech Science Foundation (grant no. P202/12/2054) and by the AirForce Office of Scientific Research, Air Force Material Command, USA (grant no. FA8655-10-1-3016).

8. REFERENCES

[1] T. Alpcan. *Network Security: A Decision and Game-Theoretic Approach*. Cambridge University Press, 2010.

[2] L. Chen and J. Leneutre. A game theoretical framework on intrusion detection in heterogeneous networks. *IEEE Transactions on Information Forensics and Security*, 4(2):165–178, 2009.

[3] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

[4] J. Czyzyk, M. Mesnier, and J. Moré. The neos server. *Computational Science & Engineering, IEEE*, 5(3):68–75, 1998.

[5] S. Dughmi. Submodular Functions: Extensions, Distributions, and Algorithms. A Survey. *CoRR*, 2009.

[6] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[7] E. Halvorson, V. Conitzer, and R. Parr. Multi-step Multi-sensor Hider-Seeker Games. *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[8] 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 *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Taipei, Taiwan*, 2011.

[9] M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, and F. Ordóñez. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. *Interfaces*, 40:267–290, 2010.

[10] M. Kodialam and T. Lakshman. Detecting network intrusions via sampling: a game theoretic approach. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. (INFOCOM)*, volume 3, pages 1880–1889. IEEE, 2003.

[11] D. Korzhyk, V. Conitzer, and R. Parr. Security games with multiple attacker resources. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[12] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.

[13] A. Krause, A. Roper, and D. Golovin. Randomized sensing in adversarial environments. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2133–2139, 2011.

[14] M. Manshaei, Q. Zhu, T. Alpcan, T. Basar, and J. Hubaux. Game theory meets network security and privacy. *EPFL, Lausanne, Tech. Rep*, 2010.

[15] G. Nemhauser and L. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. *Studies on Graphs and Discrete Programming*, pages 279–301, 1981.

[16] H. Otrok, M. Mehrandish, C. Assi, M. Debbabi, and P. Bhattacharya. Game theoretic models for detecting network intrusions. *Computer Communications*, 31(10):1934–1944, 2008.

[17] P. Paruchuri, J. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 895–902, 2008.

[18] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[19] J. Pita, C. Kiekintveld, M. Tambe, E. Steigerwald, and S. Cullen. GUARDS - Game Theoretic Security Allocation on a National Scale. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2011.

[20] L. Schrage and I. LINDO Systems. Optimization modeling with lingo. 1999.

[21] J. Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 67–74, 2008.