A Malware Detector Placement Game for Intrusion Detection [*]

Stephan Schmidt[1], Tansu Alpcan[2], Şahin Albayrak[1], Tamer Başar[3], and Achim Mueller[2]

[1] DAI-Labor, TU Berlin, Franklinstr. 28, 10587 Germany
[2] Deutsche Telekom Laboratories, TU Berlin, Ernst-Reuter-Platz 7, 10587 Germany
[3] Coordinated Science Laboratory, U. of Illinois, 1308 West Main Street, Urbana IL 61801

Abstract. We propose and investigate a game-theoretic approach to the malware filtering and detector placement problem which arises in network security. Our main objective is to develop optimal detector algorithms taking into account attacker strategies and actions. Assuming rational and intelligent attackers, we present a two-person zero-sum non-cooperative Markov security game framework as a basis for modeling the interaction between the attackers who generate malware traffic on a network and a corresponding intrusion detection system (IDS). Thus, we establish a formal model of the detector placement problem based on game theory and derive optimal strategies for both players. In addition, we test the strategies obtained in a realistic agent-based network simulation environment and compare the results of static and dynamic placement scenarios. The obtained IDS strategies and the corresponding simulation results provide interesting insights into how to optimally deploy malware detectors in a network environment.

Keywords: network-based intrusion detection, monitor placement, game theory

1 Introduction

In contemporary communication infrastructures, IP-based computer networks play a prominent role. The deployment of these networks is progressing at an exponential rate as different kinds of participants such as corporations, public authorities and individuals, rely on services and communication systems more sophisticated and complex than ever before. With regard to information security, this leads to new challenges as large amounts of data need to be transferred over open networks which may hold malicious content such as worms, viruses, or trojans. In order to deal with these threats, network operators deploy intrusion detection systems (IDS). An IDS tries to detect attempts to infiltrate a target system residing in a given network by observing various events in the networked

---

system through malware detectors and sniffers, terms which we will interchangeably in the context of this paper. When the IDS encounters a situation which it classifies as an attack by anomaly detection or by applying a set of predefined rules, it takes corresponding measures. The respective response action can be carried out in an automated or semi-automated fashion as well as by a system administrator after receiving a relevant notification from the IDS.

Network security measures such as monitors (and detectors) can be implemented in the network itself as well as at the hosts connected to the access routers of the network. The host-based approach has its merits, especially with respect to the scalability of the resulting security framework. However, as the hosts are generally not under the control of network operators, they have no way of enforcing a certain network-wide security policy. In this paper, we will focus on the deployment of detection capabilities as part of the network itself.

1.1 Problem Definition and Related Work

Network monitoring can be classified into *active* and *passive* monitoring. The former is mainly concerned with actively probing the network for identifying link failures or measuring delay times for traffic traversing the network as described, for example, in [1] and [2]. In the passive scenario, on the other hand, packets are sampled with the intention of gaining a better understanding of the flow distribution in the network, identifying malicious packets and determining on which routes they travel through the network. The approach in this paper is to gather information by passive link monitoring and using it to take responsive action against the adversary.

In terms of their mode of operation on the other hand, the monitors can be classified as hardware- and software-based. Dedicated hardware can be used to tap directly into links and inspect the traffic passing over the link. Based on the traffic volume, the thoroughness of the inspection needs to be adjusted from simple pattern recognition on gigabit links down to stateful inspection on lesser utilized links. Another hardware-based solution is the deployment of routers with inherent built-in monitoring capabilities such as Cisco's NetFlow solution. Further insight into NetFlow's mode of operation and an improvement proposal on its current implementation can be found in [3]. Software-based monitors, on the other hand, are commonly deployed on the routers. In particular, such monitors can be realized in autonomous software agents (ASAs). As a matter of fact, the test framework utilized in this paper for running the simulations also operates at an agent-based level [4].

When devising a network-based IDS, special attention needs to be paid to the placement of malware detectors as well as monitors in general. These may be implemented as part of existing network elements or deployed as separate devices. Naturally, it is not feasible to install sniffers or activate them on all network links all the time due to the cost incurred by the operation of the devices in terms of capacity, delay, and energy penalties. This is especially important in the case of ad-hoc networks where the nodes have limited energy. The identification of the strategic points for deploying active network monitors, defined

in the scope of this paper as the *detector placement problem*, is computationally complex due to the multiple trade-offs involved. The *monitor placement problem* which is more general in its nature has been thoroughly studied as a result of not only its security related implications but also of its importance in many other network-related contexts apart from security, such as network management.

One example of a mathematical solution formulation within this context has been studied in [5]. The algorithms proposed therein compute the set of routers or links for monitor deployment by maximizing a defined objective function, e.g., the total number of sampled distinct packets. Naturally, this approach can be extended to the problems of network security expounded in this paper, assuming that the deployed monitors are able to detect threats on an IP packet-level, for example by performing pattern matching against a known malware signature.

Apart from standard optimization techniques, it is also possible to study the detector placement problem using Betweenness Centrality algorithms, which originate from the field of social network analysis [6]. This family of algorithms determines a set of nodes which are most influential with respect to the overall communication between all nodes within a given social network. The application of this approach to the monitor placement problem for intrusion detection is examined in [7].

1.2 Summary of Contributions

This paper proposes a game-theoretic approach to the aforementioned detector placement problem. The mathematical field of game theory provides an extensive set of tools to model real-life network security problems. In particular, attackers attempting to gain unauthorized access to a target system residing in the network or compromise its accessibility through distributed denial of service (DDoS) attacks must be –in the worst case– expected to have complete knowledge of the internal configuration of the network such as routing states or detector locations. Thus, attackers need to be viewed as rational and intelligent players who respond to the actions taken by the IDS by choosing different targets or routes to inject the malware. Due to this adaptive behavior of the opponent, in our view, the approaches mentioned in the previous subsection are not sufficient. More precisely, while the algorithms proposed in the literature will possibly yield large sampling rates over all packets traversing the network, this may not be the case for the sampling rate of infectious packets if the attacker's behavior is not taken into account.

Motivated by the shortcomings of the approaches described above, we model the detector placement problem as a two-person non-cooperative game between the attacker and the IDS. Such a game-theoretic approach to network intrusion detection has been proposed in [8], where the authors employ a problem definition based on packet sampling. In particular, they work with the notion of a sampling budget, where the sampling effort can be distributed arbitrarily over the links of the network. This approach is useful in architectures where the routers have *built-in sampling capabilities*. In addition to the scenario considered in [8], we develop a framework that is also applicable for determining the

deployment of *dedicated hardware devices* for detecting malicious packets. We also examine a version of the game where the routing states change frequently as is the case in for example ad-hoc networks. This is modeled explicitly by the routing state change matrix and influences the strategies of the players (cf. Section 3.4).

In network practice, the main benefit of using a game-theoretic approach in solving the monitor placement problem is that due to the existence of a mixed-strategy saddle-point equilibrium, even the most skilled attacker has no way of exploiting an optimal detector placement setup when our approach is utilized in a network-based intrusion detection system. Game theory ensures that, assuming guaranteed local detection rates at the links, the global detection rate of a network-based IDS will never fall below a certain limit, which is an important feature for an IDS in general and in critical infrastructures in particular. As a concrete example, the worst-case bounds would guarantee a minimum level of service of a critical web server resource during a denial-of-service (DDoS) attack.

The main contributions of this paper can be summarized as:

- Introduction and investigation of a quantitative game-theoretic framework for the detector placement problem taking the attacker's behavior into account.
- Computation of optimal IDS and attacker strategies for different scenarios within the framework.
- Presentation of a realistic, agent-based simulation environment, in which illustrative example cases are investigated.
- Conduction of simulation studies where different strategies of the players are compared and the optimality of the proposed solutions is verified.

The outline for the rest of the paper is as follows: First, we introduce the necessary mathematical notations used and formalize the problem within a game-theoretic framework. Additionally, we compute the optimal (saddle-point) strategies for both players [9]. In Section 4, the agent-based framework for running realistic simulations based on the previously defined security game is presented. Using this simulation environment, we subsequently verify the effectiveness of the obtained solution and compare it to the results obtained using uniform as well as static strategies for each of the players. The final section contains concluding remarks as well as an outlook on possible further extensions of the game-theoretic framework.

2 The Approach and Game-Theoretic Formulation

We represent the network to be examined by an undirected graph $G = (V, E)$, where V is the set of vertices and E the set of links. Let $v_i v_j$ denote as the link between the nodes v_i and v_j. We have an attacker player controlling a subset of vertices $V_A \subseteq V$ (e.g. bot nets), while the IDS player is trying to protect a set of target systems $V_T \subseteq V$, where we make the assumption of $V_A \cap V_T = \varnothing$ without loss of generality. We note that the vertices may represent

entire subnetworks with multiple hosts as well as a single network device. This means that an attacker node $v \in V_A$ may be either a single client or a subnet from which even multiple distinct attackers operate, yet are mathematically treated as a single attacker.

We consider a 2-player (one representing the attacker(s) and one the IDS) zero-sum finite Markov game model similar to the one described in [10], where each player has a finite number of actions to choose from. The game is a zero-sum one due to the diametrically opposing interests of the players, as the attacker is trying to intrude into the target system while at the same time the IDS tries to prevent this from happening undetected.

The attacker's action space is defined as $\mathcal{A} := V_A \times V_T = \{A_1, A_2, \ldots, A_{\mathcal{A}_{max}}\}$, representing the attack routes available to the attacker from a certain node $v_i \in V_A$ from which the attack is initiated to a target $v_j \in V_T$. On the other side, the action space of the IDS is denoted by $\mathcal{D} = \{D_1, D_2, \ldots, D_{\mathcal{D}_{max}}\} \subseteq E$, which is the set of links on which the sniffers can be deployed or activated. This is only possible in the network core controlled by the IDS operator, i.e., the set $\{v_i v_j \in E$ such that $v_i, v_j \notin (V_A \cup V_T)\}$. For simplicity of the analysis, we assume that the attacker chooses a single attack state, i.e. attacker-target vertex pair, and the IDS deploys a single sniffer at a given time. This assumption on the IDS is comparable to the sampling budget assumptions of earlier studies and can easily be extended to multiple detectors.

The players interact on a network consisting of a set of nodes whose routing tables may change randomly at discrete time instances with a predefined probability. We additionally investigate the cases of static routing and imperfectly functioning (defective) detectors under static routing. These constitute the underlying stochastic systems on which the players interact. Let us represent each possible routing configuration as the set of routing states, $\mathcal{R} = \{R_1, R_2, \ldots, R_{\mathcal{R}_{max}}\}$. We will not give an exact mathematical definition of $\mathcal{R}$ since it depends on the routing protocol employed in the underlying real-life network. In *session routing* for example, for each source-sink pair $(v_i, v_j) \in \mathcal{A}$ there is a distinct path on which the packets are routed; on the other hand, in architectures that are not flow-based, all packets with the same target IP address arriving at a certain router will be forwarded to the same next router. For the sake of clarity, we will employ a simple routing protocol in our illustrative example and postpone the definition of $\mathcal{R}$ to Section 3. Similarly, we can model the failures of the detectors probabilistically to obtain a set of detector states similar to the routing ones. Here, we model the network characteristics (routing configuration changes or detector failures) as a finite-state Markov chain, which enables us to use well-established analytical tools to study the problem.

The probability of the network routing or set of detectors being in a specific state is given by the vector $\mathbf{x} := [x_1, \ldots, x_{\mathcal{R}_{max}}]$, where $0 \leq x_i \leq 1 \ \forall i$ and $\sum_{i=1}^{\mathcal{R}_{max}} x_i = 1$. The transition probabilities between environment states are then described by the transition matrix M. Different from [10], the player actions do not have any effect on the routing changes or detector failures. Then, we have $\mathbf{x}(n+1) = \mathbf{x}(n)M$, where $n \geq 1$ denotes the stage of the game. Each player is
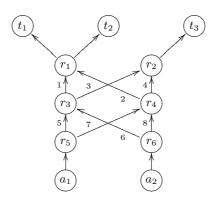
Fig. 1. Example network with two attackers $V_A = \{a_1, a_2\}$ and three target systems $V_T = \{t_1, t_2, t_3\}$. The links are labeled with numbers as a notational convenience and will be used in the figures in Section 5

associated with a set of costs that is not only a function of the other players' actions but also the state of the system. The costs of the IDS and the attacker's costs are $-c(R_l, D_i, A_j)$ and $c(R_l, D_i, A_j)$, respectively, where $R_l \in \mathcal{R}$, $D_i \in \mathcal{D}$, and $A_j \in \mathcal{A}$. We assume that each player knows its own cost at each stage of the game. The attacker cost simply represents the cost of detected malware packets, in other words a failed attack attempt, for the attacker. Notice that the IDS benefits from such a situation (approximately) equally. The IDS cost, on the other hand, represents the cost of missed malware packets, i.e., a successful attack which benefits the attacker. Thus, we have a zero-sum cost structure in the game between the attacker and the IDS.

We assume in this paper that both players know the (state) transition probabilities between the routing configurations. In this case, each player knows everything about the underlying system as well as the preferences and past actions of its opponent. Hence, players may rely on well-known Markov decision process algorithms such as value iteration [11] to calculate their own optimal mixed strategy solutions to the zero-sum game. However, this assumption can be relaxed such that one or both players utilize online learning techniques as studied in [10].

3 Illustrative Example Scenarios

In this section, we will define and study a selection of Markov security games for placing network monitors based on the framework and notations presented in Section 2. To this end, the networks depicted in Figure 1 and Figure 2 are employed as illustrative examples.

3.1 Description of the Example Networks

The first example network shown in Figure 1 is composed of eleven nodes; two systems (a_1 and a_2) are controlled by the attacker from each of which he may choose to launch an attack on any of the three target systems (t_1,t_2,t_3). Two of these target systems are connected to the same access router. With respect to the notation introduced in Section 2, we therefore get $\mathcal{A} = \{a_1t_1, a_1t_2, a_1t_3, a_2t_1, a_2t_2, a_2t_3\}$, where a_it_j corresponds to the attack from a_i on t_j in the attacker's action space. We enumerate the attacks for notational convenience as $\mathcal{A} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$, respectively.

The action space of the IDS is composed of all links where the network monitor may be placed. Note that the placement of a sniffer directly before an attacked node or before an access router (r_5, r_6) is not allowed. Hence the action space of the IDS is $D = \{r_1r_3, r_1r_4, r_2r_3, r_2r_4, r_3r_5, r_3r_6, r_4r_5, r_4r_6\}$, which is enumerated as $D = \{D_1, \ldots, D_8\}$ as shown in Figure 1. Throughout this section, it will be clear from the context whether r_ir_j refers to a specific link or to the corresponding action of the IDS. The second example network in Figure 2 is
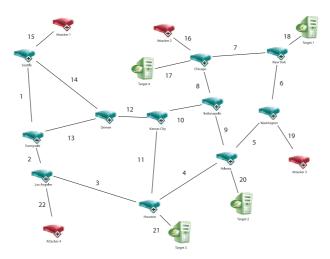

Fig. 2. Internet2 network topology with four attackers (access routers) and four target systems

inspired from the Abilene network[4] and has similar features as the first network. Both networks are rather small in order to allow to conduct tests more efficiently. However, larger network topologies can also be supported without any difficulty since our approach is not hampered by scaling issues. This is due to the fact that

---

[4] http://abilene.internet2.edu

optimal deployment is computed only once during the startup of the network-based IDS. Subsequent changes in the underlying topology due to link failures do not endanger the worst-case bound guaranteed by the mixed-strategy saddle-point equilibrium since they merely limit the ability of the attacker to reach the designated target systems.

3.2 Single Perfect Detector under Static Routing

We will start out with the simplest version of a detector placement game on the network in Figure 2. In this scenario, we assume that the IDS' budget allows only for the simultaneous operation of a single detector device. We assume first that the devices are available on all links and can be activated at any time. Alternatively, the strategies derived can also be applied in the simultaneous operation of multiple devices on a sampling basis. Furthermore, we assume that the routing tables do not change during the operation of the network, i.e., the routing is static, and hence all packets traveling from a given source to a given destination take the same path through the network. The optimal strategies[5] in this case follow from the mixed Nash equilibrium [9] of the resulting zero-sum matrix game.

3.3 Single Faulty Detector under Static Routing

In the remaining scenarios we consider the network in Figure 1. First, we modify the previous scenario by introducing the notion of *imperfect* or faulty detectors. This means that malicious packets traveling over a link on which a sniffer is deployed are detected with a certain probability p, where unlike the above scenario, $p < 1$. This behavior reflects the fact that different kinds of hardware devices available to the network operator may operate at different detection rates, for example due to different hardware equipment or signature databases.

These changes in the detection rates are modeled as finite-state Markov-chains as described in Section 2. Specifically, each monitor is associated with two states *detecting* and *not-detecting*. It can readily be observed that due to the static routing within the chosen network, the location of the monitors may be restricted to a selection of links which comprise a minimum cut between $\mathcal{A}$ and $\mathcal{D}$. Thus, we restrict the IDS' action space to 4 distinct states; in our simulations, we decided to use the minimum cut composed of the links 5, 6, 7 and 8.

3.4 Single Perfect Detector under Dynamic Routing

We next consider the possible routing configurations. Note that for each attack $A_i \in \mathcal{A}$, there are exactly two distinct paths from the attacker to the target system. Once a packet has traveled two hops and arrives at one of the routers r_3 or r_4, there is only one possible path to t_i. Therefore, the routing decision has to

---

[5] "Optimal strategy" is used here and throughout this paper to mean "mixed saddle-point strategy"

be made after the first hop. In real-life networks, it is possible that, even though each attack path has the same length (three hops), packets arriving at the ingress nodes r_5, r_6 will not be routed over the same outgoing link. For example, this is the case for flow-based resource reservation architectures or multi-protocol label switching (MPLS) domains often encountered in QoS-aware architectures. For the sake of reducing the number of environment states to preserve the simplicity of our security game we will, however, assume that all packets arriving at node r_i will be routed over the same outgoing link $r_i r_j$ but this routing configuration changes from time to time for load balancing purposes or due to failures.

As a result, we obtain four possible environment states corresponding to the four routing configurations in the example network. Throughout the remainder of this section, we will use the notational convention $\mathcal{S} = \{ll, lr, rl, rr\}$, where for example $s_3 = rl$ denotes the configuration where all packets arriving at r_5 will be routed to the right and all packets arriving at r_6 will be routed to the left. The colloquial terms "left" and "right" hereby refer to the intuitive graphical interpretation arising from Figure 1. As stated in Section 2, we characterize the routing configuration changes on the network as a finite-state Markov chain.

3.5 Multiple Perfect Detectors under Static Routing

Finally, we examine a scenario where the network operator's budget allows for the placement of more than one monitor at a time. Therefore, we relax our requirement that only one sniffer may be placed at a time. We investigate through simulations the effect of deploying multiple sniffers utilizing the optimal mixed (randomized) strategies obtained in the previous cases. The observant reader will notice that the number of possible states for the attacker and the IDS as well as for the environment was kept rather small in all of the above scenarios for instructive reasons and test data generation. Possible scalability issues arising from large player action spaces can be dealt with using a hierarchical approach and clustering schemes. We will go into further detail on such issues in Section 6.

At this point, we find it useful to reiterate an important assumption. Naturally, the network carrier or service provider implementing the IDS will be aware of the current routing configuration or detector states at all times. However, it is vital to bear in mind that we are dealing with rational and intelligent attackers. Therefore, we assume that the attacker is also aware of the routing configuration or detector capabilities as a worst-case scenario. Considering the various tools available to users for tracing the packet routes, this assumption is in fact not far-fetched. In practice, after deciding on an attack $a_i t_j$, the attacker can run a route trace from a_i to the chosen target system to find out over which path his attack packets will be sent. Similarly the attacker can also deduce the capabilities of the detectors to some extent. Hence, if a static deployment of sniffers were used, the attacker would be able to adjust his strategy accordingly.

3.6 Player Strategies

In order to compute the equilibrium strategies for the examples given above, we need to numerically define the cost function to be employed. To this end, let $p(s_l, A_i)$ denote the path of the packets of attack $A_i = a_m t_n \in \mathcal{A}$ traveling from a_m to t_n under routing configuration s_l. In addition, let the IDS deploy a sniffer at D_j. We then define the cost function c of the attacker as

$$c(s_l, A_i, D_j) = \begin{cases} 1 & \text{if } D_j \in p(s_l, A_i) \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

This means at the same time that the benefit of the IDS is 1 if the current attack uses a route on which a sniffer (operating at the moment without failures) is deployed and -1 in case it traverses the network undetected (due to lack of sniffers on its path or detector failures). In the simple case of Section 3.2 the optimal (mixed) strategies can be calculated simply by solving a zero-sum matrix game.

In all of the other scenarios we compute the optimal strategies of the players offline using a modified value iteration algorithm described in [12]. Let us define for a player the expected reward for the optimal policy starting from state $s \in \mathcal{S}$ as $V(s)$ and the expected reward for taking action a when the opponent responds with o as $Q(s, a, o)$. Here if the player is, for example, the attacker then we have $a \in \mathcal{A}$ and $o \in \mathcal{D}$. Then, one can use a Markov decision processes' value iteration-like algorithm:

$$V(s) = \min_{\Pi^A} \max_{o \in \mathcal{D}} \sum_{a \in \mathcal{A}} Q(s, a, o) \Pi^A \tag{2}$$

$$Q(s, a, o) = R(s, a, o) + \gamma \sum_{s' \in \mathcal{S}} T(s, s') V(s'), \tag{3}$$

where T is the state transition matrix. Consequently, the optimal strategy for the player (here attacker) is:

$$\Pi^A = \arg\min_{\Pi^A} \max_{o \in \mathcal{D}} \sum_{a \in \mathcal{A}} Q(s, a, o) \Pi^A \tag{4}$$

We refer to [12] and the references therein for the details of the algorithm.

The resulting attacker strategy matrix Π^A is a $|R| \times |A|$ matrix, where the entry π_{ij}^A denotes the probability of choosing attack pattern A_j when the current system state is s_i. In analogy, the IDS strategy matrix entry π_{ij}^D contains the likelihood of choosing the link corresponding to the IDS action state D_j. Note that $\sum_j \pi_{ij} = 1 \; \forall i$ since the probabilities of choosing an attack or defensive action must add up to 1 for a given state. The optimal attacker and IDS strategies calculated for the given set of parameters and the scenario in Section 3.4 are given as an example:

$$\Pi^A = \begin{pmatrix} 0.125 \ 0.125 \ 0.250 \ 0.125 \ 0.125 \ 0.250 \\ 0.147 \ 0.147 \ 0.207 \ 0.147 \ 0.147 \ 0.207 \\ 0.147 \ 0.147 \ 0.206 \ 0.147 \ 0.147 \ 0.206 \\ 0.125 \ 0.125 \ 0.250 \ 0.125 \ 0.125 \ 0.250 \end{pmatrix} \tag{5}$$

and

$$\Pi^D = \begin{pmatrix} 0.3 \ 0 \ 0.3 \ 0 \ 0.2 \ 0.2 \ 0 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 0 \ 0.5 \ 0.5 \ 0 \\ 0 \ 0 \ 0 \ 0 \ 0.5 \ 0 \ 0 \ 0.5 \\ 0 \ 0.3 \ 0 \ 0.3 \ 0 \ 0 \ 0.2 \ 0.2 \end{pmatrix} . \tag{6}$$

4 Network Security Simulator Environment

In this section, we will present the Network Security Simulator (NeSSi), a network simulation environment used for conducting the test series described in the previous section. NeSSi was designed with the objective of extending conventional network simulation tools by incorporating features which allow detailed examination and testing of security-related network algorithms. The main focus of NeSSi is to provide a realistic packet-level simulation environment for the developed algorithms. Hence, NeSSi plays a significant role as a testbed for the development of a network-level IDS which needs to efficiently detect and eliminate various malware such as viruses, worms, and trojans as they are traversing the network before reaching their designated target.

NeSSi is implemented in the Java programming language and built upon the Java-based intelligent agent componentware (JIAC) framework [4]. JIAC is a service-based middleware architecture based on the agent paradigm. Hence, agents are used within the simulator for modeling and implementing the network entities such as routers, clients, and servers. The underlying JIAC agent platform provides a rich and flexible basis for implementing and testing of various methods and algorithms in NeSSi. It allows for combining the partial knowledge of the agents residing in the network for identifying and eventually eliminating IP-based threats by monitoring the structure of the encountered IP traffic and the behavior of potentially compromised target systems. The ambitious goal of the ongoing research in our work is to be able to detect previously unknown threats through learning schemes and agent-based software monitors. Currently, we are striving to make NeSSi available to the research community by releasing it under an open source license.

The front-end of NeSSi consists of a graphical user interface that allows the creation of arbitrary IP network topologies. The communication between clients, servers, and routers takes place by real IPv4 packet transmission. This is realized using communication services between the agents. Built upon the network layer, the simulator emulates TCP and UDP protocols on the transport layer. At the application layer, the HTTP and SMTP protocols are emulated faithfully. Furthermore, NeSSi allows to capture the packets that traverse over a link and write them to the common TCPDump file format. The TCP/IP stack is emulated

in such a way that the TCP payload can be extracted with standard network inspection tools and the application layer content such as HTTP is displayed with the appropriate content. Thus, the results obtained through the simulation tool are applicable in real IP networks and can later be directly transferred. Moreover, different types of routing protocols encountered in real-life IP networks, static as well as dynamic, are supported in NeSSi.

As part of NeSSi, a sniffer agent is implemented which can be deployed on a set of links. This means that once one or multiple packets containing a previously generated signature, indicating malicious content, traverses a sniffed link, an alert is issued and/or those packets are filtered. In addition, since the TCP/IP stack is faithfully emulated, the captured traffic can also be written to a common dump file format for later post-processing with standard inspection tools.

As discussed in the previous sections, it is not feasible to deploy such monitors on all links of the network; rather, the locations at which the monitors will be placed have to be carefully selected, taking into account the incurred costs as well as the adaptive behavior of the attacker. We utilize the NeSSi framework for testing our game-theoretic approach for monitor placement as part of an IDS and describe the obtained simulation results in the next section.

5 Simulation Results

We study the illustrative scenarios of Section 3 numerically through simulations on the networks depicted in Figure 1 and Figure 2 carried out in the NeSSi environment described in Section 4. Specifically, we examine how beneficial it is for each player to utilize the optimal strategy by comparing and contrasting its results with the ones of uniformly distributed attacks and monitor deployments as well as static monitor placement at a single link.

Each simulation consists of a period of 1000 or 5000 time steps in which the attacker and the IDS update their actions, i.e. the attacker sends a malware packet with a known signature over a chosen attack path and the IDS deploys the sniffer at a link. Here we do not specify the time interval between steps but assume that it is long enough to satisfy the information assumptions made earlier. It is worth noting that the malware packets sent on the network simulated in NeSSi are real UDP packets and are captured by a realistic sniffer implementation using pattern matching algorithms and a malware signature database to filter out the packets.

First, we simulate Scenario 3.2 on the network in Figure 2 where the attacker and the IDS use the optimal random strategy calculated. The results are depicted in the left graph of Figure 3. The link numbers on the x-axis refer to the specific links with the respective labels in Figure 2 while the number of total and captured packets as well as time intervals when the detector is active on the respective link are plotted on the y-axis. Since both players use optimal strategies, some links are not even used, i.e. the total number of packets is also 0. It is interesting to note that a single link plays a significant role due to the fact that the routing

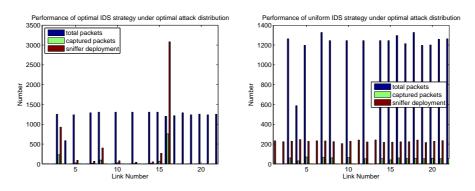tables do not change over time. This result is compared with the scenario where

Fig. 3. Performance of IDS's *optimal* (left) and *uniform* (right) monitor placement strategy on the Internet2 network topology from Figure 2 under optimal random attack and static routing

the IDS plays a uniform random placement strategy, depicted in the right-hand graph of Figure 3. As expected, the optimal strategy performs better than the uniform one in terms of the malware packets captured or filtered out.

We next relax the assumption of perfect detectors and consider the case in Section 3.3 on the network in Figure 1. The graph in Figure 4 again depicts the number of total and captured packets as well as time intervals when the detector is active versus the link numbers which refer to the respectively labeled links. We observe that the detection rate is much worse than in the previous scenario due to imperfect detectors and exploitation of their defects by the attackers. However, running the same scenario under uniform random attacks leads to a drastic increase in the aggregate number of filtered packets, as shown in the right-hand graph of Figure 4. As a result, we conclude that if the attackers find a way of exploiting the defects of the sniffers, the IDS performance degrades significantly even when an optimal strategy is deployed. Next, we study the scenario described in Section 3.4 where a single perfect detector is deployed at a time on the same network under dynamic routing. The routing configuration on the network changes every two time steps in accordance with the state transition probabilities. In the simulations, we observe that the time average of the routing states matches well with the theoretical invariant distribution.

We simulate the case where the attacker and the IDS use the optimal random strategy (5) and (6), respectively. The performance of the optimal detector placement strategy of the IDS is shown in the left-hand graph of Figure 5. In comparison, we consider the case when the IDS places the sniffer on a link dynamically with a uniform distribution. The outcome of this scenario is depicted in the right-hand graph of Figure 5, where links are plotted on the x-axis again and the packet statistics and sniffer cycles on the y-axis. We observe that the number of packets captured is, as expected, smaller than in the previous sce-
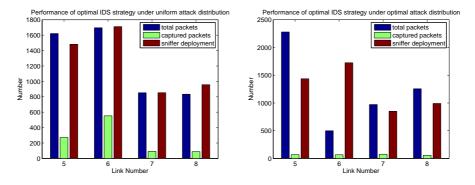
Performance of optimal IDS strategy under uniform attack distribution — Performance of optimal IDS strategy under optimal attack distribution

Fig. 4. Performance of IDS's optimal detector placement strategy under *uniform* (left) and *optimal* (right) random attacks on the network from Figure 1 with imperfect detectors
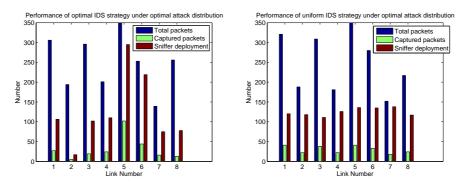

Performance of optimal IDS strategy under optimal attack distribution — Performance of uniform IDS strategy under optimal attack distribution

Fig. 5. Performance of IDS's *optimal* (left) and *uniform* (right) monitor placement strategy on the example network from Figure 1 under optimal random attack and dynamic routing

nario. Additionally, we simulate the opposite case when the attacker selects the attacks to carry out in a uniform and random fashion from its action space whereas IDS uses the optimal strategy. Unsurprisingly, the attacker is now at a disadvantage and more packets are filtered out than the ones in both of the previous simulations.

Finally, we simulate the IDS placing the sniffer at link 6 *statically*, whereas the attacker uses optimal strategy. It is interesting to observe that this static deployment does not bring a huge disadvantage to IDS in terms of the number of captured packets (left-hand graph of Figure 6). Notwithstanding, one has to remember that the attacker is unaware of the static nature of the IDS's strategy and makes decisions based on the assumption that the IDS is capable of dynamically placing the sniffer. Otherwise, it would be trivial for the attacker to exploit the static nature of the monitor and gain a significant advantage.
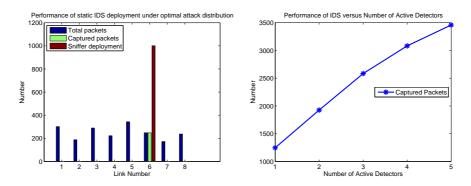

Fig. 6. Performance of IDS's *static* detector placement strategy against *optimal* random attacks on the network from Figure 1 (left) and captured packets rate dependent on the number of sniffers deployed (right) under static routing

We finally simulate the scenario 3.5 on the network 1 under static routing. The number of packets captured shown in Figure 6 exhibit a concave character, i.e., the gain from deploying additional detectors decreases as the number of sniffers increases.

6   Conclusion and Outlook

In this paper, we have proposed and investigated a game-theoretic approach to the network monitor placement in the context of network security. By modeling attackers as rational and intelligent players, we have considered a two-person zero-sum noncooperative Markov security game framework. Hence, we have taken into account the attackers' behavior as part of the detector deployment optimization process and obtained –to the best of our knowledge– a novel formulation of the problem. We have tested the strategies obtained from the

game-theoretic formulation using NeSSi in a realistic simulation environment on an example network and compared the results with the ones of uniformly random placement and attack strategies.

Our initial observations have indicated two counter-intuitive results in addition to the expected ones such as the superior performance of the optimal strategies for both players. First, the difference between optimal and uniform strategies has been relatively small and second, the static monitor placement at a carefully chosen link has performed surprisingly well. One simple explanation for the latter is that a single link may become more important due to routing configurations by making it appear on most paths. The first observation may be attributed to the small size of the example networks chosen as well as the short running time of the individual scenarios. We have also observed in the case of defective sniffers that if the attacker gain knowledge about them then the IDS performance degrades significantly as expected.

In the near future, programmable software-based monitors and routers with sophisticated capabilities are expected to be commonly deployed, leading the way to autonomous software agent (ASA)-based implementations. Development and study of algorithms for dynamically configurable and mobile ASAs which exchange the information they gathered locally with their peers, thus allowing the detection of attack types which would otherwise go unnoticed (e.g. distributed denial-of-service attacks) is a promising future research area closely related to the problem studied in this paper. Furthermore, such algorithms can be easily implemented and tested in the realistic environment provided by NeSSi.

References

1. Horton, J.D., Lopez-Ortiz, A.: On the number of distributed measurement points for network tomography. In: IMC '03: Proc. of the 3rd ACM SIGCOMM conf. on Internet measurement, New York, NY, USA, ACM Press (2003) 204–209
2. Jamin, S., Cheng, J., Yixin, J., Raz, D., Shavitt, Y., Lhixia, Z.: On the placement of internet instrumentation. In: Proc. INFOCOM 2000. Volume 1. (2000) 295–304
3. Estan, C., Keys, K., Moore, D., Varghese, G.: Building a better netflow. In: Proc. SIGCOMM '04, New York, NY, USA, ACM Press (2004) 245–256
4. Fricke, S., Bsufka, K., Keiser, J., Schmidt, T., Sesseler, R., Albayrak, S.: Agent-based telematic services and telecom applications. Communications of the ACM **44**(4) (April 2001) 43–48
5. Cantieni, G.R., Iannaccone, G., Barakat, C., Diot, C., Thiran, P.: Reformulating the monitor placement problem: Optimal network-wide sampling. Technical report, Intel Research (February 2005)
6. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology **25**(2) (2001) 163–177
7. Bloem, M., Alpcan, T., Schmidt, S., Başar, T.: Malware filtering for network security using weighted optimality measures. To appear in Proc. of IEEE Multiconference on Systems and Control, Singapore, October 1-3 (2007)
8. Kodialam, M., Lakshman, T.: Detecting network intrusions via sampling: A game theoretic approach. In: Proceedings IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. Volume 3. (April 2003) 1880–1889

9. Başar, T., Olsder, G.J.: Dynamic Noncooperative Game Theory. 2nd edn. SIAM, Philadelphia, PA (1999)
10. Alpcan, T., Başar, T.: An intrusion detection game with limited observations. In: Proc. of 12th International Symposium on Dynamic Games and Applications, Sophia-Antipolis, France (July 2006)
11. Bertsekas, D.: Dynamic Programming and Optimal Control. 2nd edn. Volume 2. Athena Scientific, Belmont, MA (2001)
12. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proc. of the Eleventh International Conference on Machine Learning, San Francisco, CA (1994) 157–163