Moving Target Defense for the Placement of Intrusion Detection Systems in the Cloud

Sailik Sengupta[1(✉)], Ankur Chowdhary[2], Dijiang Huang[2], and Subbarao Kambhampati[1]

[1] Yochan Lab, Arizona State Univeristy, Tempe, USA
{sailiks,rao}@asu.edu

[2] Secure Network and Computing Lab, Arizona State University, Tempe, USA
{achaud16,dijiang}@asu.edu

Abstract. A lot of software systems are deployed in the cloud. Owing to realistic demands for an early product launch, oftentimes there are vulnerabilities that are present in these deployed systems (or eventually found out). The cloud service provider can find and leverage this knowledge about known vulnerabilities and the underlying communication network topology of the system to position network and host-based Intrusion Detection Systems (IDS) that can effectively detect attacks. Unfortunately, deploying IDS on each host and network interface impacts the performance of the overall system. Thus, in this paper, we address the problem of placing a limited number of IDS by using the concept of Moving Target Defense (MTD). In essence, we propose an MTD system that allows a defender to shift the detection surfaces and strategically switch among the different IDS placement configurations in each round. To find a secure switching strategy, we (1) formulate the problem of placing a limited number of IDS systems in a large cloud network as a Stackelberg Game between the cloud administrator and an (external or stealthy) attacker, (2) design scalable methods to find the optimal strategies for switching IDS placements at the start of each round, and (3) formally define the problem of identifying the most critical vulnerability that should be fixed, and propose a solution for it. We compare the strategy generated by our method to other state-of-the-art strategies, showcasing the effectiveness and scalability of our method for real-world scenarios.

Keywords: Moving Target Defense · Intrusion Detection Systems
Stackelberg games

1 Introduction

System Administrators, oftentimes, use Intrusion Detection Systems (IDS) to detect on-going attacks on modern-day cyber-systems [34]. These IDS systems perform sophisticated operations – like signature-matching [3], anomaly detection [11,15], machine learning [1,17,21] etc. – to investigate either live traffic on

L. Bushnell et al. (Eds.): GameSec 2018, LNCS 11199, pp. 326–345, 2018.
https://doi.org/10.1007/978-3-030-01554-1_19

the wire (using Network-based IDS (NIDS) [2,33]), or monitor resources on a machine (using Host-based IDS (HIDS) [13,43]) to flag anomalous requests that might result in potential loss of confidentiality, integrity or availability. Cloud service providers, who host third parties on their platform, encounter non-trivial challenges when it comes to deploying these IDS that can identify vulnerabilities present in their system on account of legacy or operational constraints [12]. The foremost among these challenges is the placement of IDS on all nodes of a large network, which results in reduced performance [20,42] (also see Sect. 6.2). Moreover, third party users of the cloud platform, due to privacy and security reasons, have constraints about sharing their data with the cloud provider [6].

Thus, given a cloud service provider's performance constraints and their customer's privacy constraints, we look at the problem of placing a limited number of IDS systems in the various nodes of the cloud system. It is trivial to see that if we place IDS systems statically that only monitor certain attacks on specific nodes, an attacker (especially a *stealthy one*, i.e. one who resides inside a deployed systems and can attack a node anywhere in the network as opposed to having access to only hosts at the entry point) will eventually figure out our placement strategy [42]. At this point, a strategic attacker can always select attacks that circumvent the IDS placed, thus passing through our cloud network undetected [38]. To address this, we design a Moving Target Defense (MTD) approach for dynamic placement of IDS systems on cloud systems.

The placement mechanism for our cloud framework places both Network and Host-based IDS. We will use a NIDS called `snort` [32] for detecting malicious behavior over the network and a HIDS known as `auditd` on the hosts of our cloud system. The assumption is that NIDS is placed at the gateway of each tenant network and the HIDS is deployed on each individual VM. A dynamic switching (or MTD) strategy selectively turns `on/off` the different HIDS or NIDS systems that can be used to monitor requests or hosts, thereby shifting the detection surface at each round without the need to consider switching costs among configurations because `on/off` commands from a central server sent out only at the start of every round hardly impact performance.

The cyber-security community has mostly defined and used MTD, so far, to shift the attack surface of a system that takes away the advantage of reconnaissance that an attacker has [45]. In this work, we generalize this notion of MTD and introduce an MTD system that shifts the detection surface to keep an attacker from guessing about whether their next attack will be detected or not. In conjunction with that, the key contributions of this paper are,

- We formulate the problem of placing limited IDS systems in a large cloud-based network using MTD as a two-player Stackelberg Game between the defender and an attacker. The equilibrium of this game gives us the optimal movement strategy that the defender should use to switch between the various IDS placements.
- We obtain the utility values of the players in this game by combining (1) the Common Vulnerability Scoring System (CVSS) that has been previously used to represent the impact of attacks on the defender's system [23] and (2)

the centrality values of the nodes in which an IDS is deployed that lets us capture (i) the connectivity information and (ii) the impact on performance when an IDS is placed on that node [42].
– We design a scalable optimization problem to find the Stackelberg Equilibrium of our formulated game (Sect. 4). In this approach, we introduce an input parameter α that lets the defender balance between the security of the system and the impact on the performance of the system.
– We define the problem of finding the most critical vulnerability in a cloud environment with a strategic attacker and a multi-objective utility function and propose a method to solve it (Sect. 5).
– We demonstrate the effectiveness of our approach on a running example by comparing it to state-of-the-art deterministic, uniformly random and centrality based MTD switching strategies. We then provide experimental results in a real-world large-scale cloud-based environment that showcases the scalability of our approach (Sect. 6).

2 Related Work

Moving Target Defense [45] has been recently used to thwart a wide range of attacks against network-based [16,41] and cloud-based systems [7,9]. These methods mostly shift the attack surfaces that takes away the advantage of reconnaissance an attacker has. A stealthy and strategic adversary [5], who can reside deep within the network, can still render these methods ineffective.

For such cases, researchers have previously investigated the placement of detection systems in large network-based environments and designed both static [20] and dynamic [42] placement mechanisms based on graph-theoretic measures. Unfortunately, the former method cannot adapt its placement strategy when facing a stealthy adversary. On the other hand, the latter method, which does not incorporate the knowledge of known vulnerabilities, performs sub-optimally when facing a strategic and rational adversary.

A switching strategy for any dynamic placement method or MTD system needs to incorporate attacker modeling and thus, game theoretic reasoning for it to be effective [31,37,39]. Previously, authors in [22] have modeled an MTD system as a game called `PLADD`, based on FlipIt [40]. This work assumes that different agents control the server in different game rounds, which is an impractical setting for cloud environments. In [19], researchers assume known vulnerabilities and design a deception mechanism using a Stackelberg Model to introduce honeynets against a specific class of attackers. Authors in [36] and [35] formulate the switching between various web-stack configurations and classifiers in an ensemble respectively as a Stackelberg Game. Unfortunately, the methods to find the Stackelberg equilibrium in these cases become intractable as the number of defender strategies explodes combinatorially.

Researchers have shown that decomposition of the reward structure makes the problem of finding the Stackelberg Equilibrium computationally efficient [24]. We leverage this information and design the rewards for our game while ensuring that the Stackelberg equilibrium balances between two important metrics [23] – (1) the costs of placing IDSs (on performance, cost of countermeasure deployment etc.) and (2) the impacts on the security of our system.

Lastly, researchers have leveraged the attack graph information of a network and tried to come up with classical AI planning approaches [26] or MDP-style approaches [14,29] to find effective ways of finding critical attacks against a system. Unfortunately, these approaches cannot be easily applied in the case of dynamic systems like MTD and thus we develop an approach to find the most critical vulnerability that should be fixed in our system.

3 Game-Theoretic Modeling

In this section, we first define the threat model of our system, defining the players, their action/strategy sets using a small real-world scenario that we set up on an enterprise cloud (Fig. 1). We then describe how the rewards of this game are formulated leveraging the CVSS data and network topology information.

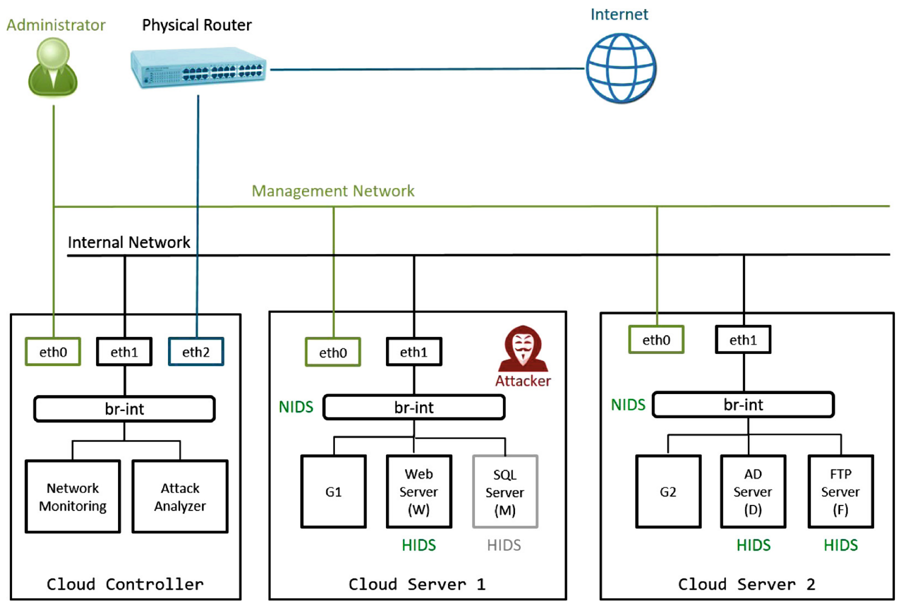


Fig. 1. Defender's system on the enterprise cloud that the attacker wants to attack.

Threat Model. In our attack model, we consider a multi-tenant cloud network. The controller node, shown in the Fig. 1, is used for network management and orchestration. The network administrator (or the defender) utilizes a management network to access controller nodes and cloud servers hosting VMs. We

Table 1. The different VMs in the defender's network, their betweenness centrality (c_b) in the graph, the known vulnerabilities in these nodes (VMs), and the corresponding Network/Host-based Intrusion Detection Systems (NIDS/HIDS) which can detect these attacks, also known as the Indicators of Compromise (IOC).

ID	VM	c_b	Vulnerability	CVE ID	IOC
a_1	G1	4	SSH Buffer Overflow	CVE-2016-6289	NIDS `sshAlert`
a_2	G2	7	`rlogin`	CVE-1999-0651	NIDS `rlogin`
a_3	W	0	Cross Side Scripting	CVE-2016-2163	HIDS `webAccess`
a_4	D	0	Weak Credentials	CVE-2001-0839	HIDS `fileIntegrity`
a_5	F	0	`vsftpd` backdoor	CVE-2015-1419	HIDS `ftpLogin`

consider two agents – the defender $\mathcal{D}$, who is trying to deploy IDS and an (external or stealthy) attacker $\mathcal{A}$, who is trying to remain undetected while attacking the system. As a running example, we will use the scenario deployed by $\mathcal{D}$ shown in Fig. 1. Furthermore, this system has a set of known vulnerabilities, that are yet to be fixed and as per our assumptions, known to both the agents $\mathcal{D}$ and $\mathcal{A}$.

We assume that the attacker $\mathcal{A}$ can be located either inside or outside the cloud network. The attacker's primary goal is to (1) compromise a VM using known vulnerabilities and (2) remain undetected while doing so. Since the attacker can utilize network probing to identify the OS and software versions, it will eventually get to know the vulnerabilities (CVEs) associated with the system, and can then systematically exploit these in order to obtain network access or elevated privileges. Furthermore, the attacker can only be detected when it attacks a vulnerability for which the corresponding IDS is in place at the time of exploitation. For stealthy attackers [5], who have to spend a lot of cost and/or effort in gaining access to an internal node, the latter is of utmost importance.

Now given the system's communication graph, we extract the set A of all the n known vulnerabilities in our system ($n = (|A|)$). For our system, we choose the a_i IDs in the first column of Table 1 to represent an *attack* (and the corresponding IDS that detects this attack). Thus, $n = 5$ and the set $A = \{a_1, a_2, a_3, a_4, a_5\}$. Note that this ID encodes a two-tuple $\langle$MachineName, CVE-ID$\rangle$. Thus, multiple attacks corresponding to a single machine will each receive a unique ID.

The defender $\mathcal{D}$, as mentioned before, has a limited budget to place only $k(< n)$ IDS mechanisms due to resource constraints. Also, we assume that, due to privacy constraints, $\mathcal{D}$ cannot place an IDS mechanism on the 'SQL Server (M)' (shown in Fig. 1). Thus, in our model, we disregard any vulnerabilities present on this node. (Note that although our system can detect a class of vulnerabilities that trigger NIDS alarms on the network interface G1 when they affect M, we exclude such vulnerabilities from our example). Now, $\mathcal{D}$ has $\binom{n}{k}$ ways in which it can deploy the k IDSs. This is the action set of $\mathcal{D}$. Formally, the defender's action set is denoted by the set $A_k = \{S \in A : |S| = k\}$. In the running example, we will assume that $k = 2$. Thus, the defender's action set is:

$$\{(a_1, a_2), (a_1, a_3), (a_1, a_4), (a_1, a_5), (a_2, a_3), (a_2, a_4), (a_2, a_5), (a_3, a_4), (a_3, a_5), (a_4, a_5)\}$$

Since, we assume a strong adversary who either knows or can find out all the attacks in our system, the action set of the attacker is the attack set $A = \{a_1, a_2, a_3, a_4, a_5\}$ itself.

In game theory, this action set is often referred to as the set of pure strategies, where each action (either a placement strategy or an attack) is a pure strategy (for $\mathcal{D}$ or $\mathcal{A}$ respectively). As stated earlier, if a defender chooses a pure strategy, i.e., any one out of the ten pure strategies shown, to deploy k IDS systems, the attacker, with reconnaissance on its side, will eventually figure out $\mathcal{D}$'s strategy and start choosing attacks that do not trigger these alarms. In order to address this limitation, the defender can play a mixed strategy, i.e. have a probability associated with playing each pure strategy and at the start of each round pick one by randomly sampling a pure strategy from the set of pure strategies. Note that this is similar to applying the concept of Moving Target Defense where the defender chooses to switch randomly among the different deployment configurations (i.e. by choosing one of the ten IDS placements in our case) at the start of each time period.

Common Vulnerability Scoring System (CVSS). The CVSS metric provides two quantitative scores for each CVE present in our system – (1) the Impact Score (IS) that represents the effect a particular attack has on the Confidentiality, Integrity, and Availability of a system and (2) the Exploitability Score (ES), which encodes the complexity of actually exploiting a particular vulnerability. The system defines a way to combine both of these scores to calculate a third score, known as the Base Score (BS) that tries to consider both the impact of an attack *vs.* the difficulty in exploiting it.

The CVSS scores thus leverage the knowledge of cybersecurity experts across the globe to provide a numerical value corresponding to each (known) vulnerability that reflects its severity and expertise necessary to exploit it. We, inspired by other research work before us [27,36,44], use the CVSS to calibrate the reward values of our game.

3.1 Stackelberg Games

Having defined the players and their action (or pure strategy) sets, there are additional real-world aspects that we want to incorporate in the formulation of our game. One such aspect is that the defender, who hosts the system that an attacker attacks, plays first. To accurately model this scenario, we use the concept of Stackelberg games in which one player ($\mathcal{D}$) acts before the other player ($\mathcal{A}$) plays and find the Stackelberg Equilibrium of these games, in which the leader's ($\mathcal{D}$) strategy is contingent upon the fact that the follower ($\mathcal{A}$) can observe D's strategy and play accordingly. Thus, in this adversarial leader-follower game, $\mathcal{D}$ can simulate $\mathcal{A}$ in their mind and decide on a mixed strategy that gives it the highest utility keeping in mind (that a rational) $\mathcal{A}$ will choose the best action ($\in A$), i.e. the action that maximizes A's reward, in response.

3.2 Utility Modeling

Having designed the action sets of both the players, we can now specify the utilities for both the players when each of them commits to a pure strategy. Note that just to enumerate all the utility values for our game we would have to specify $2 \cdot \binom{n}{k} \cdot n$ values corresponding to the reward values for each of the players $\mathcal{D}$ and $\mathcal{A}$ in the normal form game matrix. With this general reward structure, finding the mixed-strategy Stackelberg equilibrium of this game would be computationally inefficient, specifically $O(\binom{n}{k})$ [10]. Thus, we now devise a particular reward structure that captures all the aspects of our problem and lets us efficiently compute the equilibrium strategy.

For each attack $a \in A$, if $\mathcal{D}$ places an IDS to detect it, we will say that $\mathcal{D}$ *covers* it. Otherwise, we say that a is left *uncovered*. Since the defender can allocate only IDS resources to cover k elements in A, the remaining $n-k$ attacks will remain uncovered at any point in time. We will now decompose the reward structure of this game and define four types of utility values corresponding to each attack $a \in A$.

$$\langle U^{\mathcal{D}}_{c,a}, U^{\mathcal{D}}_{u,a}, U^{\mathcal{A}}_{c,a}, U^{\mathcal{A}}_{u,a} \rangle$$

where $U^{\mathcal{D}}_{c,a}$ and $U^{\mathcal{D}}_{u,a}$ denotes the utilities that a defender gets for covering or not covering an attack a respectively. Similarly, $U^{\mathcal{A}}_{c,a}$ and $U^{\mathcal{A}}_{u,a}$ represent the utility an attacker gets when they use an attack a that is covered (and thus gets detected) or not covered (and thus avoids detection) respectively. The values for these symbols are obtained by leveraging the knowledge of security experts as encoded in the Common Vulnerabilities Scoring System (CVSS) [28] and the realistic costs of deploying IDSs. For each attack a_i in the set of attack actions A, we will represent these scores as IS_{a_i}, ES_{a_i} and BS_{a_i} using CVSS metrics, previously discussed in Sect. 3.

Cost of Deploying IDS. We denote the cost of deploying an IDS corresponding to an attack $a \in A$ as $\hat{c}_a$. For our example, we assume the cost of deploying an IDS (shown in the IOC column of Table 1) to be proportional to the betweenness centrality of the VMs on which the IDS is deployed because a VM with high betweenness centrality will affect the latency of routing packets or the latency of processing a request. Also, the centrality values are normalized in the interval $[0, 10]$ to be comparable to the CVSS metrics IS_a, ES_a and BS_a as discussed in Sect. 3. Note that the model in this paper allows another user to define $\hat{c}_a$ in a different way.

We now leverage these defined metrics to design the following rewards for the four utilities associated with each attack a present in our system,

$$U^{\mathcal{D}}_{c,a} = -1 * \hat{c}_a \;,\; U^{\mathcal{D}}_{u,a} = -1 * IS_a$$
$$U^{\mathcal{A}}_{c,a} = -1 * ES_a \;,\; U^{\mathcal{A}}_{u,a} = +1 * BS_a$$

We now provide the rationale for modeling the rewards in this particular manner. The value of $U^{\mathcal{D}}_{c,a}$ is negative since even if it detected an attack, it incurred a cost in order to detect it and moreover there is no extra positive reward given to $\mathcal{D}$ for protecting their system, which is supposed to be the primary functionality. When $\mathcal{D}$ does not place an IDS for detecting the attack a, it incurs a negative utility ($U^{\mathcal{D}}_{u,a}$) equal to IS_a if the attacker uses attack a.

For the attacker $\mathcal{A}$, if it chooses an attack action a which the defender covers (i.e. can detect), it gets a negative utility $U^{\mathcal{A}}_{c,a}$ proportional to the time and cost it had to invest in doing it, which is (somewhat) measured by ES. Also, as $\mathcal{A}$ gains nothing by doing this attack (since the defender can deploy a countermeasure on detection [8]), no positive value is added to it. Lastly, when the attacker uses an attack for which the defender has not placed an IDS, we give a positive utility that (conceptually) adds the IS and subtracts the cost (ES) of performing the attack. Since BS already captures this trade-off, we use it directly.

4 Computing the Stackelberg Equilibrium

We need to solve for the Stackelberg Equilibrium of our game to obtain probability values for each configuration mentioned in A_k, where $A_k \subset A$ such that $|A_k| = k$. Unfortunately, since there are $\binom{n}{k}$ such probabilities (corresponding to each element in A_k), solving for all these variables at once will not yield an efficient solution. Instead, we will solve for the probabilities p_a which represents the probability that a certain attack $a \in A$ is covered by an IDS in a round.

To that extent, we first describe a method that can help in generating the marginal strategies for the defender by solving n ($= |A|$) Linear Programs. Note that the solution can be found in polynomial time in our case because of the particular reward structure our game has. Then, we shall propose an efficient Mixed Integer Quadratic Program (MIQP) method based on this method that helps us to obtain the same marginal strategy, but by solving just one optimization problem. We show that although this formulation, in the general case, is known to computationally hard to solve, in our case, by efficient use of the branch-and-cut mechanism, we can solve it in polynomial time.

4.1 Multiple LP Method

Let T denote the set of k *tokens* that the defender $\mathcal{D}$ can allocate to cover k of the n attacks. Allocating a token to an attack a means that $\mathcal{D}$ has placed the IDS that can detect the particular attack. Now, let the variables p_a represent the probability with which an attack a is covered by one of the k tokens and $p_{a,t}$ represent the probability with which a particular attack a is covered by a particular token $t \in T$. Having defined the probabilities p_a, the defender's expected utility for deploying an IDS to detect a particular attack a^* should be $U^{\mathcal{D}}_{u,a^*} * (1 - p_{a^*}) + U^{\mathcal{D}}_{c,a^*} * p_{a^*}$ [24,25]. Note that, for our scenario, this does not capture the cost $\mathcal{D}$ incurs in deploying the other $k - 1$ IDS mechanisms. Thus, we modify the defender's utility to $U^{\mathcal{D}}_{u,a^*} * (1 - p_{a^*}) + \frac{1}{k}\sum_{a \in A} U^{\mathcal{D}}_{c,a} * p_a$, where the second term denotes the average cost for a particular deployment configuration.

On the other hand, we can simply define the attacker's expected utility for using a particular attack a as $U^{\mathcal{A}}_{c,a} * p_a + U^{\mathcal{A}}_{u,a} * (1 - p_a)$. We now present the optimization problem that maximizes the defender's objective function and the attacker's utility given that an attacker chooses to use the attack a^*.

$$\begin{aligned} \max \quad & \alpha \cdot \frac{1}{k} \sum_{a \in A} U^{\mathcal{D}}_{c,a} p_a + (1-\alpha) \cdot U^{\mathcal{D}}_{u,a^*} (1 - p_{a^*}) \qquad (1) \\ & s.t. \quad p_a \in [0,1] \;\; \forall\, a \in A \\ & \qquad\; p_{t,a} \in [0,1] \;\; \forall\, a \in A, t \in T \\ & \sum_{a \in A} p_{t,a} = 1 \;\; \forall\, t \in T \\ & \sum_{t \in T} p_{t,a} = p_a \;\; \forall\, a \in A \\ & U^{\mathcal{A}}_{c,a} p_a + U^{\mathcal{A}}_{u,a} (1 - p_a) \le U^{\mathcal{A}}_{c,a^*} p_{a^*} + U^{\mathcal{A}}_{u,a^*} (1 - p_{a^*}) \end{aligned}$$

where α is an input parameter that allows the defender to a trade the performance of the system with respect to the security of the system (and vice versa). In the extreme case when $\alpha = 0$, the defender optimizes only for security and completely ignores the fact that deploying k IDSs might affect the performance of the system. In this case, as shown in Sect. 6, $\mathcal{D}$ ends up randomizing more between the deployment configurations of the system. On the other hand, when $\alpha = 1$, the defender optimizes for performance, hardly placing an IDS on systems that affect performance even when it is detrimental to security. We discuss the effects of selecting various α-s in Sect. 6.

Before we dive into what the constraints mean, note that this is a Linear Program (LP) and thus, can be solved in polynomial time. The first two sets of constraints ensure that the optimization variables p_a and $p_{t,a}$ are valid probabilities. The third set of constraints ensures that all the tokens are utilized in covering the different attacks in A. The equality of this constraint is possible in our case since (1) all our tokens are homogeneous, i.e. any token $t \in T$ can be used to cover any attack $a \in A$ and (2) the number of tokens k $(= |T|)$ is less than the number of attacks n $(= |A|)$. Thus, we prune away solutions that do not fully utilize all the tokens. The fourth set of constraints ensure that the probabilities of allocating various tokens to cover an attack a add up to the probability that a is covered. The final set of constraints ensure that the attacker selecting a^* maximizes their utility. Lastly, note that given the values of $p_{t,a}$ one can easily obtain p_a using the fourth set of constraints.

To obtain the (globally) optimal solution (and thus find the optimal marginal strategy) for the defender, we can iterate over all the n attack choices made by the attacker and pick the solution that maximizes $\mathcal{D}$'s utility. Note that, here we enforce the attacker to select a pure strategy as opposed to a mixed strategy. This is not a limitation since for any mixed strategy the attacker can pick in this Stackelberg Game, there always exists a pure strategy in support of it [30].

As the number of VMs and vulnerabilities, i.e., n, increase, this solution method needs to solve a large number of LPs. Thus, we now propose an efficient

MIQP that finds the solution in one go and provides an efficient branch-and-cut algorithm for solving it in polynomial time.

4.2 Compiling Multiple LPs into an Efficient Mixed Integer Quadratic Program (MIQP)

Now, we first introduce n binary switch variables, one for each attack $a \in A$ and represent it as w_a. When the attacker exploits vulnerability a (i.e. uses the attack action a), $w_a = 1$. Otherwise, $w_a = 0$. We now propose the following optimization problem,

$$\max \quad \alpha \cdot \frac{1}{k} \sum_{a \in A} U^{\mathcal{D}}_{c,a} p_a + (1-\alpha) \cdot w_a * U^{\mathcal{D}}_{u,a}(1-p_a) \tag{2}$$

$$\begin{aligned}
s.t. \quad w_a &\in \{0,1\} \;\; \forall\, a \in A \\
p_a &\in [0,1] \;\; \forall\, a \in A \\
p_{t,a} &\in [0,1] \;\; \forall\, a \in A, t \in T \\
\sum_{a \in A} w_a &= 1 \\
\sum_{a \in A} p_{t,a} &= 1 \;\; \forall\, t \in T \\
\sum_{t \in T} p_{t,a} &= p_a \;\; \forall\, a \in A
\end{aligned}$$

$$0 \le v_a - (U^{\mathcal{A}}_{c,a} p_a + U^{\mathcal{A}}_{u,a}(1-p_a)) \le (1-w_a) * M \quad \forall\, a \in A$$

where M represents a large number with respect to the maximum reward the attacker can get, i.e. $M \gg 10$, and v_a is the utility value of the attacker at equilibrium. The first constraint ensures that the switch variables are binary. The fourth constraint enforces the attacker to select a pure strategy since the switch variable corresponding to only one attack can be turned on in a feasible solution. As mentioned in the previous section, this is not a limiting assumption. Lastly, the final set of constraints encodes the complementary slackness condition of the attacker's utility maximization problem [30].

As the defender plays first, it can reason about the attacker picking each attack and select the strategy which gives $\mathcal{D}$ the maximum reward. If the attacker responds to the defender's strategy with attack a^*, then $w_{a^*} = 1$. In that case, the RHS of the last constraint (with a^*) becomes zero and along with the LHS, equality holds. Thus, v_{a^*} is $\mathcal{A}$'s utility value. For all the other attacks $a(\neq a^*)$ that were not selected by $\mathcal{A}$, both the inequalities can be trivially satisfied (as M is a large number) by selecting an appropriate value for v_a.

Theorem 1. *MIQP defined in Eq. 2 produces the same solution as the set of LPs described in Eq. 1.*

Proof. Let us say that when attacker selects an attack a_1, the defender gets the highest utility as per Eq. 1. Now, let us say that Eq. 2 decides that the defender's

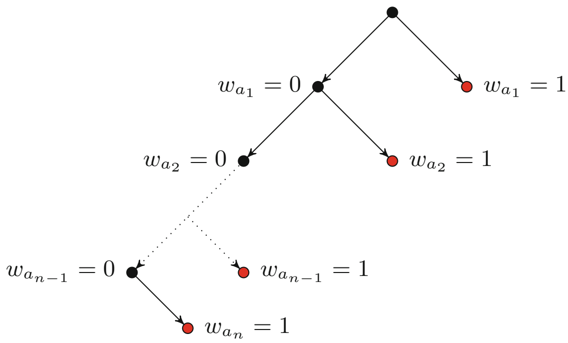
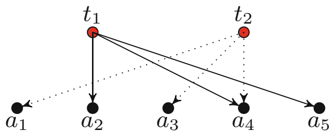


Fig. 2. Branch-and-cut tree for the proposed MIQP. (Color figure online)

Table 2. Player utilities for each vulnerability depending on whether (or not) an IDS is deployed to detect the attacks that exploit it.

Attack	a_1	a_2	a_3	a_4	a_5
$U^{\mathcal{D}}_{c,a}$	-5.7	-10.0	0.0	0.0	0.0
$U^{\mathcal{D}}_{u,a}$	-6.4	-6.4	-2.9	-6.4	-2.9
$U^{\mathcal{A}}_{c,a}$	-8.6	-10	-8.6	-10	-10
$U^{\mathcal{A}}_{u,a}$	6.8	7.5	4.3	7.5	5.0

Table 3. Probability of allocating a token (in order to deploy the corresponding IDS) for detecting each attack.



	a_1	a_2	a_3	a_4	a_5
t_1	0	0.44	0	0.22	0.34
t_2	0.45	0	0.34	0.21	0

utility is strictly better when attacker selects any another attack $a_2 (\neq a_1)$, and thus, $w_{a_2} = 1$. Notice that if this is true, then the objective function value of LP when $a^* = a_2$ is strictly greater than the objective function value of the LP with $a^* = a_1$. But that is a contradiction. Hence, the MIQP defined in Eq. 2 must select a_1 for the attacker.

Similarly, we can prove the other way–that a solution that is optimal for the MIQP (Eq. 2) is also optimal for the LP case. ■

Theorem 2. *MIQP defined in Eq. 2 can be solved in polynomial time with the branch-and-cut method.*

Proof. To prove this, we first represent the branch-and-cut tree for our MIQP in Fig. 2. In that, notice that the right children (shown in red) correspond to an LP problem (similar to the one defined in Eq. 1) where only a particular attack a_i is selected ($w_{a_i} = 1$) and other attacks are not used by the attacker. Since no children of any right child (red node) can generate another solution, the search tree below them can be pruned away. Now, the tree can have at most $n - 1$ left children which correspond to at most n right children, which in turn corresponds to at most n LP problems that need to be solved. Since each LP can be solved in polynomial time and we will solve no more than n LPs, this MIQP can be solved in polynomial time. ■

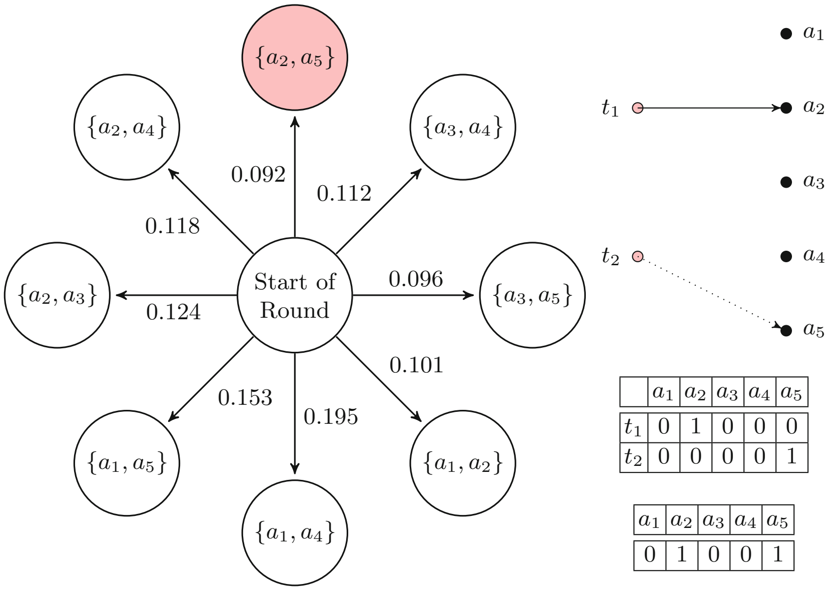


Fig. 3. Optimal mixed strategy of the defender for our scenario (when $\alpha = 0.1$). The probability values for picking up one of the eight IDS placements at the start of each round are written on the edges. For the strategy $\{a_2, a_5\}$ (colored in Pink), the allocation matrix is shown on the right. (Color figure online)

4.3 Obtaining Implementable Strategies

Although we have obtained the values p_a and $p_{t,a}$, there are no guarantees that we will be able to convert these marginal probabilities into $\binom{n}{k}$ probability values that correspond to a defender's deployment strategies, i.e. one that can be implemented in practice. In order to convert these into *implementable strategies*, we use the general version of the Birkhoff Von-Neumann Theorem as stated in [25]. We state this here for completeness.

Birkhoff Von-Neumann Theorem. *Consider an $k \times n$ matrix P with real numbers $p_{t,a} \in [0, 1]$, such that for each $1 \leq t \leq k$, $\sum_{a=1}^{n} p_{t,a} \leq 1$, and for each $1 \leq a \leq n$, $\sum_{t=1}^{k} p_{t,a} \leq 1$. Then, there exist matrices $P^1, P^2, \ldots, P^q$ and weights $w^1, w^2, \ldots, w^q \in (0, 1]$, such that (1) $\sum_{x=1}^{q} w^x = 1$; (2) $\sum_{x=1}^{q} w^x P^x = M$; (3) for each $1 \leq x \leq q$, the elements of M^x are $p^x_{t,a} \in \{0, 1\}$ and (4) for each $1 \leq x \leq q$, we have for each $1 \leq t \leq k$, $\sum_{a=1}^{n} p^x_{t,a} \leq 1$ and for each $1 \leq a \leq n$, $\sum_{t=1}^{k} p^x_{t,a} \leq 1$.*

This theorem guarantees that given the probability matrix $p_{t,a}$, we can always obtain the probabilities of the $\binom{n}{k}$ implementable strategies. The third and fourth equalities in the optimization problem in 1 ensure that the constraint structure imposed on P is a *bi-hierarchy*, which authors in [4] show as a sufficient condition for any marginal probability matrix P to be *implementable*.

Input: Utility Matrix
Output: a^*
Result: Finds and outputs the most critical vulnerability that results in the highest defender utility when fixed

```
max_def_util ← −∞;
a* ← None;
while a ∈ A do
    A′ ← A \ a;
    obj_val, ← solve MIQP (2) with action set A′;
    if obj_val > max_def_util then
        max_def_util ← obj_val;
        a* ← a;
    end
    return a*
end
```

Algorithm 1. Algorithm to find the most critical vulnerability in the Defender's system, which when fixed results in the highest utility.

For our example, assuming that the cost associated with deploying each IDS on a certain VM is a function of the latency it creates. Furthermore, since VMs that are responsible for communication between other VMs would impact the latency the most when an IDS is placed on it. Thus, we assume time impact on the overall latency of the system is equal to the normalized and scaled betweenness centrality of the nodes in our network ($\in [0, 10]$). With that, the utility values for the attacker and defender are shown in Table 2. We first use these values to solve for the optimal marginal strategy (shown in Fig. 3) using the MIQP described in 2. We then use Theorem 1 to obtain the mixed strategies that the defender can actually use to deploy the IDS systems (shown in Fig. 3).

5 Identifying the Most Critical Vulnerability

In real-world scenarios, system administrators, who have a list of known vulnerabilities it should address, have limited developer resources to fix all of the known CVEs in their system at once. Thus, the question of which vulnerability they should fix in order to improve the security of the system is a critical one. In our case, since (1) the rewards of the formulated game are not zero-sum and (2) the defender wants to balance a multi-objective function (that tries to balance the security and usability metrics), figuring out the (critical) vulnerability that $\mathcal{D}$ needs to fix become even more difficult.

Given that we can find the utilities for the defender using Eq. 2, we can ask the question *which attack a* when removed would produce the maximum utility for $\mathcal{D}$. A simple algorithm would be to iterate over all the attacks, removing them one by one, reformulating the MIQP and selecting the attack that maximizes the defender's utility when removed. We describe this idea formally in Algorithm 1 and use it to find the most critical vulnerability of our system. The utilities obtained by removing one vulnerability at a time are shown below (for $\alpha = 0.1$).

$$\langle a_1 : -1.90; a_2 : -1.70; a_3 : -2.30; a_4 : -2.23; a_5 : -2.27 \rangle$$

Thus, in our system, a_2 is the most critical vulnerability since fixing a_2 will result in the highest (gain in) defender's utility.

6 Experiments

We present the results of two different experiments – (1) comparison of our placement strategy (Fig. 3) with existing approaches, and (2) implementation of the Stackelberg Game Strategy (SGS) on a large cloud network instance.

6.1 Comparison with Existing Strategies

In this section, we compare our approach to three other MTD strategies in the context of our running example where $n = 5$ and $k = 2$:

(1) Deterministic Pure Strategy (DPS). This strategy selects a single pure strategy out of the $\binom{5}{2}$ placement strategies. As per work by [20], for DPS, we place IDS to detect a_1 and a_2 (since G1 and G2 are the most critical VMs), which are on the critical paths for any attack flow. Note that, in the context of a stealthy attacker who can exploit any vulnerability in the system, the definition of a critical node, on which an IDS can be deployed, is not clear. Thus, DPS has an inherent disadvantage when compared to MTD strategies, which we now describe.

	a_1	a_2	a_3	a_4	a_5
URS	0.4	0.4	0.4	0.4	0.4
DPS	1	1	0	0	0
CBS	0.52	0.73	0.25	0.25	0.25

Fig. 4. Table showcasing the marginal probabilities with which IDS is places on a node for the different strategies.

(2) Uniform Random Strategy (URS). In this case, we select each of the $\binom{5}{2}$ placements or pure strategies with an equal probability of 0.1. In this case, each attack a is covered in four (out of the ten) pure strategies since having placed an IDS (or token which denotes an IDS was placed) for a, there are $\binom{4}{1} = 4$ ways of placing the other token. Thus, the marginal probabilities are $0.1 * 4 = 0.4$.

(3) Centrality Based Strategy (CBS). This strategy, motivated in the work by [42], has previously been shown to be effective for detecting stealthy bot-nets when PageRank is used as a centrality measure. Since our network is an undirected graph, we use the betweenness centrality measure for evaluation. Since only two of our nodes (G1 and G2) have non-zero values for betweenness centrality, we switch between seven of the ten configurations – three in which only a_1 is covered, three in which only a_2 is covered and one in which both a_1 and a_2 are covered. Since $G1$, on which a_1 is present has a lower centrality value in comparison to $G2$, on which a_2 is present, the first three configurations are less likely than the next three. The last configuration, in which both a_1 and a_2 are covered, is the most likely configuration. The marginal probabilities for covering each attack in the system, as per this strategy, is shown in Fig. 4.

Effectiveness of Our Approach. We plot the defender's utility value for our approach and compare it to all the other approaches. The results are shown in Fig. 5. When adversaries are strategic, i.e. can reason about defender strategies and act rationally to maximize their utility, our method clearly dominates the other methods (see the plots for CBS(min), URS(min) and DPS).

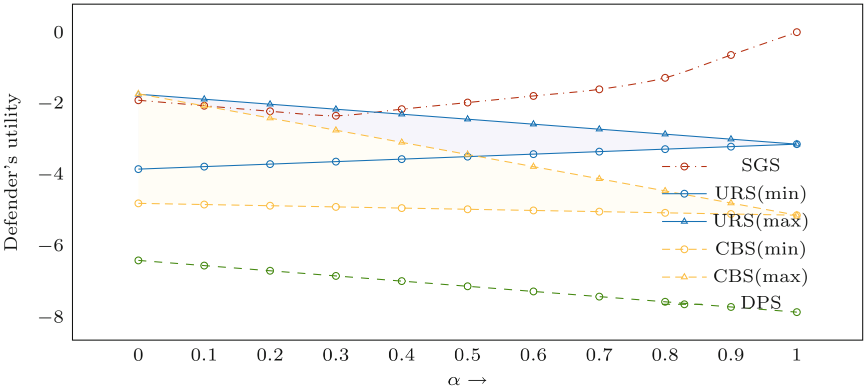


Fig. 5. Defender's utility for the various MTD strategies as the security-usability trade-off value (α) varies from zero to one.

On the other hand, if the attacker is irrational, i.e., selects attacks that do not maximize their profit, Stackelberg Equilibrium may not always be the best strategy. We plot the best case for the other MTD strategies (see URS(max) and CBS(max)) and it turns out that only URS is a little better when $\alpha \in (0, 0.37]$. In this range, our algorithm selects nodes with high centrality measure to improve security in the case of a strategic attacker. This increases the deployment cost and reduces the multi-objective function value, letting URS dominate. CBS on the other hand with no information about the known attacks or performance costs, switches only among the useless and performance expensive configurations, being strictly dominated by SGS. Note that none of the mechanisms we compare against adapt to the security and performance trade-off that is important to the defender. Thus, as the value of α changes, the marginal probabilities for selecting nodes using CBS, URS or DPS remain constant, resulting in straight line plots. On the other hand, SGS, our intelligent switching mechanism, solves the multi-objective optimization when coming up with its mixed strategy.

When α is low (i.e. $\in [0, 0.29]$), our method switches among eight out of the ten pure strategies. As α increases further and the costs start to matter, it places IDS systems more on nodes that impact performance the least. Beyond a certain value (when $\alpha > 0.76$) it realizes that the cost of placing IDS on G1 and G2 (for detecting $a1$ and $a2$) are extremely high on the performance of the system and sticks to only (three) strategies where neither G1 nor G2 is covered.

6.2 Testing on a Large Cloud Network

The setup comprised of 15 VMs and 42 CVEs distributed uniformly on a flat network 10.0.0.0/24, as shown in the Fig. 6. In this experiment, we will measure the throughput for the server (10.0.0.15) hosting an ssh application on port 5002 as the number of IDS systems placed increases. We now describe the different NIDS and HIDS agents pre-configured on the system with the known attack signatures to detect the intrusion attempts.

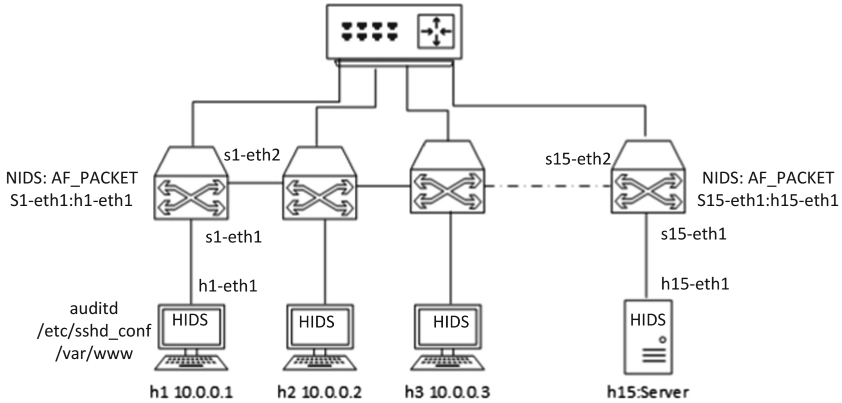


Fig. 6. Testing bandwidth on a flat network with 15 VMs and multiple Network and Host Intrusion Detection Systems (NIDS and HIDS).

Network-Based IDS. Snort [32] was configured to run in IDS (intrusion-detection) as well as IPS (intrusion-prevention) mode. For instance, the attack signature below checks the payload for shellcode targeting remote buffer overflow vulnerability on ssh service running on port 5022.

```
alert TCP any any -> 10.0.0.15 5002 (msg:"EXPLOIT ssh remote
    overflow"; content:"/bin/sh"; reference:Bugtraq,2347;
   reference:cve,2008-5161; sid:1324; rev:6;)
```

The AF packet, which is an IPS configuration, creates a bridge between inspected interfaces (e.g., h1-eth1:s1-eth1). This leads to increased packet processing latency since each packet on a particular bridge is inspected against all traffic patterns which are part of signatures.

Host-Based IDS. Auditd [18] was configured to monitor file integrity of configuration files such as /etc/sshd_conf and binary files for vulnerable services present on the network. A daemon was configured on each inspected host to generate an alert if there is a change in the hash value of inspected files.

The goal of this experiment was to measure the impact of the HIDS/NIDS deployment on the throughput of the service being accessed by normal users.

We show that as $\mathcal{D}$ places more IDSs (1 to 15), we observe a substantial drop in the throughput of the system from 18 Gbps to 6 Gbps (see Fig. 7). This shows that deployment of IDS without considering the impact on network latency can affect the Quality of Service (QoS) for legitimate users in a cloud network.

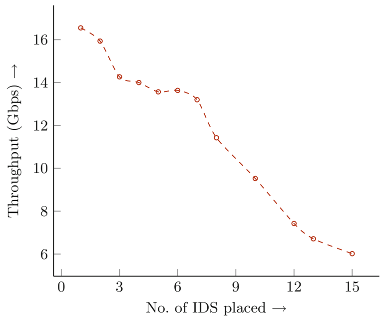


Fig. 7. Change in throughput of the flat network as the number of NIDS and HIDS deployed increases.

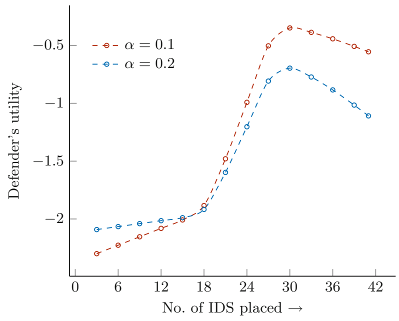


Fig. 8. Change in defender's utility value as the number of NIDS and HIDS deployed increases.

In Fig. 8, we vary the number of IDS systems placed in the system and see how the defender utilities vary. Initially, as the number of IDS increases from 2 to 17, the defender's utility increases at a slow rate since there are too few IDS systems to detect attacks on all the 42 vulnerabilities. As the number of IDS systems are increased beyond 18, the defender's utility starts to increase substantially in each step. At this point, if the attacker does not pick their attack strategically, it is detected with high probability. However, the placement of more IDSs beyond a certain point (30) as shown in the Fig. 8, results in a substantial decrease in throughput, outweighing the benefits of security provided by IDS. Lastly, the most critical vulnerability found in this system was CVE-2013-2207.

7 Conclusion and Future Work

In this paper, we addressed the problem of placing a fixed number of IDS systems in a large cloud environment by proposing a Moving Target Defense (MTD) approach for shifting the detection surface. We formulated this problem as a two-player general-sum Stackelberg Game between the cloud administrator (our defender) and an attacker. We then designed two scalable algorithms that can (1) find the Stackelberg Equilibrium of the formulated game, which lets the cloud service provided a balance between the security and usability of their system, and (2) find the most critical vulnerability in their system. We assumed that the attacker is rational, i.e. he will try to exploit the known vulnerabilities present of VMs in the cloud network by scanning the network. Also, a sophisticated attacker may perform reconnaissance over an extended period of time and use zero-day attacks that cannot be detected by IDS [38]. We plan to model these types of attackers in the future.

Acknowledgements. This research is supported in part by the AFOSR grant FA9550-18-1-0067, ONR grants N00014-16-1-2892, N00014-18-1-2442, N00014-18-12840, the NASA grant NNX17AD06G, the NRL N00173-15-G017, NSF Grants 1642031, 1528099, and 1723440, and NSFC Grants 61628201 and 61571375. The first author is also supported in part by the IBM Ph.D. Fellowship 2018-19.

References

1. Al-Jarrah, O., Arafat, A.: Network intrusion detection system using neural network classification of attack behavior. J. Adv. Inf. Technol. **6**(1), 1–6 (2015)
2. Bakshi, A., Dujodwala, Y.B.: Securing cloud from DDOS attacks using intrusion detection system in virtual machine. In: Second International Conference on Communication Software and Networks, ICCSN 2010, pp. 260–264. IEEE (2010)
3. Brown, D.J., Suckow, B., Wang, T.: A survey of intrusion detection systems. Department of Computer Science, University of California, San Diego (2002)
4. Budish, E., Che, Y.K., Kojima, F., Milgrom, P.: Designing random allocation mechanisms: theory and applications. Am. Econ. Rev. **103**(2), 585–623 (2013)
5. M. I. Center: APT1: Exposing One of Chinas Cyber Espionage Units. Mandian.com (2013)
6. Chen, D., Zhao, H.: Data security and privacy protection issues in cloud computing. In: 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE), vol. 1, pp. 647–651. IEEE (2012)
7. Chowdhary, A., Alshamrani, A., Huang, D., Liang, H.: MTD analysis and evaluation framework in software defined network (MASON). In: Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization, pp. 43–48. ACM (2018)
8. Chowdhary, A., Pisharody, S., Huang, D.: SDN based scalable MTD solution in cloud network. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense, pp. 27–36. ACM (2016)
9. Chung, C.J., Khatkar, P., Xing, T., Lee, J., Huang, D.: NICE: network intrusion detection and countermeasure selection in virtual network systems. IEEE Trans. Dependable Secure Comput. **10**(4), 198–211 (2013)
10. Conitzer, V., Sandholm, T.: Computing the optimal strategy to commit to. In: Proceedings of the 7th ACM Conference on Electronic Commerce, pp. 82–90. ACM (2006)
11. Dastjerdi, A.V., Bakar, K.A., Tabatabaei, S.G.H.: Distributed intrusion detection in clouds using mobile agents. In: Third International Conference on Advanced Engineering Computing and Applications in Sciences, ADVCOMP 2009, pp. 175–180. IEEE (2009)
12. Debar, H., Dacier, M., Wespi, A.: Towards a taxonomy of intrusion-detection systems. Comput. Netw. **31**(8), 805–822 (1999)
13. Deshpande, P., Sharma, S., Peddoju, S., Junaid, S.: HIDS: a host based intrusion detection system for cloud computing environment. Int. J. Syst. Assur. Eng. Manag. **9**, 1–10 (2014)
14. Durkota, K., Lisỳ, V., Bosanskỳ, B., Kiekintveld, C.: Optimal network security hardening using attack graph games. In: IJCAI, pp. 526–532 (2015)

15. Garfinkel, T., Rosenblum, M., et al.: A virtual machine introspection based architecture for intrusion detection. In: NDSS, vol. 3, pp. 191–206 (2003)
16. Hu, Z., Zhu, M., Liu, P.: Online algorithms for adaptive cyber defense on Bayesian attack graphs (2017)
17. Ibrahim, L.M.: Anomaly network intrusion detection system based on distributed time-delay neural network (DTDNN). J. Eng. Sci. Technol. **5**(4), 457–471 (2010)
18. Ilgun, K.: USTAT: a real-time intrusion detection system for UNIX. University of California Santa Barbara Master thesis (1992)
19. Jajodia, S., Park, N., Serra, E., Subrahmanian, V.: Share: a stackelberg honey-based adversarial reasoning engine. ACM Trans. Internet Technol. (TOIT) **18**(3), 30 (2018)
20. Jha, S., Sheyner, O., Wing, J.: Two formal analyses of attack graphs. In: Proceedings of 15th IEEE Computer Security Foundations Workshop, pp. 49–63. IEEE (2002)
21. Jo, S., Sung, H., Ahn, B.: A comparative study on the performance of intrusion detection using decision tree and artificial neural network models. J. Korea Soc. Digit. Ind. Inf. Manag. **11**(4), 33–45 (2015)
22. Jones, S., et al.: Evaluating moving target defense with PLADD. Technical report, Sandia National Labs-NM, Albuquerque (2015)
23. Kheir, N., Cuppens-Boulahia, N., Cuppens, F., Debar, H.: A service dependency model for cost-sensitive intrusion response. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 626–642. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15497-3_38
24. Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., Tambe, M.: Computing optimal randomized resource allocations for massive security games. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 689–696. International Foundation for Autonomous Agents and Multiagent Systems (2009)
25. Korzhyk, D., Conitzer, V., Parr, R.: Complexity of computing optimal stackelberg strategies in security resource allocation games. In: AAAI (2010)
26. Letchford, J., Vorobeychik, Y.: Optimal interdiction of attack plans. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, pp. 199–206. International Foundation for Autonomous Agents and Multiagent Systems (2013)
27. Maleki, H., Valizadeh, S., Koch, W., Bestavros, A., van Dijk, M.: Markov modeling of moving target defense games. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense, pp. 81–92. ACM (2016)
28. Mell, P., Scarfone, K., Romanosky, S.: Common vulnerability scoring system. IEEE Secur. Priv. **4**(6), 85–89 (2006)
29. Panda, S., Vorobeychik, Y.: Near-optimal interdiction of factored MDPs. In: Conference on Uncertainty in Artificial Intelligence (2017)
30. Paruchuri, P., Pearce, J.P., Marecki, J., Tambe, M., Ordonez, F., Kraus, S.: Playing games for security: an efficient exact algorithm for solving Bayesian stackelberg games. In: Proceedings of the 7th International Joint Conference on Autonomous agents and Multiagent Systems, vol. 2, pp. 895–902. International Foundation for Autonomous Agents and Multiagent Systems (2008)
31. Pita, J., et al.: Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track, pp. 125–132. AAMAS (2008)

32. Roesch, M., et al.: Snort: lightweight intrusion detection for networks. In: LISA 1999, pp. 229–238 (1999)
33. Roschke, S., Cheng, F., Meinel, C.: An extensible and virtualization-compatible ids management architecture. In: Fifth International Conference on Information Assurance and Security, IAS 2009, vol. 2, pp. 130–134. IEEE (2009)
34. Rowland, C.H.: Intrusion detection system. US Patent 6,405,318, 11 June 2002
35. Sengupta, S., Chakraborti, T., Kambhampati, S.: MTDeep: boosting the security of deep neural nets against adversarial attacks with moving target defense. arXiv preprint arXiv:1705.07213 (2017)
36. Sengupta, S., et al.: A game theoretic approach to strategy generation for moving target defense in web applications. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp. 178–186. International Foundation for Autonomous Agents and Multiagent Systems (2017)
37. Sinha, A., Nguyen, T.H., Kar, D., Brown, M., Tambe, M., Jiang, A.X.: From physical security to cybersecurity. J. Cybersecur. **1**(1), 19–35 (2015)
38. Tankard, C.: Advanced persistent threats and how to monitor and deter them. Netw. Secur. **2011**(8), 16–19 (2011)
39. Vadlamudi, S.G., et al.: Moving target defense for web applications using Bayesian stackelberg games. In: Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems, pp. 1377–1378. International Foundation for Autonomous Agents and Multiagent Systems (2016)
40. Van Dijk, M., Juels, A., Oprea, A., Rivest, R.L.: Flipit: The game of "stealthy takeover". J. Cryptol. **26**(4), 655–713 (2013)
41. Venkatesan, S., Albanese, M., Amin, K., Jajodia, S., Wright, M.: A moving target defense approach to mitigate DDOS attacks against proxy-based architectures. In: 2016 IEEE Conference on Communications and Network Security (CNS), pp. 198–206. IEEE (2016)
42. Venkatesan, S., Albanese, M., Cybenko, G., Jajodia, S.: A moving target defense approach to disrupting stealthy botnets. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense, pp. 37–46. ACM (2016)
43. Vieira, K., Schulter, A., Westphall, C., Westphall, C.: Intrusion detection for grid and cloud computing. IT Prof. **12**(4), 38–43 (2010)
44. Zhu, Q., Başar, T.: Game-theoretic approach to feedback-driven multi-stage moving target defense. In: Das, S.K., Nita-Rotaru, C., Kantarcioglu, M. (eds.) GameSec 2013. LNCS, vol. 8252, pp. 246–263. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02786-9_15
45. Zhuang, R., DeLoach, S.A., Ou, X.: Towards a theory of moving target defense. In: Proceedings of the First ACM Workshop on Moving Target Defense, pp. 31–40. ACM (2014)