Moving Target Defense for Web Applications using Bayesian Stackelberg Games

Satya Gautam Vadlamudi, Sailik Sengupta, Subbarao Kambhampati
Yochan Group, School of CIDSE
Arizona State University
{gautam, sailik.sengupta, rao}@asu.edu

Marthony Taguinod, Ziming Zhao, Adam Doupé, Gail-Joon Ahn
SEFCOM Lab, School of CIDSE
Arizona State University
{mtaguino, zmzhao, doupe, gahn}@asu.edu

ABSTRACT

Web applications form a critical component of cyber security systems as they act as a gateway for many institutions. Vulnerabilities in web applications allow malicious actors to access and/or modify restricted data. Here the hackers have the opportunity to perform reconnaissance so as to gain knowledge about the web application layout before launching an attack, whereas the defender (administrator of the web application) must secure the application even with its potential vulnerabilities. In order to mask such vulnerabilities which are primarily associated with different individual configurations, Moving Target Defense systems were proposed wherein the defender switches between various configurations thereby making it difficult to attack with success, while maintaining a seamless experience for the genuine users. However, the design of good quality switching strategies is still an open problem which is crucial for the effectiveness of the Moving Target Defense approach. In this paper, we present a way to find effective switching strategies by modeling this ecosystem as a Bayesian Stackelberg game with the administrator as the leader and the hackers as the followers, which as we show succinctly captures various aspects of the Moving Target Defense systems. Furthermore, we show how to determine which vulnerability areas should be addressed first once the system is deployed and which attacker type uncertainties should be calibrated with high precision, for increasing the security of the web application. We present experimental results on a representative web application system demonstrating the utility of switching strategies obtained using the proposed method, and we discuss various future directions that are unique to the web application domain.

Keywords

Cyber security, Web applications, Moving target defense, Bayesian Stackelberg games

1. INTRODUCTION

A shorter version of this paper appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), May 9–13, 2016, Singapore.*

Web applications are the most widely used means for businesses to provide services over the Internet. Often times, sensitive business and user data is managed and processed by these applications. Consequently, vulnerabilities in web applications pose risks to the security and privacy of both businesses and users. For instance, the JP Morgan Chase breach in 2014 affected 76 million US households [44], where Bloomberg reported that the hackers exploited an overlooked flaw in one of the websites of the bank [41]. Therefore, web application security is of paramount importance to both businesses and consumers alike.

Several techniques and tools based on static analysis (white-box) and dynamic analysis (black-box) have been proposed to discover the vulnerabilities in web applications [1, 14, 20, 9, 10], so that the vulnerabilities can be removed before the attackers discover and exploit them. However such efforts are not enough due to the increasing complexity of modern web applications and the limited development and deployment time [52], whereas the attackers can perform reconnaissance and attack. To address these challenges, a Moving Target Defense (MTD) based approach was proposed in [47] to secure web applications, which complements the aforementioned vulnerability analysis techniques through a defense-in-depth mechanism.

The MTD based approach dynamically configures and shifts systems over time, at different layers of the web application stack, to increase the uncertainty and complexity for the attackers to perform probing and attacking [8, 54], while ensuring that the system is available for legitimate users. Here, the window of attack opportunities decreases and the cost of an attack increases. Further, even if an attacker succeeds in finding a vulnerability at one point, it may not be effective at other times because of the moving defense system, thus making the web application more resilient. In [47], various aspects that can support Moving Target Defense approach are presented such as, using multiple implementation languages, multiple database instances with synchronization, etc. in different layers of web application architecture, along with ways to switch between them. However, the design of good quality switching strategies is left as an open problem—which is key to effectively leverage various move options—thereby maximizing the complexity for the attacker and minimizing the damage for the defender.

In this paper, we focus on the design of effective policies

for movement in the Moving Target Defense system that maximize the resilience of the web application, given the set of components and configurations of the system which can be "moved around." We observe that many of the features of this problem can be captured effectively by modeling it as a Bayesian Stackelberg game. Note that, similar to a Stackelberg game where the leader acts first and the follower(s) can observe and act accordingly, here the web application system is deployed first with an MTD policy and the attacker(s) can perform reconnaissance and attack. Furthermore, there is often uncertainty about the type of attacker(s) who can attack the system, which can be effectively captured by modeling it as a Bayesian game where each of the agents in the game could be of multiple types with respective probabilities. Accordingly, we model the overall problem of Moving Target Defense as a Bayesian Stackelberg game and solve it to obtain effective *movement policies*, which is the primary contribution of this paper. In addition, we propose techniques to find most critical vulnerabilities and most sensitive attacker types, which help in improving the security of web application systems. To the best of our knowledge, this is the first work that maps Moving Target Defense as a Bayesian Stackelberg game.

The problem of finding an optimal policy for the leader to commit to in a Bayesian Stackelberg game is known to be NP-hard [6]. In this paper, as a demonstration, we use the Decomposed Optimal Bayesian Stackelberg Solver (DOBSS) method [35] to find an optimal movement policy for the defender which is shown to be superior to the methods based on Harsanyi transformation [16] and the ones that invoke multiple linear programs [6]. One could adopt any of the existing algorithms of their choice for solving BSGs.

2. RELATED WORK

Moving target defense systems are gaining attention in cyber security for their ability to increase the complexity for the attackers and therefore be more resilient. In [48], it is shown through tool-based (CORE, Nmap) and manual penetration tests that platform diversity and rotation improves the security, and that the likelihood of a successful attack decreases proportionally with the time between rotations. In [53], a model for adaptive attacker strategy based on genetic algorithms against Moving Target Defense systems is presented and evaluated over a defender with diverse strategies. Game theoretic approaches have been used in attack surface shifting in [31] where stochastic extensive form games are used to model the scenario. In [19], the moving target defense approach is evaluated by modeling it as a game called PLADD which they created based on FlipIt [49], where players compete to control a shared resource. However, the above techniques have not considered the reconnaissance aspect of the attackers which is central in case of applications like that of the web application security (In [33], it was observed that reconnaissance is an important attack phase for dynamic networks), demanding for a leader-follower approach for optimizing the rewards. In [4], a game theoretic leader-follower type approach is presented for dynamic platform defenses where the strategies are chosen so that they are diverse, based on statistical analysis, rather than uniformly distributed, and it was shown that such strategies outperform simple randomization strategies. However, they do not consider the uncertainty in the attacker model which is another important aspect for web applications. Both the uncertainty and the reconnaissance aspects can be handled via Bayesian Stackelberg games which we believe are more appropriate for modeling the web applications domain.

In [13], models to evaluate the effectiveness of diversity based moving target techniques are presented. They measure the impact of various attacks such as circumvention, brute force, etc. via probability of success of an attack. In [34], a quantitative evaluation of dynamic platform techniques is presented where their observations include– threat models being a crucial factor in the level of protection that can be provided by a particular defense technique. In [40], the authors perform an empirical game theoretic analysis of moving target defense systems by modeling the objectives, attack costs and the ability of the defender to detect the attack actions, and find that the defender tends to proactively move when the ability to detect is hampered. Methods for strategically placing honey pots for network security are explored in [23] using normal form games with mixed strategies [32], stackelberg games [6], and deception games [46]. An overview of the security games which could be used in cyber security is presented in [45].

Over the years, several applications have been mapped to Bayesian Stackelberg games, such as, patrolling in ports [42], airports [39, 36] and transit systems [30], robotic patrolling [3], malicious packet detection in computer networks [50], and so on. Many algorithms have been proposed to efficiently solve the problems of Stackelberg security games based on multiple linear programs, MIQP, MILP, and branch-and-price [6, 17, 22, 35]. A technique based on hierarchical decomposition and branch-and-bound was proposed in [18] along with faster approximate versions, which was shown to be superior to the above methods. Variations of the security games involving uncertainty in reward values and uncertainty in different types of attackers have also been pursued [24, 28, 37].

3. WEB APPLICATION DOMAIN AND MOVING TARGET DEFENSE

Before presenting the cyber security problem with web applications, we first present a brief overview of the web application domain and its functionality which is useful for understanding the challenges involved and solution strategies. Figure 1 shows a distributed web application structure with components belonging to the server-side and the client-side [47].

The server-side typically includes the following layers from top to bottom:

- The logic layer which implements the application business logic using high-level programming languages such as Java, PHP, or Python.
- The web server layer which receives HTTP requests from clients, parses them, and passes the request to the appropriate server-side program. Examples of web servers include Apache web server, Windows IIS, and Nginx.
- The data storage layer that stores the web application state and user data. Examples of data storage systems are SQL databases such as MySQL, PostgreSQL, and MSSQL.
- The operating system layer that provides the runtime environment for the web server layer and database storage layer.
- The infrastructure layer that runs the operating systems.

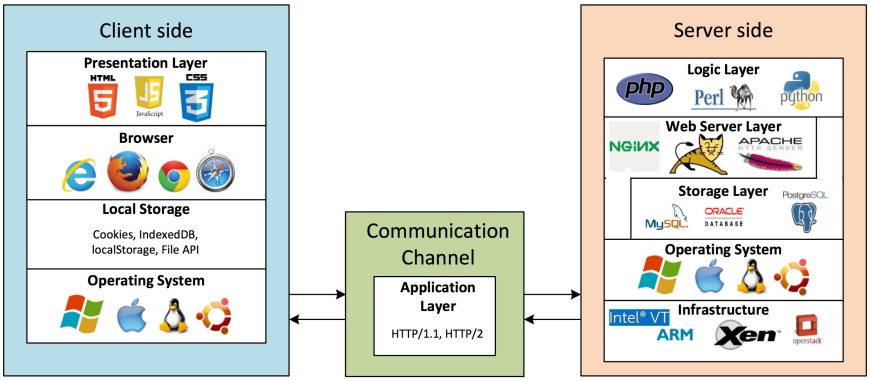

Figure 1: A modern web application structure and its components.

This may be either a physical machine or a virtualization platform that manages multiple virtual machines.

The communication channels between the client and the server are typically based on the HTTP protocol and its derivatives such as HTTPS, SPDY, and HTTP/2.

The client-side is responsible for converting the HTTP responses from the server into actionable graphical interface for the user, and has the following layers:

- The logic layer or presentation layer which is written using HTML, CSS, and JavaScript, the latter of which provides a way for the server to execute application logic on the client.
- The browser which retrieves the presentation layer code from the server, interprets it, and presents it as a graphical user interface to the user.
- The storage layer which is used by the presentation layer to store data. Examples include cookies, localStorage, IndexedDB, and File APIs.
- The operating system which the browser runs on.

If any layer in Figure 1 is compromised, all the layers above it become untrustworthy. For instance, if the operating system layer on the server-side is compromised, then the data storage, web server, and server-side logic are all compromised. Further, since the presentation layer (on the client side) is created by the server and is sent across the communication channel, a compromise of the server or the communication channel compromises the presentation layer.

Adversaries can attack a chosen layer in Figure 1 through its interfaces from layers above. For instance, in a drive-by download attack on the client browser layer [7], an attacker uses JavaScript in the presentation layer to coerce the browser and its extensions to download additional malware that controls the victims' computers. In this case, the attacker uses the presentation layer to exploit a vulnerability in the browser layer that leads to arbitrary code execution in the browser address space. Such injected arbitrary code can in turn exploit a vulnerability in the client operating system to escalate its privilege and further infect the client machine. Furthermore, the malicious JavaScript code might be delivered by an attacker exploiting a vulnerability in the server-side logic layer, using a reflected or stored cross-site scripting (XSS) vulnerability.

In this scenario, Moving Target Defense (MTD) was proposed to make the web application structure more resilient to attacks [47]. The basic idea of moving target defense is to periodically shift and change system configurations over time to increase complexity and cost for the attackers. MTD does not remove vulnerabilities directly but limits the exposure of vulnerabilities, so that the opportunities for the attacks can be decreased. The effectiveness of an MTD approach depends on how many components can be moved and how they are moved. Instruction set randomization [21, 2] to disrupt binary code injection attacks is an instance of MTD. An MTD can be either static (movement happens only at/before the launch of a program) or dynamic (movement may be allowed while the program is live), the latter being more flexible but more difficult to implement.

For movement in layers specific to web applications, i.e., the logic layer, the storage layer, the presentation layer and the browser, the following methods were proposed [47]: In the logic layer, implementation languages of web applications can be switched using translators, which can improve resilience w.r.t. language specific and framework specific vulnerabilities. In the storage layer, running and synchronizing multiple database instances having different implementations can be helpful. In the presentation layer, adding tags to HTML fields to hide real values and introducing randomness into JavaScript code by mutating tokens can increase complexity for the attackers. In the browser, moving and changing rendering engines, JavaScript interpreters, and XML parsers can improve the resilience. For movement in other layers, such as the operating system layer and the infrastructure layer, the relevant techniques can be found in [51, 5, 29]. A movement can happen from any valid combination of configurations in various layers to any other valid combination, where a combination of configurations is valid if the functionality of the system for the regular users of the application is not affected.

Now that we know what movements are possible, in the following, we will present how optimal strategies for movement can be obtained.

4. MAPPING OF MOVING TARGET DEFENSE AS BAYESIAN STACKELBERG GAME

Now we present the methodology for obtaining optimal strategies for movement in the MTD framework. We observe that the scenario of the defender (administrator of the web application) and the attacker is very similar to that of the leader and the follower in Bayesian Stackelberg Games. In both the cases, the defender chooses a certain strategy which can observed by the attacker before making an at-

Table 1: A Stackelberg game with leader as the row player and follower as the column player. Values at each position indicate the rewards for the leader and the follower respectively.

	F_1	F_2
L_1	2,0	4,2
L_2	0,1	5,0

tack. Therefore the defender must find and follow such a strategy which results in minimal damage assuming that the attackers know that strategy.

4.1 Bayesian Stackelberg Games

Before presenting the exact mapping details of Moving Target Defense as Bayesian Stackelberg game (BSG), we briefly describe BSGs. A Stackelberg game consists of a leader who commits to a strategy first, and then a follower adopts a strategy which maximizes its own reward based on the leader's strategy. Here the leader has an advantage because she gets to make the first move (choose a strategy and commit), which is illustrated with a simple game as shown in Table 1. In this game consisting of one leader and one follower, leader is the row player with strategies L_1 & L_2 available to it, and follower is the column player with strategies F_1 & F_2 available to it. Note that, the leader can choose L_1 to commit to since it is the one that makes the first move, and then the follower will be forced to choose F_2 to commit to, resulting in 4 reward points for the leader. Further, if the leader commits to a uniform mixed strategy [6, 36] of choosing L_1 and L_2 with 0.5 probability each, then the follower would choose F_2 to maximize its rewards and the payoff for leader becomes 4.5. In general, mixed strategies are at-least as good as pure strategies (since all pure strategies are also part of the mixed strategies) and many a time better.

A Bayesian game contains a set of n agents where each agent i could be of any of the T_i types. For the Bayesian Stackelberg games in this paper, we assume that there are two agents, the leader and the follower, where the leader has a single type ($|T_1| = 1$, assuming it is the first agent) and the attacker may have multiple types. Each agent i has a set of strategies available to it S_i and a reward/utility function $R_i : T_2 \times S_1 \times S_2 \rightarrow \mathbb{R}$. The objective here is to find an optimal mixed strategy for the leader to commit to (one which maximizes her rewards), given that the follower(s) may know that strategy when choosing their strategy. This solution concept is called as Strong Stackelberg Equilibrium [25, 26, 27].

4.2 Mapping of Moving Target Defense

Now we present the details of how the Moving Target Defense could be mapped as a Bayesian Stackelberg game to determine an optimal movement policy. The defender in MTD can be mapped to the leader in BSG, and the attacker-to the follower. Whereas the defender (web application) is of a single type, clearly the attacker could be of multiple types consistent with our formulation of Bayesian Stackelberg games above.

Next, we list the strategies available for the leader and the follower. Note that, the movements of the defender can be viewed as choosing strategies as follows: Let us assume that the defender has to choose a move with time period τ once the web application is started with a particular configuration. Different moves in Moving Target Defense would lead to different configurations at different layers (we call each set of valid configurations as a 'combination'). So, a move in MTD from one particular combination to another combination could be mapped to a strategy in BSG. Further, the number of moves available to the defender at each point is the same as the total number of valid combinations (of configurations at different layers) since we observe that the web application can be switched from any given combination to any other combination (this may be restricted if needed).

Finally, each of the attack options of the attacker in MTD could be viewed as a strategy for the follower in BSG. Note that, the attack options to be considered here can be either the attacks for which the fixes are not yet deployed in the web application or estimated models of possible future attacks. Each attacker type in MTD has a set of attack options which may or may not be overlapping with other attacker types. These could correspond to different follower types with different set of strategies available for each of them, which is allowed by the BSG formulation.

Whether a particular attacker type has access to a given attack option is dependent on (and hence formulated based on) the following criteria:

- The difficulty of carrying out the action/attack, for instance a database vulnerability is easier to find and exploit in comparison to an arbitrary code execution resulting in remote code execution exploits.
- The popularity of the target technology, as more popular implementations of services result in higher chance of exploits and wider array of available tools.

And, the choice of reward values for the defender and the attacker are guided by the following considerations:

- The effect of the action/attack on the target website. For instance, sometimes, exploits that compromise a web-site's availability are more severe than an exploit allowing access to data history, and vice-versa. Note that, the "effect" here it is dependent on the particular website as well as the particular attacker type, since a very complex and clever attack may sometimes only be able to uncover data which is less critical to the defender (website). Also, it is also dependent on how much an attacker can leverage out of the data acquired. Therefore, the rewards here are based on both the defender (the criticality of various data/functionality) and the attacker type (ability to leverage the success of various attacks).
- Whether the attack leads to detection of the attacker, which depends on the number of exploits they target in each attack. For instance, an attack involving a database injection will have a limited scope of detection in comparison to an attack involving both a database injection and a remote code execution.

It is worth noting again that the reward values for both attacker and defender are dependent on the scenario (the particular website and the criticality of the data/functionality under consideration). For instance, a news aggregate website such as Reddit only stores non-critical data such as poster user-names, passwords, and post history; as such, the reward values for attackers executing database attacks will not be as high compared to remote code execution exploits, in which attackers will always gain maximum benefit, due to allowing access to all other components of the website and

Table 2: Rewards in case of an MTD with the defender as the row player and the attacker as the column player. First entry in each box is the reward value for the defender and the second entry is the reward value for the attacker.

	A1: A MySQL Injection	A2: A PostgreSQL Injection	A3: A Python Remote Code Execution	A4: A PHP Remote Code Execution	A5: A1&A3	A6: A2&A3	A7: A1&A4	A8: A2&A4	A9: No Attack
MySQL& Python	-2,6	5,-8	-10,10	5,-2	-10,10	-5,10	2,6	10,-10	0,0
PostgreSQL& Python	5,-8	-2,6	-10,10	5,-2	-5,10	-10,10	10,-10	2,6	0,0
MySQL& PHP	-2,6	5,-8	5,-2	-10,10	2,6	10,-10	-10,10	-5,10	0,0
PostgreSQL& PHP	5,-8	-2,6	5,-2	-10,10	10,-10	2,6	-5,10	-10,10	0,0

organization. Consequently, defender reward values would not be negatively affected as much, in addition to the chance of being able to detect an attacker.

4.3 Solving Bayesian Stackelberg Games

Given a set of strategies for the leader (whose indices are in the set L), the follower types (whose indices are in the set T), their individual set of strategies (whose indices are in the set F^t, $t \in T$), the reward matrices for the defender and the follower for each follower type: R^t and C^t, the probabilities of each of the follower types p^t s.t. $\sum_{t \in T} p^t = 1$, an optimal mixed policy/strategy $\boldsymbol{x}$ for the defender could be found by solving the following Mixed Integer Quadratic Program (MIQP) that is based on the Decomposed Optimal Bayesian Stackelberg Solver (DOBSS) [35]:

$$
\begin{aligned}
&\max_{x,n,a} \sum_{l \in L} \sum_{t \in T} \sum_{f \in F^t} p^t R^t_{lf} x_l n^t_f \quad \text{s.t.} \\
&\sum_{l \in L} x_l = 1 \\
&\sum_{f \in F^t} n^t_f = 1 \\
&0 \leq (a^t - \sum_{l \in L} C_{lf} x_l) \leq (1 - n^t_f) M \\
&x_l \in [0...1] \\
&n^t_f \in \{0, 1\} \\
&a^t \in \mathbb{R}
\end{aligned}
\tag{1}
$$

where M is a large positive number.

The objective here is to find a mixed strategy $\boldsymbol{x}$ which maximizes the reward for the defender considering various strategies of different types of the follower through n^t_f. The first and fourth constraints show that x_l can take non-negative real values whose sum should be 1. The second and fifth constraints show that the follower can choose one of the pure strategies to attack. Here only the reward-maximizing pure strategies for the follower are considered since for a given fixed mixed strategy $\boldsymbol{x}$ of the leader, each follower type faces a problem with fixed linear rewards. Therefore, there exists an optimal strategy for the follower which is pure (note that, all optimal mixed strategies in this case are just the combinations of different optimal pure strategies). The third and the sixth constraints capture the maximization of rewards for the follower types. Note that, the right inequality in the third constraint is non-existent whenever $n^t_f = 0$ and the left inequality captures the maximization of the reward for the follower, and when $n^t_f = 1$, both the left and right inequalities will result in equality which happens with a strategy that maximizes the reward for the follower.

4.4 A Working Example

Table 3: Different types of the attacker with their capabilities and the associated probabilities of them attacking the defender.

Type Name	Attack Capabilities	Probability
Database Hacker	A1, A2, A9	0.50
Mainstream Hacker	A1,A4,A7, A9	0.35
Nation State	All	0.05
Script Kiddie	A1,A2,A3,A4,A9	0.10

Now, we present an example of the Moving Target Defense system which is mapped as a Bayesian Stackelberg game. Table 2 shows the moves and reward values for the defender and the follower. Here the defender has four moves/strategies to choose from: Platforms that combine MySQL and Python, or PostgreSQL and Python, or PostgreSQL and PHP, or PostgreSQL and PHP. And the attacker types may have access to one or more of the nine attacks/strategies. Note that, the attacks A5–A8 are combinations of the attacks A1–A4. Further, positions where both the reward values are positive indicate that although the attacker was successful in their effort, the effect on the data and service was not substantial whereas the defender got useful information about the attack. For instance, a scenario can occur where none of the attackers actions match the configuration of the defender, therefore no attack will be successful. However, the attacker will still gain information on the defender's configuration that can then be used to build a successful attack. Meanwhile, the defender also benefits by being able to acquire information on the attacker and by the system availability being unaffected due to the attack failing to compromise the data or the system.

Table 3 presents the details of various types of the attacker and the probabilities with which they might attack. There are four types listed here, namely: database hackers who excel at infiltrating the back-end storage component of web-applications that contain sensitive information such as credit card information, social security numbers, etc; mainstream hackers who mainly target popular technologies in order to inflict damage across a larger number of applications implemented using the same technology; nation state hackers who are well funded and professionally trained to be flexible and effective; and script kiddies who, on the other hand, have limited training and knowledge in carrying out effective attacks and instead use existing software/scripts to execute attacks. In general, a nation state hacker is expected to have a much larger superset of actions compared to that of the others, and have higher impact (larger reward values).

The formulation and mapping remain the same allowing the consideration of all such aspects through more detailed and larger tables.

Note that, here the reward values for the defender and the attacker corresponding to each type are constant based purely on the strategies chosen by them. That is, the reward matrices corresponding to different types of the follower are simply selections of appropriate columns for each of the rows in Table 2. As discussed before, this need not be the case in general, and the rewards corresponding to a given attack can vary depending on the attacker type.

5. CRITICALITY AND SENSITIVITY ANALYSIS USING THE BSG MAPPING

In this section, we show how the proposed mapping from web application moving target defense to Bayesian Stackelberg games and the resulting strategies can be analyzed to further increase the overall security. First, we consider the problem of finding most critical vulnerabilities in the web application, which is a much needed feature for defenders, since they usually have to put their limited time and energy to harden the weakest link in their system once the system is deployed (or even before the system is deployed). Second, we aim to find most sensitive attacker types in the sense that a change in their probability will influence the optimal reward for defenders the most, which helps in knowing which probabilities are to be estimated with as high precision as possible.

5.1 Finding Most Critical Vulnerabilities

Once the defender makes the web application system with MTD live for users with the optimal mixed strategy obtained above, her work is only partially complete. Then onwards the defender spends time on finding and fixing various vulnerabilities, and accordingly updating the tables and strategies, so that the security of the web application is kept high to the best of the abilities. In this scenario, it is useful to focus on those vulnerabilities or attacks first which are not effectively masked by the optimal mixed strategy with which the system is deployed. These are the most critical vulnerabilities for the current deployment of the system. In the following we show how to find such vulnerabilities.

Note that, whenever a vulnerability corresponding to an attack(s) is fixed by the defender, the corresponding attack(s) will no longer be part of the input reward tables. Accordingly, new optimal mixed strategies can be computed. Therefore, one can measure the criticality of various vulnerabilities by removing the corresponding attack(s) from the input tables and getting new optimal reward values, and comparing them. Removal of whichever vulnerabilities result in highest new optimal reward value can be termed as the most critical vulnerabilities, as fixing any of them would result in higher rewards for the defender compared to the rest. Therefore, such vulnerabilities should be chosen to address first for improving the security.

Here, if the defender wants to find *one most critical vulnerability*, then it would involve solving n instances of the Bayesian Stackelberg Game, where n is the total number of vulnerabilities. In each instance, attacks corresponding to one of the vulnerabilities are removed from Table 3. Note that, if removing a particular vulnerability leads to an attacker type being left with only the *No Attack* option. Then, that attacker type can be ignored and its probability can be distributed amongst the remaining attacker types proportional to their existing probabilities.

Further, if the defender wants to find *the most critical vulnerability set of size 2*, then it would involve solving ${}^nC_2$ instances of the Bayesian Stackelberg Game where different combinations of vulnerabilities are removed on each occasion. Note that, the most critical vulnerability set of size 2 may not be the same as putting together the most critical vulnerability and the next most critical vulnerability obtained when solving the problem of finding one most critical vulnerability above (this is due to different attacker types having different combinations of attack options, thus forcing non-trivial interdependencies amongst the individual attack options). In general, if the defender wants to find *the most critical vulnerability set of size k*, then it would involve solving ${}^nC_k$ instances corresponding to different combinations of k vulnerabilities. Note that, the above approach is a brute force way of solving the problem which may not scale to large sized inputs, and we consider exploring efficient solutions as part of our future work.

5.2 Finding Most Sensitive Attacker Types

It is often the case that the web application administrator (defender) does not know accurately the probability with which a particular type of attacker might attack the system. In this scenario, it is important to analyze as to how much does the accuracy of the probabilities such as the ones shown in Table 3 impact the overall rewards of a given optimal strategy obtained by solving Equation 1. Thereby, the defender can make an effort to obtain the probability values corresponding to the sensitive attacker types accurately.

In order to analyze the sensitivity of various attacker types, we suggest the following approach: For each attacker type i, vary the corresponding probability p_i by $\pm x\%$ ($p_i^{new} = p_i(1 \pm \frac{x}{100})$) where x is the *sensitivity factor*, which can be varied from a low value to a high value as needed. Now, note that, $p_a = \frac{p_i \times x}{100}$ needs to be adjusted or distributed amongst the probabilities of the remaining attacker types. Here, it is important to make sure that this distribution is done such that the sensitivity of attacker i actually stands out, and the impact of change in the probability of values of other attacker types is minimal. For this, we propose to distribute p_a amongst the other attacker types using a weighted model as per their existing probabilities as shown below. For attacker j ($\neq i$), its new probability would be:

$$p_j^{new} = p_j(1 \mp \frac{p_a}{\sum\limits_{k(\neq i)} p_k}) \tag{2}$$

Note that, when $x\%$ is added to the probability p_i, then the sign in the above equation becomes negative, and vice-versa.

Once we set up this distribution, the sensitivity of an attacker type can be analyzed as follows: Let R_o be the overall reward value for the defender when the optimal mixed strategy obtained with probabilities $\{p_i\}$ is used with the new probabilities $\{p_i^{new}\}$, and R_n be the optimal reward value for the defender with an optimal strategy obtained with probabilities $\{p_i^{new}\}$ directly. Then, we compute the *Normalized Loss in Rewards (NLR)* for the defender as follows:

$$\text{NLR} = \frac{R_n - R_o}{R_n} \tag{3}$$

NLR acts as a measure for comparing sensitivities of different attacker types (note that, it is always ≥ 0 since $R_n \geq R_o$

Table 4: Reward values and gains for the defender with different probabilities for the attacker types. Each row corresponds to a given probability of the Database hacker attacking, and each column corresponds to a given probability of Mainstream hacker (remaining being that of the Nation State hacker). Values for each combination denote the rewards with uniform mixed strategy (UMS), BSG, and the gain respectively.

		P(Mainstream Hacker)																	
		0.0			0.2			0.4			0.6			0.8			1.0		
		UMS	BSG	Gain	UMS	BSG	Gain	UMS	BSG	Gain	UMS	BSG	Gain	UMS	BSG	Gain	UMS	BSG	Gain
P(DB Hacker)	0.0	-0.75	-0.38	**0.38**	-0.75	-0.05	**0.70**	-0.75	0.28	**1.02**	-0.75	0.60	**1.35**	-0.75	2.50	**3.25**	-0.75	4.79	**5.54**
	0.2	-0.60	-0.34	**0.26**	-0.60	-0.20	**0.40**	-0.60	0.12	**0.73**	-0.60	1.75	**2.35**	-0.60	4.03	**4.63**			
	0.4	-0.45	-0.01	**0.44**	-0.45	-0.01	**0.44**	-0.45	0.99	**1.44**	-0.45	3.27	**3.72**						
	0.6	-0.30	0.33	**0.63**	-0.30	0.33	**0.63**	-0.30	2.51	**2.81**									
	0.8	-0.15	0.66	**0.81**	-0.15	1.76	**1.91**												
	1.0	-0.00	1.00	**1.00**															

by virtue of optimality of R_n). The attacker types for which the loss (NLR value) is higher than the rest are the most sensitive attacker types, and their probabilities should be calibrated with high accuracy for maximizing the security.

6. EXPERIMENTAL RESULTS

Now, we present the experimental results related to generation of optimal strategies of the defender, the performance of the DOBSS based MIQP method [35], and the gain in the reward values for the defender compared to the case where a uniform mixed strategy [6, 36] is adopted. We also present our findings on the most critical vulnerabilities and most sensitive attacker types in the given example. We performed the experiments on a machine with Intel Core 2 Quad CPU Q9400 2.66GHz and 2GB RAM. For solving the MIQP, we used the Gurobi Optimizer [15].

6.1 Comparison of the Optimal BSG Strategy with Uniform Mixed Strategy

First, we show the results obtained when the working example presented in the previous section is fed as input to the algorithm. The optimal mixed strategy obtained is: $\langle 0.43, 0.41, 0, 0.16 \rangle$ respectively for the four configurations available for the defender and the corresponding optimal reward value for the defender is 0.91 whereas the reward value when using a uniform mixed strategy (0.25 for each configuration) is -0.55. Since we considered negative reward values for losing positions and positive for winning positions, clearly using BSGs lead to winning strategies for the defender whereas a uniform mixed strategy could be a losing one. It means that the defender should choose the combination MySQL & Python with a probability 0.43, PostgreSQL & Python with probability 0.41 and PostgreSQL & PHP with probability 0.16. The algorithm has taken 1.121s time on an average over 20 runs using all the 4 cores of the machine (with 4 threads) for producing the output.

Next, we present the results when the probabilities for the attacker types is varied in the given working example. Here we consider only three types of attackers for ease of presentation of results: Database (DB) hacker, Mainstream hacker, and the Nation State hacker. Table 4 shows the reward values for the defender when a Uniform Mixed Strategy (UMS) is employed, when BSG mapping is used, and the gain due to BSG over the uniform strategy, for different probability values corresponding to the types DB hacker and Mainstream hacker (the remaining probability being attributed to the Nation State hacker).

A row with probability 0.2 and a column with probability 0.4 would correspond to the attacker types- DB hacker with probability 0.2, Mainstream hacker with probability 0.4 and the Nation State hacker with probability 0.4 $(1 - 0.2 - 0.4)$. The values at the corresponding position indicate that employing a uniform mixed strategy (where all 4 strategies of the defender are chosen with equal probability 0.25) would result in a reward value of -0.6 for the defender, whereas a BSG based mixed strategy would result in a reward value of 0.12 resulting in a gain of 0.73 (subject to round off errors) for BSG against the uniform mixed strategy. Note that the sum of probabilities of the different attacker types attacking should be equal to 1. Therefore, the positions where the sum of probabilities exceeds 1 are blank, as they are not applicable.

Note that, BSG based method being optimal in nature is guaranteed to perform at-least as good as the uniform mixed strategy. The tabulated results help in getting a perspective as to how significant the gains could be over the uniform mixed strategy. On an average (and also in most cases), we see that the uniform mixed strategy results in a $-$ve reward value of -0.50 for the defender, whereas the BSG based mixed strategy results in a $+$ve reward value of 1.14, demonstrating its clear advantage (winning as opposed to losing).

Table 5: Results of finding the (one) most critical vulnerability. The most critical one is highlighted in bold.

Vulnerabilities addressed	New Optimal Reward
$\{v_1\}$	0.344
$\{v_2\}$	0.825
$\mathbf{\{v_3\}}$	**2.514**
$\{v_4\}$	1.305

6.2 Most Critical Vulnerabilities

Now, we present the results of analysis for finding the most critical vulnerabilities in the given example. Note that, each of the first four attacks A1–A4 in Table 2 correspond to unique vulnerabilities (say, v_1–v_4) which in-turn impact the attacks A5–A8. Accordingly, we have considered fixing each of these vulnerabilities separately to examine- which of them are the most critical ones. Note that, fixing v_1 would result in the removal of attacks A1, A5 & A7. Similarly, fixing v_2 would account for A2, A6 & A8, v_3 would account for A3, A5 & A6, and v_4 corresponds to A4, A7 & A8.

Tables 5, 6, and 7 show the results of finding the most critical sets of vulnerabilities of sizes 1, 2, and 3, respectively.

Table 6: Results of finding the most critical sets of vulnerabilities of size 2. The most critical ones are highlighted in bold.

Vulnerabilities addressed	New Optimal Reward
$\{v_1, v_2\}$	-0.5
$\{v_1, v_3\}$	1.795
$\{v_2, v_4\}$	1.415
$\mathbf{\{v_2, v_3\}}$	**2.514**
$\mathbf{\{v_2, v_4\}}$	**2.514**
$\{v_3, v_4\}$	1.0

Table 7: Results of finding the most critical sets of vulnerabilities of size 3. The most critical ones are highlighted in bold.

Vulnerabilities addressed	New Optimal Reward
$\mathbf{\{v_1, v_2, v_3\}}$	**2.5**
$\mathbf{\{v_1, v_2, v_4\}}$	**2.5**
$\{v_1, v_3, v_4\}$	1.0
$\{v_2, v_3, v_4\}$	1.0

Addressing any one of the highlighted sets of vulnerabilities in each case would lead to maximum reward positions for the defender, and hence preferred. The number of vulnerabilities to be considered at a time can be decided based on the resources available with the defender.

6.3 Most Sensitive Attacker Types

Finally, we present the results analyzing the sensitivity of different attacker types and their probabilities in the given example. As mentioned in the previous section, this is examined by computing the Normalized Loss in Rewards (NLR) values for the defender for each change in probability. Figure 2 shows the NLR values for the four types of attacker when Sensitivity factor is varied from -100% to 100%. We see that the Mainstream hacker and the Script Kiddie are more sensitive attacker types compared to the rest, in the given example. The former type is sensitive due to the high reward values its attack options have and the high probability value it holds, whereas the latter type is sensitive due to the probability distribution it will exert on other types if its value is underestimated. Database hacker is not sensitive due to the low reward values its attack options have for the defender, and Nation State is relatively less sensitive as change in its value is being neutralized by the rise in all other types' values since they together have all the attack options of the Nation State in the given example. Therefore, the probability values corresponding to the Mainstream hacker and the Script Kiddie should be estimated with high precision as part of input, for obtaining effective movement strategies.

7. DISCUSSION

Note that, the strategies of the attackers used for obtaining the mixed strategy for the defender are either known attacks which are not yet addressed in the web application or an estimation of possible future attacks. Clearly, this estimation is vital to the design of effective MTDs. One way to develop good estimates for the unknown attacks could be via analysis of the attack graphs. However, since the complete attack graphs are often very large, one may have to reason on simpler but realistic models of the actual attack graphs. These models, being approximate in nature, no longer confirm to the closed world assumptions. Hence one may analyze these with the help of frameworks such as closed-world reasoning under incomplete information [11, 12].

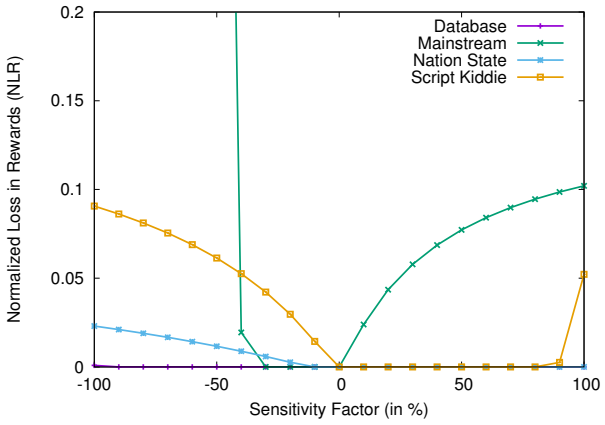

Figure 2: NLR values for various attacker types when their probabilities are modified ranging from -100% to 100%.

Impact and modeling of uncertainties in reward matrix values and probabilities of the types of attacker were studied in the literature [24, 37, 38], which may be useful for web application security as well. Also, teaming between multiple defenders has been explored in [43] which can be explored in the case of web applications for better security using/across multiple servers.

Finally, note that the attacker models for web applications evolve over time as the attackers will be able to perform reconnaissance and discover new vulnerabilities, which needs to be taken into account as time passes from the launch of the web application. Further, each move that the defender makes in the case of web applications may involve costs such as system downtime, etc. which need to be kept within given limits. Also, the switching between different configurations is assumed to be from all to all, which may be restricted to limited combinations for performance/feasibility reasons, which then brings-in the state information into the game, requiring planning policies for the defender. These variations of the security games have not been studied previously and can be pursued.

8. CONCLUSION

We considered the problem of finding effective movement policies when securing web applications through Moving Target Defense. An important aspect of this problem is that the attackers can perform reconnaissance about the strategy of the defender before attacking. We observe that this behavior is similar to that of the leader-follower relationship in Stackelberg security games. Further, we note that the uncertainty in the type of attacker could be captured via Bayesian games. Accordingly, we present an effective methodology to map the Moving Target Defense problem as a Bayesian Stackelberg game so as to obtain optimal mixed

strategies maximizing the rewards for the defender (minimizing the damage). We have also proposed techniques to identify most critical vulnerabilities and most sensitive attacker types, which help in improving the security of web application systems further. Experimental results over a representative example from the web application domain show the significant gains of the proposed approach over the uniform mixed strategy for the defender.

9. ACKNOWLEDGMENTS

This work was partially supported by the grants from National Science Foundation (NSF-SFS-1129561) and the Center for Cybersecurity and Digital Forensics at Arizona State University.

REFERENCES

[1] D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna. Saner: Composing static and dynamic analysis to validate sanitization in web applications. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 387–401. IEEE, 2008.

[2] E. G. Barrantes, D. H. Ackley, T. S. Palmer, D. Stefanovic, and D. D. Zovi. Randomized instruction set emulation to disrupt binary code injection attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 281–289. ACM, 2003.

[3] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 57–64. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[4] K. M. Carter, J. F. Riordan, and H. Okhravi. A game theoretic approach to strategy determination for dynamic platform defenses. In *Proceedings of the First ACM Workshop on Moving Target Defense*, MTD '14, pages 21–30, New York, NY, USA, 2014. ACM.

[5] M. Carvalho and R. Ford. Moving-target defenses for computer networks. *Security Privacy, IEEE*, 12(2):73–76, Mar 2014.

[6] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, EC '06, pages 82–90, New York, NY, USA, 2006. ACM.

[7] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th international conference on World wide web*, pages 281–290. ACM, 2010.

[8] A. Cui and S. J. Stolfo. Symbiotes and defensive mutualism: Moving target defense. In *Moving Target Defense*, pages 99–108. Springer, 2011.

[9] A. Doupé, L. Cavedon, C. Kruegel, and G. Vigna. Enemy of the state: A state-aware black-box web vulnerability scanner. In *USENIX Security Symposium*, 2012.

[10] A. Doupé, W. Cui, M. H. Jakubowski, M. Peinado, C. Kruegel, and G. Vigna. deDacota: toward preventing server-side XSS via automatic code and data separation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1205–1216. ACM, 2013.

[11] O. Etzioni, K. Golden, and D. S. Weld. Tractable closed world reasoning with updates. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994.*, pages 178–189, 1994.

[12] O. Etzioni, K. Golden, and D. S. Weld. Sound and efficient closed-world reasoning for planning. *Artif. Intell.*, 89(1-2):113–148, 1997.

[13] D. Evans, A. Nguyen-Tuong, and J. Knight. Effectiveness of moving target defenses. In *Moving Target Defense*, pages 29–48. Springer, 2011.

[14] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna. Toward automated detection of logic vulnerabilities in web applications. In *USENIX Security Symposium*, pages 143–160, 2010.

[15] Gurobi. Gurobi optimizer 5.0. *URL: http://www. gurobi. com*, 2012.

[16] J. C. Harsanyi and R. Selten. A generalized nash solution for two-person bargaining games with incomplete information. *Management Science*, 18(5-part-2):80–106, 1972.

[17] M. Jain, E. Kardes, C. Kiekintveld, F. Ordóñez, and M. Tambe. Security games with arbitrary schedules: A branch and price approach. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[18] M. Jain, C. Kiekintveld, and M. Tambe. Quality-bounded solutions for finite bayesian stackelberg games: Scaling up. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pages 997–1004, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[19] S. Jones, A. Outkin, J. Gearhart, J. Hobbs, J. Siirola, C. Phillips, S. Verzi, D. Tauritz, S. Mulder, and A. Naugle. Evaluating moving target defense with pladd. Technical report, Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), 2015.

[20] N. Jovanovic, C. Kruegel, and E. Kirda. Static analysis for detecting taint-style vulnerabilities in web applications. *Journal of Computer Security*, 18(5):861–907, 2010.

[21] G. Kc, A. D. Keromytis, and V. Prevelakis. Countering code-injection attacks with instruction-set randomization. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 272–280. ACM, 2003.

[22] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 689–696, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[23] C. Kiekintveld, V. Lisỳ, and R. Píbil. Game-theoretic foundations for the strategic use of honeypots in network security. In *Cyber Warfare*, pages 81–101. Springer, 2015.

[24] C. Kiekintveld, J. Marecki, and M. Tambe. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pages 1005–1012, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[25] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, 2010.

[26] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in Security Games: An Extended Investigation of Interchangeability, Equivalence, and Uniqueness. *J. Artif. Int. Res.*, 41(2):297–327, May 2011.

[27] G. Leitmann. On generalized stackelberg strategies. *Journal of Optimization Theory and Applications*, 26(4):637–643, 1978.

[28] J. Letchford, V. Conitzer, and K. Munagala. Learning and approximating the optimal strategy to commit to. In M. Mavronicolas and V. Papadopoulou, editors, *Algorithmic Game Theory*, volume 5814 of *Lecture Notes in Computer Science*, pages 250–262. Springer Berlin Heidelberg, 2009.

[29] Y. Li, R. Dai, and J. Zhang. Morphing communications of cyber-physical systems towards moving-target defense. In *Communications (ICC), 2014 IEEE International Conference on*, pages 592–598, June 2014.

[30] S. Luber, Z. Yin, F. M. D. Fave, A. X. Jiang, M. Tambe, and J. P. Sullivan. Game-theoretic patrol strategies for transit systems: the TRUSTS system and its mobile app. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, pages 1377–1378, 2013.

[31] P. Manadhata. Game theoretic approaches to attack surface shifting. In S. Jajodia, A. K. Ghosh, V. Subrahmanian, V. Swarup, C. Wang, and X. S. Wang, editors, *Moving Target Defense II*, volume 100 of *Advances in Information Security*, pages 1–13. Springer New York, 2013.

[32] J. Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[33] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein. Finding focus in the blur of moving-target techniques. *Security & Privacy, IEEE*, 12(2):16–26, 2014.

[34] H. Okhravi, J. Riordan, and K. Carter. Quantitative evaluation of dynamic platform techniques as a defensive mechanism. In *Research in Attacks, Intrusions and Defenses*, pages 405–425. Springer, 2014.

[35] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[36] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed ARMOR protection: the application of a game theoretic model for security at the los angeles international airport. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Industry and Applications Track Proceedings*, pages 125–132, 2008.

[37] J. Pita, M. Jain, F. Ordóñez, M. Tambe, S. Kraus, and R. Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 369–376, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[38] J. Pita, M. Jain, M. Tambe, F. OrdÃşÃśez, and S. Kraus. Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition. *Artificial Intelligence*, 174(15):1142 – 1171, 2010.

[39] J. Pita, M. Tambe, C. Kiekintveld, S. Cullen, and E. Steigerwald. GUARDS: Game Theoretic Security Allocation on a National Scale. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, pages 37–44, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[40] A. Prakash and M. P. Wellman. Empirical game-theoretic analysis for moving target defense. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, MTD '15, pages 57–65, New York, NY, USA, 2015. ACM.

[41] J. Robertson and M. Riley. JPMorgan Hack Said to Span Months Via Multiple Flaws. In *Bloomberg*, 2014.

[42] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 13–20, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[43] E. A. Shieh, A. X. Jiang, A. Yadav, P. Varakantham, and M. Tambe. Unleashing Dec-MDPs in Security Games: Enabling Effective Defender Teamwork. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 819–824, 2014.

[44] J. Silver-Greenberg, M. Goldstein, and N. Perlroth. JPMorgan Chase Hacking Affects 76 Million Households. In *The New York Times*, 2014.

[45] A. Sinha, T. Nguyen, D. Kar, M. Brown, M. Tambe, and A. X. Jiang. From physical security to cyber security. *Journal of Cybersecurity*, 2016.

[46] J. Spencer. A deception game. In *American Mathematical Monthly*, pages 416–417, 1973.

[47] M. Taguinod, A. Doupé, Z. Zhao, and G.-J. Ahn. Toward a Moving Target Defense for Web Applications. In *Proceedings of 16th IEEE International Conference on Information Reuse and Integration (IRI)*, 2015.

[48] M. Thompson, N. Evans, and V. Kisekka. Multiple os rotational environment an implemented moving target defense. In *Resilient Control Systems (ISRCS), 2014 7th International Symposium on*, pages 1–6, Aug 2014.

[49] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest. Flipit: The game of "stealthy takeover". *Journal of Cryptology*, 26(4):655–713, 2013.

[50] O. Vaněk, Z. Yin, M. Jain, B. Bošanský, M. Tambe, and M. Pěchouček. Game-theoretic resource allocation for malicious packet detection in computer networks. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, pages 905–912, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[51] S. Vikram, C. Yang, and G. Gu. Nomad: Towards non-intrusive moving-target defense against web bots. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 55–63. IEEE, 2013.

[52] D. Wichers. Owasp top-10 2013. *OWASP Foundation*, 2013.

[53] M. Winterrose, K. Carter, N. Wagner, and W. Streilein. Adaptive attacker strategy development against moving target cyber defenses. *arXiv preprint arXiv:1407.8540*, 2014.

[54] R. Zhuang, S. A. DeLoach, and X. Ou. Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 31–40. ACM, 2014.