Sequential Manipulative Attacks in Security Games

Thanh H. Nguyen[1], Arunesh Sinha[2]
[1]University of Oregon
tnguye11@uoregon.edu
[2]Singapore Management University
aruneshs@smu.edu.sg

ABSTRACT

This paper studies the problem of sequential manipulative attacks in Stackelberg security games in which the attacker attempts to orchestrate its attacks over multiple time steps to mislead the defender's learning of the attacker's behavior, which will eventually influence the defender's patrol strategy towards the attacker's benefit. Previous work along this line of research only focuses on one-shot games in which the defender only learns the attacker's behavior and then design his corresponding strategy once. Our work, on the other hand, investigates the long-term impact of the attacker's manipulation in which current attack and defense choices of players will determine the future learning and patrol strategy of the defender. In particular, we have three key contributions. First, we introduce a new multi-step manipulative attack game model that captures the impact of sequential manipulative attacks carried out by the attacker over the entire time horizon. Second, we propose a new algorithm to compute an optimal manipulative attack plan for the attacker, which tackles the challenge of multiple connected optimization components involved in the computation across multiple time steps. Finally, we present our preliminary experimental results on the impact of such misleading attacks, showing a significant benefit for the attacker and loss for the defender.

KEYWORDS

Security Games, Adversarial Learning, Sequential Attacks

ACM Reference Format:
Thanh H. Nguyen[1], Arunesh Sinha[2]. 2021. Sequential Manipulative Attacks in Security Games. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), London, UK, May 3–7, 2021*, IFAAMAS, 7 pages.

1 INTRODUCTION

Stackelberg security games (SSGs) have been widely applied for solving many real-world problems in public safety and security, cybersecurity, and conversations [5, 22, 25, 28]. In fact, there have been several applications of SSGs such as ARMOR (used by police officers for protecting airport terminals at Los Angeles International Airport) [22], PROTECT (used by US Coast Guard officers to protect ferries) [25], and PAWS (used by NGOs in multiple national parks across the world for protecting wildlife) [5], etc. SSGs are well known for well capturing strategic interactions between a defender and an attacker over a set of important targets.

In recent work, machine learning-based techniques have been used for modeling and predicting the attacker's behavior based on collected attack data [7, 11, 26, 30]. Several different proposed behavior models were shown to have a high prediction accuracy of attack activities in different security-related domains. For example, in PROTECT, Quantal Response was used to predict decision making of the attacker in the domain of ferry protection [25]. In addition, in the PAWS-related work, different models such as Quantal Response, logistic regression, and decision tree, etc were used to capture the behavior of poachers (i.e., predicting where the poachers are likely to set trapping tools such as snares to catch wild animals)[5, 7, 11, 26]. These behavior models, after being trained, will be incorporated in to determining an optimal strategy of the defender.

However, as pointed out in previous work, since the defender relies on some attack data to make prediction, the attacker can intentionally change its attack behavior to misleading the defender's learning [19, 20]. A failed learning result would cause the defender to choose ineffective patrolling strategies, which will benefit the attacker in the end. Intuitively, the attacker is perfectly rational, but pretends to act in a boundedly rational manner. The attacker may suffer an immediate loss for deviating from a myopic optimal response, but it will gain a significantly more benefit as a result of the defender's deteriorated strategies. In this work, we focus on analyzing such manipulative attacks of the attacker.

There are, in fact, some existing work which belong to this line of research [19, 20]. The existing work mainly studies one-shot game scenarios in which the defender only performs the learning of the attacker behavior once and then commits to a single defense strategy afterward. However, in many real-world domains such as wildlife protection, the defender and attacker often interact in a repeated manner. That is, at each time step, given a historical attack and patrol data, the defender updates his learning of the attacker's behavior and re-generates a new defense strategy while the attacker responds accordingly by launching a certain number of attacks. These new defense and attack actions are then collected for the future use. This learning-patrolling-attack loop will continue until the end of the time horizon. In this multi-step interaction scenario, it is clear that the existing one-shot SSG studies may fail to capture the long-term impact of the attacker's manipulation.

In this work, we study the problem of sequential manipulative attacks in multi-step SSGs. We aim at investigating the long-term manipulative decisions of the attacker and the accumulative impact of the attacker's manipulation on both the defender and attacker's utility. We provide the following three key contributions. First, we introduce a new multi-step manipulative attack game model. In our game model, the defender follows a two-stage learning-patrolling at each time step to play. On the other hand, at each time step, the attacker attempts to find an optimal attack strategy given the current

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, London, UK. © 2021 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

defense strategy, taking into account the tradeoff between the immediate utility loss for playing boundedly rational at current time step and the future utility gain for misleading the defender. Second, we present a new algorithm to compute such optimal manipulative attack plan for the attacker. The key challenge of computing an optimal attack plan is that it involves multiple connected optimization components over the entire time horizon, which is not straightforward to solve. In order to tackle this computational challenge, our new algorithm follows the Projected Gradient Descent (PGD) approach to iteratively update the attack plan based on the gradient of the attacker's utility with respect to its attacks. Inspired by hyper-parameter learning, we then determine this gradient based on the recursive relationships of the gradient components involved in the gradient updating steps of the inner optimization levels. Finally, we provide a preliminary experimental analysis on the impact of the attacker's sequential attack manipulation on the accumulated utility of both players. We show that the attacker gains a substantially higher utility wile the defender suffers a significant loss as a result of the attacker's manipulation.

2 RELATED WORK

Attacker behavior modeling is an important research line in SSGs which focuses on building behavior models of the attacker in various security-related domains such as wildlife protection [7, 11, 30]. Several different models were proposed before, including Quantal Response, decision trees, and neural nets-based models. These models enable the defender to predict boundedly rational decisions of human attackers such as poachers using historical attack data as well as other domain-dependent data. For example, in wildlife protection, rangers can collect poaching signs such as snares during their patrols [5]. These observations of rangers are then used as poaching data to predict poaching activities in the future. These behavior models, after being trained, are then integrated into generating effective patrolling strategies for the defender.

However, there is a rising concern about the vulnerability of these learning-patrolling methods against a deceptive attacker who intentionally maneuvers its attacks to *fool* the defender's learning. Such manipulation of the attacker could lead to poor-quality patrolling strategies of the defender. Previous work has demonstrated that weakness of the learning-patrolling methods in one-shot SSGs [19, 20]. For example, Nguyen et. al study the security situation in which a rational attacker (among other boundedly-rational attackers) tries to influence a portion of attack data (which that attacker can control) in one-shot games where the defender relies on Quantal Response to predict the behavior of the entire attacker population [19]. Beside this line of work, there is another research direction which studies the attacker's manipulation when the defender uses attack data to predict the attacker's type. Related work has so far looked into a simple learning situation in which the defender uses the Bayes rule method to update his belief about the attacker's type over time [21].

Our work is also related to adversarial learning in machine learning in the sense that there is an adversary who attempts to fail machine learning algorithms by, for example, attacking the training/testing data or interfering with the learning process. Poisoning attacks (i.e., altering the training data) are perhaps the most closely related to our work [10, 13, 15, 27, 31]. Different attack methods were designed to deteriorate the performance of standard machine learning algorithms such as Support Vector Machine and neural nets, etc. Differentiating from this research line, in our problem, decision quality (which is measured via utilities of players) in terms of defense and attack strategies is the ultimate objective of both the defender and attacker, rather than just the prediction accuracy.

Finally, in a somewhat related research area, Secrecy and Deception in Security Games, previous work investigated situations in which information available to the defender and attacker is asymmetric [3, 6, 8, 9, 23, 29, 32]. They then determine how the defender should strategically reveal or disguise his information to the attacker so as to influence the attacker's decision for the sake of the defender's benefit.

In this work, we study the attacker's manipulations of its sequential attacks over multiple time steps whereas the defender uses Quantal Response to predict the attacker's behavior.[1] Unlike the one-shot SSGs, in this multi-step scenario, reasoning about the attacker's decisions at each time step has to take into account the impact of such decisions on future interaction outcomes of players.

3 PRELIMINARIES

Stackelberg security games (SSGs). SSGs are a class of leader-follower games in which a defender has to allocate a limited number of security resources, S, over a set of important targets $\{1, \dots, N\}$ to protect these targets against an attacker. In one-short SSGs, a pure strategy of the defender is an assignment of security resources to the targets. A mixed strategy of the defender is a probability distribution over these pure strategies. In the context of no resource-scheduling constraints, a mixed strategy of the defender can be equivalently represented as a marginal probability vector $\mathbf{x} = \{x_n\}$ where $x_n \in [0, 1]$ is the marginal coverage probability at target n and $\sum_n x_n \leq S$. We denote by $\mathbf{X} = \{\mathbf{x} : \sum_n x_n \leq S, 0 \leq x_n \leq 1, \forall n\}$ the set of all feasible mixed strategies of the defender. In one-short SSGs, the attacker is assumed to be aware of the defender's mixed strategy and attacks one of the targets accordingly.

In SSGs, the players' payoff depends on which target the attackers attacks and whether the defender is protecting that target or not. In particular, when the attacker attacks a target n, if the defender is not protecting n, the attacker will receive a reward of R^a_n while the defender gets a penalty of P^a_n. Conversely, if the defender is protecting n, the attacker gets a penalty $P^a_n < R^a_n$ and the defender obtains a reward $R^d_n > P^d_n$. Given a mixed strategy of the defender $\mathbf{x}$, when the attacker attacks n, the defender and attacker's expected utility at n is computed as follows:

$$U^d_n(x_n) = x_n(R^d_n - P^d_n) + P^d_n \tag{1}$$

$$U^a_n(x_n) = x_n(P^a_n - R^a_n) + R^a_n \tag{2}$$

A standard game-theoretic solution concept in SSGs is Strong Stackelberg Equilibrium (SSE) in which players play optimally against each other. We denote by $br(\mathbf{x}) \in \arg\max_n U^d_n(x_n)$ the attacker's best response to the defender's strategy $\mathbf{x}$. Then, formally, a pair $(\mathbf{x}^*, br(\mathbf{x}^*))$ form an SSE if and only if:

- The attacker plays a best response: $br(\mathbf{x}^*)$.

[1]Note that, even though we focus on the QR-behavior modeling scenario in this work, our methods can be generalized to any differentiable behavior models.

- The defender plays an optimal strategy:

$$\mathbf{x}^* \in \arg\max_{\mathbf{x}} U^d_{br(\mathbf{x})}(x_{br(\mathbf{x})})$$

Quantal Response attack behavior model. Quantal Response is a well-known behavior model used in both behavioral economics and game theory [16, 17, 30]. While an SSE considers a perfectly rational attacker, QR assumes a boundedly rational attacker who attacks each target n with the following probability:

$$q_n(\mathbf{x}, \lambda) = \frac{e^{\lambda U^a_n(x_n)}}{\sum_{n'} e^{\lambda U^a_{n'}(x_{n'})}}$$

where $\lambda \geq 0$ is the model parameter which controls the attacker's rationality. Intuitively, the higher the value of λ is, the more rational the attacker is. In particular, when $\lambda = 0$, the attacker is non-strategic; it attacks each target uniformly at random. On the other hand, when $\lambda = +\infty$, the attacker is perfectly rational; it only attacks targets with the highest expected utility. In practice, the model parameter λ is derived based on historical attack data collected by the defender and is optimized via the Maximum Likelihood Estimation (MLE) [18]. Given the attacker plays according to QR, the defender and attacker's utility is computed as follows:

$$U^d(\mathbf{x}, \lambda) = \sum_n q_n(\mathbf{x}, \lambda) U^d_n(x_n)$$

$$U^a(\mathbf{x}, \lambda) = \sum_n q_n(\mathbf{x}, \lambda) U^a_n(x_n)$$

Multi-step sequential learning, patrolling, and attacking. In many real-world security domains such as wildlife protection, the defender and attacker repeatedly interact with each other through a multi-step learning-patrolling-attacking loop. The one-shot SSG model is then extended to capture such security scenarios. The overview of this multi-stage interaction loop is illustrated in Figure 1.

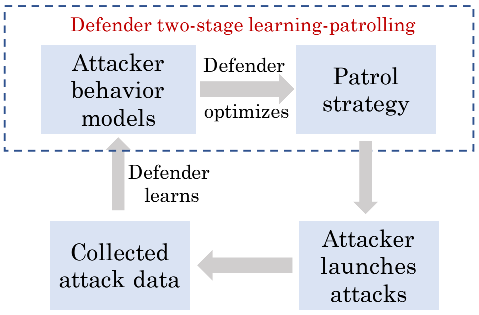


Figure 1: Learning-patrolling-attacking loop in the multi-step Stackelberg security games

Formally, at each time step t, let's denote by $(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1})$ the historical patrolling strategies and attacks at previous time steps. $(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1})$ is also the data the defender uses to learn the attacker's behavior. In particular, $\mathbf{X}_{t-1} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}\}$ where $\mathbf{x}_{t'} = \{x_{t',1}, x_{t',2}, \ldots, x_{t',N}\}$ with $t' \leq t-1$ is the defender's mixed strategy at time step t'. In addition, $\mathbf{Z}^{t-1} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{t-1}\}$ where $\mathbf{z}_{t'} = \{z_{t',1}, z_{t',2}, \ldots, z_{t',N}\}$ is the attack density distribution at time step t' (i.e., $z_{t',n}$ is the number of times the attackers attacks target n at time step t'). At each time step t, the defender's strategy $\mathbf{x}_t$ are determined based on a two-stage learning-patrolling:

- Learning: The defender optimize the QR parameter λ_t based on $(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1})$, which is the result of the MLE problem:

$$\max_{\lambda} logL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \lambda)$$
$$\text{s.t. } \lambda \geq 0$$

where $logL$ is the log-likelihood function:

$$logL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \lambda) = \sum_{t'=1}^{t-1} \sum_{n=1}^{N} z_{t',n} \log[q_n(\mathbf{x}_{t'}, \lambda)]$$

- Patrolling: Given the learning outcome λ_t, the defender finds an optimal strategy $\mathbf{x}_t$ accordingly, which is an optimal solution of the following optimization problem:

$$\max_{\mathbf{x}} U^d(\mathbf{x}, \lambda_t)$$
$$\text{s.t. } \sum_n x_n \leq S$$
$$0 \leq x_n \leq 1, \forall n$$

which maximizes the defender's utility with respect to the attacker's QR-behavior parameter λ_t.

At the first time step $t = 1$, in particular, the defender does not have any data. Therefore, the defender can choose any strategy $\mathbf{x}_1$ to play. For example, the defender can play the *SSE* strategy.

4 MANIPULATIVE SEQUENTIAL ATTACKS

Since the defender relies on historical attack data to learn the attacker's behavior, a clever attacker can orchestrate its attacks to *fool* the defender, influencing the defender's learning and as a result, leading to ineffective patrolling strategies which benefit the attacker. In our model, the attacker is perfectly rational, but *pretends* to be boundedly rational to mislead the defender. By acting in this manipulative way, the attacker suffers some immediate utility loss (for playing boundedly rational) but would obtain a significant long term benefit as the result of its influence on the defender's patrolling strategies. The attacker goal is to find an optimal manipulative sequential-attack strategy that maximizes the attacker's accumulative expected utility across the entire time horizon, given the trade-off between the loss and benefit of such pretentious boundedly rational playing. In this paper, we will focus on analyzing such manipulative attacks in the security scenario in which the defender follows the QR model to predict the attacker's behavior. However, our method can be extend to any differentiable behavior model such as SUQR and neural nets, etc.

Formally, this problem can be represented as the following:

$$\max_{\mathbf{z}} \sum_t U^a(\mathbf{x}_t, \mathbf{z}_t) \tag{3}$$

$$\text{s.t. } \lambda_t \in \arg\min_{\lambda \geq 0} logL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \lambda), \forall t \tag{4}$$

$$\mathbf{x}_t \in \arg\max_{\mathbf{x} \in \mathbf{X}} U^d(\mathbf{x}, \lambda_t), \forall t \tag{5}$$

$$\sum_n z_{t,n} \leq K, z_{t,n} \in \mathbb{N}, \forall n, t \tag{6}$$

which maximizes the attacker's accumulated expected utility over the entire time horizon. In particular, the expected utility of the attacker at time step t is computed as follows:

$$U^a(\mathbf{x}_t, \mathbf{z}_t) = \sum_{n'} z_{t,n'} U^a_{n'}(x_{t,n'})$$

Constraints (4–5) represent the two-stage learning-patrolling of the defender at each time step t. Constraint (6) ensure that the attacker can only launch at most K attacks at each time step. The constant K represents the attacker's limited capability in influencing the defender's learning.

Overall, the problem (3–6) consists of multiple connected optimization levels. The decision on which targets and how frequently to attack at each time step not only influence the utility outcome at current time step but also affect the learning outcomes of the defender in future time steps. As a result, that attack decision of the attacker will contribute to the future utility outcomes that the attacker will receive. Apparently, the problem (3–6) is challenging to solve. We propose to relax the attack variables $\{z_{t,n}\}$ to be continuous and then apply the Projected Gradient Descent (PGD) approach to solve it. Essentially, starting with some initial values of $\mathbf{z}^0 = \{\mathbf{z}^0_1, \mathbf{z}^0_2, \ldots, \mathbf{z}^0_T\}$, PGD iteratively updates the values of these attack variables based on the gradient step. Let's denote by $F = \sum_t U^a(\mathbf{x}_t, \mathbf{z}_t)$, at each update iteration i, given the current estimation $\mathbf{z}^{i-1} = \{\mathbf{z}^{i-1}_1, \mathbf{z}^{i-1}_2, \ldots, \mathbf{z}^{i-1}_T\}$, we update:

$$\mathbf{z}^i = \mathbf{z}^{i-1} + \alpha \frac{dF}{d\mathbf{z}^{i-1}} \tag{7}$$

where $\alpha > 0$ is the step size. PGD then projects the updated value into the feasible region by finding the closest point in the region $\mathbf{Z} = \{\mathbf{z} : \sum_{t,n} z_{t,n} \leq K, z_{t,n} \geq 0, \forall t, n\}$.This projection step is done via solving the following optimization problem:

$$\min_{\mathbf{z} \in \mathbf{Z}} ||\mathbf{z}^i - \mathbf{z}||_2$$

which is a convex optimization problem and therefore can be solved optimally using any optimization solver (such as Matlab Optimization) This update process will continue until convergence (i.e., the update does not improve the attacker utility in (3)). Once converged, we obtain a local optimal solution of (3–6). By running this iterative process multiple times with different initial values of the attack variables, we get multiple local optimal solutions. The final solution will be the best with the highest accumulated utility for the attacker among the local optimal ones.

Now, the core of PGD is to compute the gradient of the attacker utility at a value $\mathbf{z}$ of the attack variables:

$$\begin{aligned}\frac{dF}{d\mathbf{z}} &= \sum_t \frac{dU^a(\mathbf{x}_t, \mathbf{z}_t)}{d\mathbf{z}} \\ &= \frac{dz_{t',n'}}{d\mathbf{z}} U^a_{n'}(x_{t',n'}) + \sum_{t'} \sum_{n'} z_{t',n'} (P^a_{n'} - R^a_{n'}) \frac{dx_{t',n'}}{d\mathbf{z}}\end{aligned}$$

which depends on the two gradient components $\frac{dz_{t',n'}}{d\mathbf{z}}$ and $\frac{dx_{t',n'}}{d\mathbf{z}}$. The first component, $\frac{dz_{t',n'}}{d\mathbf{z}}$, is the gradient of the number of attacks at each target and time step with respect to other targets and steps. This component is simply determined as follows:

$$\begin{aligned}\frac{\partial z_{t',n'}}{\partial z_{t,n}} &= 0 \text{ if } t' \neq t \text{ or } n' \neq n \\ \frac{\partial z_{t',n'}}{\partial z_{t,n}} &= 1, \text{ otherwise.}\end{aligned}$$

The second component is $\frac{dx_{t',n'}}{d\mathbf{z}} = \{\frac{\partial x_{t',n'}}{\partial z_{t,n}}\}$, which is the gradient of the defender's strategy at each time step with respect to the number of attacks across all targets and time steps. Note that $\frac{\partial x_{t',n'}}{\partial z_{t,n}}$ is non-zero only when $t' > t$ since the defender's strategy at each step only depends on the historical attacks at previous time steps. By applying the chain rule, it can be decomposed into two parts:

$$\begin{aligned}\frac{\partial x_{t',n'}}{\partial z_{t,n}} &= \frac{\partial x_{t',n'}}{\partial \lambda_{t'}} \cdot \frac{\partial \lambda_{t'}}{\partial z_{t,n}}, \forall t' > t \\ \frac{\partial x_{t',n'}}{\partial z_{t,n}} &= 0, \forall t' \leq t\end{aligned}$$

The challenge is that, even though $x_{t',n'}$ depends on $\lambda_{t'}$ and $\lambda_{t'}$ depends on $z_{t,n}$, we do not have a closed form of $x_{t',n'}$ and $\lambda_{t'}$ as a function of $\lambda_{t'}$ and $z_{t,n}$, respectively. Inspired by hyper-parameter learning [14], we assume the defender uses the projected gradient descent approach to optimize her learning and strategy in (4) and (5). In the following, we present our PGD-based method to estimate the gradient components $\frac{\partial x_{t',n'}}{\partial \lambda_{t'}}$ and $\frac{\partial \lambda_{t'}}{\partial z_{t,n}}$ for all $t' > t$.[2]

4.1 Compute the gradient $\frac{d\mathbf{x}_t}{d\lambda_t}$

The defender strategy at time step t, $\mathbf{x}_t$, is an optimal solution of:

$$\begin{aligned}\max_{\mathbf{x}} \; & U^d(\mathbf{x}, \lambda_t) \\ \text{s.t.} \; & \sum_n x_n \leq S \\ & 0 \leq x_n \leq 1, \forall n\end{aligned}$$

The above problem is a non-convex optimization problem, which we propose to solve using projected gradient descent. Our technique of obtaining $\frac{d\mathbf{x}_t}{d\lambda_t}$ is to differentiate through the steps of this PGD algorithm; this approach has been referred to as hyper or (sometimes) meta gradient in literature [1].

If PGD is used, then we have: starting with an initial strategy $\mathbf{x}^0 \in \mathbf{X}$ which is randomly generated, at each iteration of the gradient descent $i \geq 1$, given the current defender strategy $\mathbf{x}^{i-1}$, we update:

$$\mathbf{x}^{i'} = \mathbf{x}^{i-1} + \alpha \frac{\partial U^d(\mathbf{x}^{i-1}, \lambda_t)}{\partial \mathbf{x}^{i-1}} \tag{8}$$

Then the updated (possibly infeasible) strategy is projected back to the feasible region. We obtain a new feasible strategy $\mathbf{x}^i$ which is

[2]In [19], they propose a different approach to compute these gradient components in one-shot SSGs by exploiting the underlying characteristics of the defender's learning and patrolling (such as the uniqueness of the optimal solution). Their approach is applicable only when the Quantal Response model is used for predicting the attacker's behavior. On the other hand, our approach, in fact, can be applied for any differentiable behavior models of the attacker including SUQR and neural nets, etc.

an optimal solution of the following optimization problem:

$$\min_{\mathbf{x}} ||\mathbf{x} - \mathbf{x}^{i'}||_2 \tag{9}$$

$$\text{s.t.} \sum_n x_n \leq S \tag{10}$$

$$0 \leq x_n \leq 1, \forall n \tag{11}$$

The problem (9–11) is a convex optimization problem, which can be easily solved using any convex solver. Clearly, $\mathbf{x}^i$ is a function of $\mathbf{x}^{i'}$. Therefore, we have the gradient decomposition:

$$\frac{d\mathbf{x}^i}{d\lambda_t} = \frac{d\mathbf{x}^i}{d\mathbf{x}^{i'}} \cdot \frac{d\mathbf{x}^{i'}}{d\lambda_t} \tag{12}$$

In the following, we will show how to compute the two gradient components on the RHS of (12). First, we denote by $G(\mathbf{x}^{i-1}, \lambda_t) = \frac{\partial U^d(\mathbf{x}^{i-1},\lambda_t)}{\partial \mathbf{x}^{i-1}}$, which is a function of $(\mathbf{x}^{i-1}, \lambda)$. By taking the derivative on both side of the equation (8) with respect to λ_t, we obtain:

$$\frac{d\mathbf{x}^{i'}}{d\lambda_t} = \frac{d\mathbf{x}^{i-1}}{d\lambda_t} + \delta\frac{dG(\mathbf{x}^{i-1}, \lambda_t)}{d\lambda_t} = \frac{d\mathbf{x}^{i-1}}{d\lambda_t} + \alpha\left[\frac{\partial G}{\partial \lambda_t} + \frac{\partial G}{\partial \mathbf{x}^{i-1}} \cdot \frac{d\mathbf{x}^{i-1}}{d\lambda_t}\right]$$

$$\implies \frac{d\mathbf{x}^{i'}}{d\lambda_t} = \alpha\frac{\partial G}{\partial \lambda_t} + \left[\alpha\frac{\partial G}{\partial \mathbf{x}^{i-1}} + diag(\vec{1})\right] \cdot \frac{d\mathbf{x}^{i-1}}{d\lambda_t}$$

which show that we can compute the gradient $\frac{d\mathbf{x}^{i'}}{d\lambda_t}$ recursively.

Next, in order to compute the gradient $\frac{d\mathbf{x}^i}{d\mathbf{x}^{i'}}$, which is the gradient of the projected strategy $\mathbf{x}^i$ with respect to the gradient-based updated strategy $\mathbf{x}^{i'}$. For the sake of presentation, we abstractly reformulate the projection problem (9–11) as the following convex optimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{x}^{i'}) \tag{13}$$

$$\text{s.t. } A\mathbf{x} \leq b \tag{14}$$

where $f(\mathbf{x}, \mathbf{x}^{i'}) = ||\mathbf{x} - \mathbf{x}^{i'}||_2$ and $A = \begin{bmatrix} \vec{1}^T \\ diag(\vec{1}) \\ -diag(\vec{1}) \end{bmatrix}$ and $b = \begin{bmatrix} S \\ \vec{1} \\ \vec{0} \end{bmatrix}$ with $\vec{1}$ is an $N \times 1$ vector of all ones. Following the results presented in [4], since (13–14) is a convex optimization problem, we can apply the Implicit Function Theorem [12, 24] upon the KKT conditions [2] of this problem, to obtain the gradient $\frac{d\mathbf{x}^i}{d\mathbf{x}^{i'}}$, formulated as follows:

$$\begin{bmatrix} \nabla^2_{\mathbf{x}^i} f(\mathbf{x}^i, \mathbf{x}^{i'}) & \mathbf{A}^T \\ diag(\eta)\mathbf{A} & diag(\mathbf{A}\mathbf{x}^i - b) \end{bmatrix} \begin{bmatrix} \frac{d\mathbf{x}^i}{d\mathbf{x}^{i'}} \\ \frac{d\eta}{d\mathbf{x}^{i'}} \end{bmatrix} = -\begin{bmatrix} \frac{d\nabla_{\mathbf{x}^i} f(\mathbf{x}^i,\mathbf{x}^{i'})}{d\mathbf{x}^{i'}} \\ 0 \end{bmatrix} \tag{15}$$

$$\implies \begin{bmatrix} \frac{d\mathbf{x}^i}{d\mathbf{x}^{i'}} \\ \frac{d\eta}{d\mathbf{x}^{i'}} \end{bmatrix} = -\begin{bmatrix} \nabla^2_{\mathbf{x}^i} f(\mathbf{x}^i, \mathbf{x}^{i'}) & \mathbf{A}^T \\ diag(\eta)\mathbf{A} & diag(\mathbf{A}\mathbf{x}^i - b) \end{bmatrix}^{-1} \begin{bmatrix} \frac{d\nabla_{\mathbf{x}^i} f(\mathbf{x}^i,\mathbf{x}^{i'})}{d\mathbf{x}^{i'}} \\ 0 \end{bmatrix} \tag{16}$$

where η is the dual variable of with respect to $\mathbf{x}^i$.

Based on the above analysis, we present Algorithm 1 which computes the gradient $\frac{d\mathbf{x}_t}{d\lambda_t}$. Overall, Algorithm 1 runs *nRound*, each round find a local optimal strategy solution and its gradient with respect to λ_t. At each round *round*, Algorithm 1 starts by initializing a defender strategy $\mathbf{x}^0$. Then at each iteration i, the algorithm updates the defender's strategy as well as its corresponding gradient (lines (6–9)). The function $Project(\mathbf{x}^{i'}, \mathbf{X})$ returns the projected strategy and its gradient $(\mathbf{x}^i, \frac{d\mathbf{x}^i}{d\mathbf{x}^{i'}})$, which is based on the computation described in (13–16) This iteration process stops when the update does not increase the defender's utility (i.e., $\delta U \leq 0$). Finally, the optimal defender's strategy and its gradient is determined based on the maximum defender's utility over all the rounds (line (12–14)).

Algorithm 1: Compute the gradient $\frac{d\mathbf{x}_t}{d\lambda_t}$

```
1  Initialize $optU = -\infty$;
2  for $round = 1 \rightarrow nRound$ do
3      Initialize $\mathbf{x}^0$; $\delta U = +\infty$; $i = 0$;
4      while $\delta U > 0$ do
5          Update $i = i + 1$;
6          Compute $\mathbf{x}^{i'} = \mathbf{x}^{i-1} + \alpha\frac{\partial U^d(\mathbf{x}^{i-1},\lambda_t)}{\partial \mathbf{x}^{i-1}}$;
7          Compute $\frac{d\mathbf{x}^{i'}}{d\lambda_t} = \alpha\frac{\partial G}{\partial \lambda_t} + \left[\alpha\frac{\partial G}{\partial \mathbf{x}^{i-1}} + diag(\vec{1})\right] \cdot \frac{d\mathbf{x}^{i-1}}{d\lambda_t}$;
8          Compute $(\mathbf{x}^i, \frac{d\mathbf{x}^i}{d\mathbf{x}^{i'}}) = Project(\mathbf{x}^{i'}, \mathbf{X})$;
9          Compute $\frac{d\mathbf{x}^i}{d\lambda_t} = \frac{d\mathbf{x}^i}{d\mathbf{x}^{i'}} \cdot \frac{d\mathbf{x}^{i'}}{d\lambda_t}$;
10         Update $\delta U = U^d(\mathbf{x}^i, \lambda_t) - U^d(\mathbf{x}^{i-1}, \lambda_t)$;
11     if $optU < U^d(\mathbf{x}^i, \lambda_t)$ then
12         Update $optU = U^d(\mathbf{x}^i, \lambda_t)$;
13         Update $\mathbf{x}_t = \mathbf{x}^i$;
14         Update $\frac{d\mathbf{x}_t}{d\lambda_t} = \frac{d\mathbf{x}^i}{d\lambda_t}$;
15 Return $(\mathbf{x}_t, \frac{d\mathbf{x}_t}{d\lambda_t})$;
```

4.2 Compute partial derivative $\frac{\partial \lambda_t}{\partial \mathbf{z}_{t'}}$ where $t' < t$

The learning outcome, λ^t, is an optimal solution of:

$$\max_{\lambda \geq 0} \; logL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \lambda) \tag{17}$$

where $\mathbf{X}_{t-1} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}\}$ and $\mathbf{Z}_{t-1} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{t-1}\}$ are the defender's strategies and the attacker's attacks at previous time steps. Note that the defender's strategy at first time step $t = 1$, $\mathbf{x}_1$, is not determined based on any training data. The log-likelihood function $logL$ is concave in λ. Therefore, the problem (17) can be solved optimally using any optimization solver. If we use gradient descent to solve the above optimization problem, then we have: starting with some initial value of λ, denoted by λ^0, which is randomly generated, at each iteration i, given the current value λ^{i-1}, we update:

$$\lambda^i = \lambda^{i-1} + \alpha\frac{\partial logL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \lambda^{i-1})}{\partial \lambda^{i-1}} \tag{18}$$

We denote by $H(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \lambda^{i-1}) = \frac{\partial logL(\mathbf{X}_{t-1},\mathbf{Z}_{t-1},\lambda^{i-1})}{\partial \lambda^{i-1}}$. By taking the derivative of both sides of the above equation, we obtain:

$$\frac{\partial \lambda^i}{\partial \mathbf{z}_{t'}} = \frac{\partial \lambda^{i-1}}{\partial \mathbf{z}_{t'}} + \alpha\left[\sum_{t''=t'+1}^{t-1} \frac{\partial H}{\partial \mathbf{x}_{t''}} \cdot \frac{\partial \mathbf{x}_{t''}}{\partial \mathbf{z}_{t'}} + \frac{\partial H}{\partial \lambda^{i-1}} \cdot \frac{\partial \lambda^{i-1}}{\partial \mathbf{z}_{t'}} + \frac{\partial H}{\partial \mathbf{z}_{t'}}\right] \tag{19}$$

$$\frac{\partial \mathbf{x}_{t''}}{\partial \mathbf{z}_{t'}} = \frac{d\mathbf{x}_{t''}}{d\lambda_{t''}} \cdot \frac{\partial \lambda_{t''}}{\partial \mathbf{z}_{t'}}, t' + 1 \leq t'' \leq t - 1 \tag{20}$$

which shows that we can compute the gradient $\frac{\partial \lambda^i}{\partial \mathbf{z}_{t'}}$ can be computed recursively. Therefore, we present Algorithm 2 to compute the gradient $\frac{\partial \lambda_t}{\partial \mathbf{z}_{t'}}$ with $t' < t$. The inputs of Algorithm 2 include: (i) Previous defense strategies $\mathbf{X}_{t-1}$; (ii) Previous attacks $\mathbf{Z}_{t-1}$; and (iii) the gradients $\{\frac{\partial \lambda_{t''}}{\partial \mathbf{z}_{t'}}\}$ and $\{\frac{d\mathbf{x}_{t''}}{d\lambda_{t''}}\}$, for all $t'' > t'$ and $t'' < t$. Note that the gradient $\{\frac{d\mathbf{x}_{t''}}{d\lambda_{t''}}\}$ is computed based on Algorithm 1.

Algorithm 2: Compute the gradient $\frac{\partial \lambda_t}{\partial \mathbf{z}_{t'}}$ where $t' < t$

```
1  Input: $\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \{\frac{\partial \lambda_{t''}}{\partial \mathbf{z}_{t'}}\}$ , and $\{\frac{d\mathbf{x}_{t''}}{d\lambda_{t''}}\}$, for all $t > t'' > t'$
2  Initialize $\lambda^0$ and $\delta L = +\infty$;
3  while $\delta L > 0$ do
4      Update $i = i + 1$;
5      Compute $\lambda^i = \lambda^{i-1} + \alpha \frac{\partial logL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \lambda^{i-1})}{\partial \lambda^{i-1}}$;
6      if $\lambda^i < 0$ then
7          $\lambda^i = 0$;
8          $\frac{\partial \lambda^i}{\partial \mathbf{z}_{t'}} = 0$;
9      else
10         $\frac{\partial \lambda^i}{\partial \mathbf{z}_{t'}} = \frac{\partial \lambda^{i-1}}{\partial \mathbf{z}_{t'}} + \alpha \left[ \sum_{t''=t'+1}^{t-1} \frac{\partial H}{\partial \mathbf{x}_{t''}} \cdot \frac{d\mathbf{x}_{t''}}{d\lambda_{t''}} \cdot \frac{\partial \lambda_{t''}}{\partial \mathbf{z}_{t'}} + \frac{\partial H}{\partial \lambda^{i-1}} \cdot \frac{\partial \lambda^{i-1}}{\partial \mathbf{z}_{t'}} + \frac{\partial H}{\partial \mathbf{z}_{t'}} \right]$;
11     Update $\delta L = logL(\cdot, \cdot, \lambda^i) - logL(\cdot, \cdot, \lambda^{i-1})$;
12 Return ($\lambda_t = \lambda^i$, $\frac{\partial \lambda_t}{\partial \mathbf{z}_{t'}} = \frac{\partial \lambda^i}{\partial \mathbf{z}_{t'}}$) for all $t' < t$;
```

At each iteration i, Algorithm 2 updates the QR parameter as well as its gradient (λ^i, $\frac{\partial \lambda^i}{\partial \mathbf{z}_{t'}}$) based on (18–20). Lines (7–8), in particular, is the projection step which guarantees that the learning outcome to be greater than zero. This iteration process stops when the update does not increase the log-likelihood objective (i.e., $\delta L \leq 0$).

5 PRELIMINARY EXPERIMENT

In our experiments, we evaluate both solution quality and runtime performance of our proposed algorithm. We aim at analyzing the impact of the attacker's sequential manipulative attacks on both the defender and attacker's utility across the entire time horizon.

We generate game payoffs uniformly at random within the range $[0, 10]$ for the rewards and the range $[-10, 0]$ for the penalties of players at each target. We vary the number of targets, which is chosen from the set $\{8, 10, 12, 14, 16\}$. We also examine different resource-target ratios (i.e., $\frac{S}{N} = 0.3$ and $\frac{S}{N} = 0.5$), as well as different number of time steps (i.e., $T = 4$ and $T = 8$). In our games, the maximum number of attacks at each time step is limited to $K = 50$. Each of our data points is averaged over 30 game instances.

Our solution quality results are shown in Figures 2 and 3. In these figures, the x-axis represents the number of targets in the games while the y-axis refers to the attacker's utility (Figure 2) or the defender's utility (Figure 3) on average per time step. We compare two cases: *Manipulated* — the attacker plays manipulatively by following our algorithm; and *None* — the attacker plays myopic optimally at each time step. Figure 2 shows that the attacker gains a significant higher utility for manipulating its attacks at every step compared to the *None* case. In addition, by comparing between the

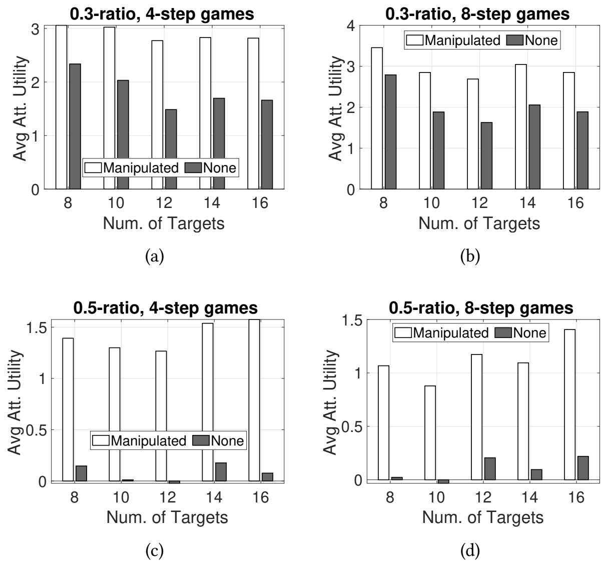


Figure 2: Attacker Utility Evaluation

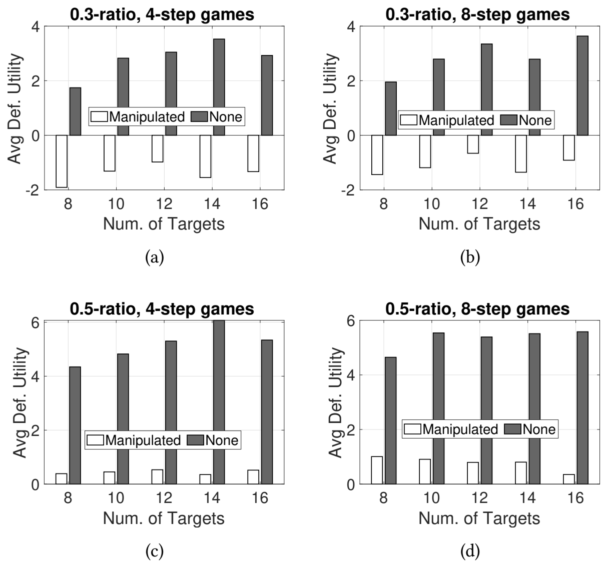
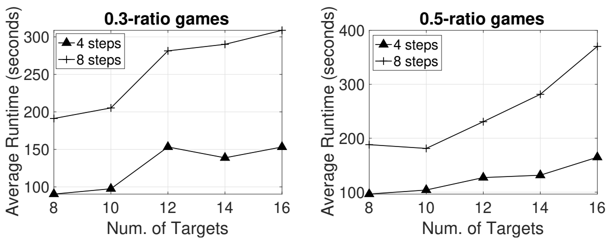


Figure 3: Defender Utility Evaluation

$\frac{S}{N} = 0.3$ and the $\frac{S}{N} = 0.5$ resource-target ratio cases (Figures 2(a)(b) versus Figures 2(c)(d)), the utility gain (i.e., the difference between *Manipulated* and *None*) that the attacker obtains is shown to increase when this ratio increases. This result makes sense since the defender's strategy space is expanded when we increases the ratio $\frac{S}{N}$. Thus the attacker has more flexibility to influence the defender's strategy choices, which results in a higher gain for the attacker. On the other hand, the utility of the attacker in both *Manipulated* and *None* decreases when $\frac{S}{N}$ increases. This is because the defender's coverage probability at each target increases when $\frac{S}{N}$ increases,



(a) (b)

Figure 4: Runtime Performance

which clearly lowers the attacker's expected utility at each target (the attacker's expected utility is a decreasing function of the defender's coverage probability at each target). Conversely, we observe opposite trends for the defender's utility (Figure 3). That is, the defender suffers a substantial utility loss (*Manipulated* versus *None*) as a result of the attacker's manipulation.

In both Figures 2 and 3, comparing between the 4-step and 8-step games, we find mixed results regarding the long-term impact of the attacker's manipulation. In particular, when the ratio is $\frac{S}{N} = 0.5$, Figures 2(c)(d) show that the attacker gains less average utility per time step in the 8-step games, indicating that the impact of the attacker's manipulation is lessen when increasing number of steps. However, in the case of ratio $\frac{S}{N} = 0.3$, that trend does not hold (Figures 2(a)(b)). The impact results on the defender side are also similar in the sense that the long-term impact of the attacker's manipulation fluctuates across different number of targets and resource-target ratios. Finally, the correlation between the players' utility and the number of targets is unclear.

Our evaluation on runtime performance is shown in Figure 4. The x-axis is the number of targets while the y-axis represents the runtime of our algorithm on average in seconds. Overall, Figure 4 shows that the algorithm's runtime gradually increases when the number of targets increases. The runtime reaches approximately 300 seconds in 16-target, 8-step games when the ratio is $\frac{S}{N} = 0.3$ (Figure 4(a)). On the other hand, it reaches approximately 370 seconds in the similar game setting but with $\frac{S}{N} = 0.5$ (Figure 4(b)).

6 SUMMARY

In this work, we study the problem of sequential manipulative attacks in multi-step SSGs in which the attacker intentionally changes its attack behavior to fool the defender's learning, and thus influence the defender's strategies towards its long-term benefit. We propose a new security game model which captures such manipulative attacks. We then develop a new algorithm which determines an optimal manipulative attack plan for the attacker. Our algorithm follows the SGD-based approach and leverages the ideas of hyper-parameter learning and implicit function theorem to estimate the gradient components required for the gradient step update in SGD. We provide preliminary empirical results which analyze the impact of the attacker's manipulation on both players' utility accumulated in the entire time horizon. The results clearly show a significant benefit for the attacker and loss for the defender.

Acknowledgement. This work was supported by MURI grant W911NF-20-1-0344 from the US Army Research Office.

REFERENCES

[1] Y. Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[2] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[3] G. Brown, M. Carlyle, D. Diehl, J. Kline, and K. Wood. A two-sided optimization for theater ballistic missile defense. *Operations Research*, 53(5):745–763, 2005.

[4] P. Donti, B. Amos, and J. Z. Kolter. Task-based end-to-end model learning in stochastic optimization. In *Neural Information Processing Systems*, 2017.

[5] F. Fang, T. H. Nguyen, R. Pickles, W. Y. Lam, G. R. Clements, B. An, A. Singh, M. Tambe, and A. Lemieux. Deploying paws: Field optimization of the protection assistant for wildlife security. In *IAAI-16*, 2016.

[6] J. Farrell and M. Rabin. Cheap talk. *Journal of Economic Perspectives*, 10(3):103–118, 1996.

[7] S. Gholami, A. Yadav, L. Tran-Thanh, B. Dilkina, and M. Tambe. Don't put all your strategies in one basket: Playing green security games with imperfect prior knowledge. In *AAMAS '19*, pages 395–403, 2019.

[8] Q. Guo, B. An, B. Bosansky, and C. Kiekintveld. Comparing strategic secrecy and Stackelberg commitment in security games. In *IJCAI*, 2017.

[9] K. Hendricks and R. P. McAfee. Feints. *Journal of Economics & Management Strategy*, 15(2):431–456, 2006.

[10] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *AISec*, 2011.

[11] D. Kar, B. Ford, S. Gholami, F. Fang, A. Plumptre, M. Tambe, M. Driciru, F. Wanyama, A. Rwetsiba, and M. Nsubaga. Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In *AAMAS '17*, 2017.

[12] S. G. Krantz and H. R. Parks. *The implicit function theorem: history, theory, and applications.* Springer Science & Business Media, 2012.

[13] D. Lowd and C. Meek. Adversarial learning. In *ACM SIGKDD*, 2005.

[14] D. Maclaurin, D. Duvenaud, and R. Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[16] D. McFadden et al. Conditional logit analysis of qualitative choice behavior. 1973.

[17] R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

[18] K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[19] T. H. Nguyen, A. Sinha, and H. He. Partial adversarial behavior deception in security games. In *IJCAI*, 2020.

[20] T. H. Nguyen, N. Vu, A. Yadav, and U. Nguyen. Decoding the imitation security game: Handling attacker imitative behavior deception. In *24th European Conference on Artificial Intelligence*, 2020.

[21] T. H. Nguyen, Y. Wang, A. Sinha, and M. P. Wellman. Deception in finitely repeated security games. In *AAAI-19*, 2019.

[22] 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 *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132, 2008.

[23] Z. Rabinovich, A. X. Jiang, M. Jain, and H. Xu. Information disclosure as a means to security. In *AAMAS '15*, pages 645–653, 2015.

[24] W. Rudin. *Principles of Mathematical Analysis.* McGraw - Hill Book C., 1986.

[25] 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 *AAMAS*, 2012.

[26] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe. Stackelberg security games: Looking beyond a decade of success. In *IJCAI*, pages 5494–5501, 2018.

[27] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang. Vital: Visual tracking via adversarial learning. In *IEEE CVPR*, 2018.

[28] M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned.* Cambridge University Press, 2011.

[29] H. Xu, Z. Rabinovich, S. Dughmi, and M. Tambe. Exploring information asymmetry in two-stage security games. In *AAAI*, pages 1057–1063, 2015.

[30] R. Yang, C. Kiekintveld, F. Ordonez, M. Tambe, and R. John. Improving resource allocation strategy against human adversaries in security games. In *IJCAI*, 2011.

[31] X. Zhang, X. Zhu, and L. Lessard. Online data poisoning attack, 2019.

[32] J. Zhuang, V. M. Bier, and O. Alagoz. Modeling secrecy and deception in a multi-period attacker-defender signaling game. *European Journal of Operational Research*, 203:409–418, 2010.