Behavioral Learning in Security Games: Threat of Multi-Step Manipulative Attacks

Thanh H. Nguyen[1], **Arunesh Sinha**[2]

[1]University of Oregon, [2]Rutgers University
thanhhng@cs.uoregon.edu, arunesh.sinha@rutgers.edu

Abstract

This paper studies the problem of multi-step manipulative attacks in Stackelberg security games, in which a clever attacker attempts to orchestrate its attacks over multiple time steps to mislead the defender's learning of the attacker's behavior. This attack manipulation eventually influences 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 learns the attacker's behavior and then designs a corresponding strategy only 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 determine the future learning and patrol planning of the defender. This paper has 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 extensive experimental results on the impact of such misleading attacks, showing a significant benefit for the attacker and loss for the defender.

1 Introduction

Stackelberg security games (SSGs) have been widely applied for solving many real-world problems in public safety and security, cybersecurity, and conversations (Pita et al. 2008; Tambe 2011; Shieh et al. 2012; Fang et al. 2016). In recent work in SSGs, machine learning-based techniques have been used for modeling and predicting the attacker's behavior based on collected attack data (Yang et al. 2011; Kar et al. 2017; Sinha et al. 2018; Gholami et al. 2019). For example, in PROTECT, Quantal Response was used to predict decision making of the attacker in the domain of ferry protection (Shieh et al. 2012). In addition, in the PAWS-related work, different models such as Quantal Response, SUQR, and SHARP, etc were used to capture the behavior of poachers (i.e., predicting where the poachers are likely to set trapping tools to catch wild animals) (Fang et al. 2016; Kar et al. 2017, 2015; Sinha et al. 2018; Gholami et al. 2019). These behavior models, after being trained, are used to determine 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 mislead the defender's learning (Nguyen et al. 2020; Nguyen, Sinha, and He 2020). Consequently, the learned adversary behavior model deviates from the true behavior model, and causes the defender to generate ineffective patrolling strategies, which benefits 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 significantly more long-term benefit as a result of the defender's deteriorated strategies. In this work, we focus on analyzing such manipulative attacks of the attacker.

The existing works in SSGs mainly study one-shot game scenarios in which the defender only learns the attacker behavior once and then commits to a single defense strategy afterward (Nguyen et al. 2020; Nguyen, Sinha, and He 2020). However, in many real-world domains such as wildlife protection, the defender and attacker often interact in a repeated manner (Fang et al. 2016). That is, at each time step, given historical attack and patrol data, the defender updates his model 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 continues until the end of the time horizon. In this multi-step interaction scenario, it is clear that the existing one-shot SSG studies 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 such 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 learning-patrolling process 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 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 (Bengio 2000), 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 an extensive experimental analysis on the impact of the attacker's attack manipulation on the accumulated utility of both players. We show that the attacker gains a substantially higher utility while 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 (Yang et al. 2011; Kar et al. 2017; Gholami et al. 2019). Several different models were proposed before, including Quantal Response (Yang et al. 2011), SUQR (Nguyen et al. 2013), and SHARP (Kar et al. 2015) models. These models enable the defender to predict boundedly rational decisions of human attackers such as poachers using historical attack data. For example, in wildlife protection, rangers can collect poaching signs such as snares during their patrols (Fang et al. 2016). These observations are then used to predict poaching activities in the future.

However, there is a rising concern about the vulnerability of these learning-patrolling methods in the presence of a deceptive attacker who intentionally maneuvers its attacks to *fool* the defender's learning. Previous work has demonstrated that weakness of the learning-patrolling methods in one-shot (sometimes Bayesian) SSGs (Gan et al. 2019; Nguyen et al. 2020; Nguyen, Sinha, and He 2020); our focus is on multiple steps. A multi-step 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 (Nguyen et al. 2019).

Our work is also related to adversarial learning in machine learning in the sense of attacking the training/testing data or interfering with the learning process. Poisoning attacks (i.e., altering the training data) are the most closely related to our work (Lowd and Meek 2005; Song et al. 2018; Huang et al. 2011; Madry et al. 2017; Zhang, Zhu, and Lessard 2019; Demontis et al. 2019; Biggio and Roli 2018; Papernot et al. 2018). Different attack methods were designed to deteriorate the performance of standard machine learning algorithms such as SVMs and neural nets, etc. Differentiating from this research line, in our problem, multi-step decision quality (which is measured via utilities of players) in terms of players' strategies is the ultimate objective of the players, rather than just the prediction accuracy.

Finally, in secrecy and deception in SSGs, previous work investigated situations in which information available to the defender and attacker is asymmetric (Guo et al. 2017; Xu et al. 2015; Rabinovich et al. 2015; Hendricks and McAfee 2006; Brown et al. 2005; Farrell and Rabin 1996; Zhuang, Bier, and Alagoz 2010). 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.

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 over a set of important targets $[N] = \{1, \ldots, N\}$ to protect these targets against an attacker. In one-shot SSGs, a pure strategy of the defender can be viewed as a *subset* of targets that can be covered by his security resources. A mixed strategy of the defender is a distribution over the defender's pure strategies. We consider generic SSGs in which the defender's mixed strategies can be represented as a marginal probability vector $\mathbf{x} = \{x_1, \ldots, x_N\}$ with resource constraints can be captured by a set of linear constraints $\mathbf{A}\mathbf{x} \leq b$. Here, $x_n \in [0, 1]$ is the marginal coverage of the defender at target n. We denote by $\mathbf{X} = \{\mathbf{x} : \mathbf{A}\mathbf{x} \leq b\}$ the set of all mixed strategies.

In SSGs, the players' payoff depends on which target the attacker 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_n^a while the defender gets a penalty of P_n^d. Conversely, if the defender is protecting n, the attacker gets a penalty $P_n^a < R_n^a$ and the defender obtains a reward $R_n^d > P_n^d$. 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:

$$U_n^d(x_n) = x_n R_n^d + (1 - x_n) P_n^d \quad (1)$$

$$U_n^a(x_n) = x_n P_n^a + (1 - x_n) R_n^a \quad (2)$$

A standard game-theoretic solution concept in SSGs is Strong Stackelberg Equilibrium (SSE) in which players play optimally against each other given that the attacker is aware of the defender strategy before taking any action.

Learning attacker behavior One of the very first behavior model used to predict the attacker behavior is Quantal Response, a well-known behavior model used in both behavioral economics and game theory (MCFADDEN 1973; McKelvey and Palfrey 1995; Yang et al. 2011). While an SSE considers a perfectly rational attacker, QR assumes a boundedly rational attacker who attacks each target n with a probability proportional to the attacker's expected utility at that target. Later on, SUQR, an extension of Quantal Response, which uses a linear combination of various domain features to reason about the attacker's behavior (Nguyen et al. 2013). Building upon the success of QR and SUQR, Kar et. al introduce a new model, named SHARP, that augments a two-parameter probability weighting function into the SUQR model to predict poachers' behavior in wildlife protection (Kar et al. 2015). Among all these models, the

final prediction of the attacker's behavior can be abstractly captured using the following soft-max function:

$$q_n(\mathbf{x}, \theta) = \frac{e^{f(x_n,\theta)}}{\sum_{n'} e^{f(x_{n'},\theta)}}$$

which is the probability the attacker attacks target n. The function $f(x_n, \theta)$ represents the behavior model used by the defender, indicating that the attacker's decision depends on the defender's strategy x_n (in addition to other domain features such as the attacker rewards and penalties, etc. which we omit from the formulation for the sake of presentation). Finally, $\theta \in \mathbb{R}^m$ is the model parameter vector.

4 Manipulative Attack Game Model

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 can be then extended to capture such security scenarios.

Formally, at each 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 distribution at time step $t' \leq t-1$ (i.e., $z_{t',n}$ is the number of times the attackers attacks target n in time step t'). The horizon is T time steps and we denote $[T] = \{1, \ldots, T\}$.

Defender's learning and patrolling

At each step t, the defender's strategy $\mathbf{x}_t$ follows a two-stage learning-patrolling process to determine his strategy at t:

- **Learning**: The defender optimizes the model parameter vector θ_t at step t based on $(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1})$, which is the result of the following minimization problem:

  $$\min_{\theta \in \Theta} L(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta)$$

  where $L(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta)$ is the defender's loss function and Θ is the set of possible values of θ.
- **Patrolling**: Given the learning outcome θ_t, the defender finds an optimal strategy $\mathbf{x}_t$ accordingly, which is an optimal solution of the following optimization problem:

  $$\max_{\mathbf{x} \in \mathbf{X}} U^d(\mathbf{x}, \theta_t)$$

  which maximizes the defender's utility with respect to the learned parameter θ_t, denoted by $U^d(\mathbf{x}, \theta_t)$, with:

  $$U^d(\mathbf{x}, \theta_t) = \sum\nolimits_n q_n(\mathbf{x}, \theta_t) U_n^d(x_n)$$

  where $q_n(\mathbf{x}, \theta_t)$ is the predicted probability the attacker attacks target n at step t and $U_n^d(x_n)$ is the defender's expected utility if the attacker actually attacks t.

At the first time step $t = 1$, in particular, the defender does not have any data. Therefore, the defender can choose a particular strategy $\mathbf{x}_1$ to play, such as the SSE strategy. This repeated learning-patrolling process has been used in the PAWS application for generating ranger patrols in the wildlife protection domain (Fang et al. 2016).

Attacker Manipulation

Since the defender relies on 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 bounded rational to mislead the defender. By acting in this manipulative way, the attacker suffers some immediate utility loss (for playing bounded rational) but would obtain a significant long term benefit as the result of its influence on the defender's patrolling strategies. The attacker's 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.

Our paper focuses on analyzing such manipulative attacks, assuming the attacker knows the defender's learning model. In real-world security domains, the attacker may not know the learning model used by the defender. In this case, the attacker can assume a certain behavior model and plan its deceptive attacks accordingly (this assumed model may be different from the behavior model used by the defender). Later in the experiment section, we will analyze the impact of the attacker's knowledge and model assumption on the attacker manipulation outcomes. In particular, we empirically show that in the black-box attack scenario, the attacker's manipulation designed based on a surrogate learning model (which may be different from the defender's actual model) is still highly effective in terms of significantly deteriorating the defender's behavior learning and patrol planning.

Formally, the problem of finding an optimal manipulative attack strategy can be represented as the following:

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

$$\text{s.t. } \theta_t \in \arg\min_{\theta \in \Theta} L(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta), \forall t \tag{4}$$

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

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

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:

$$U^a(\mathbf{x}_t, \mathbf{z}_t) = \sum\nolimits_n z_{t,n} U_n^a(x_{t,n})$$

which depends on the frequency the attacker would attack each target n at each step t, $z_{t,n}$, and the attacker's expected utility, $U_n^a(x_{t,n})$. Constraints (4–5) represent the two-stage learning-patrolling of the defender at each step t. Constraint (6) ensures that the attacker can only launch at most K attacks at each step. The constant K represents the attacker's limited capability in influencing the defender's learning.

5 Attack Manipulation Computation

Overall, (3–6) consists of multiple connected optimization levels. The decision on which targets and how frequently to attack at each step not only influences the utility outcome at

current step but also affects 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. The problem (3–6) is challenging to solve given that all optimization levels are inter-connected and each optimization level itself is non-convex.

Projected Gradient Descent. We propose to relax the attack variables $\mathbf{z}_t = \{z_{t,n}\}$ for all time steps t to be continuous and then apply the Projected Gradient Descent (PGD) approach to solve it. Essentially, starting with some initial values of $\mathbf{z}^0 = \{\mathbf{z}_1^0, \mathbf{z}_2^0, \ldots, \mathbf{z}_T^0\}$, PGD iteratively updates the values of these attack variables based on the gradient step. Denote by $F = \sum_t U^a(\mathbf{x}_t, \mathbf{z}_t)$ the attacker's accumulative utility across the entire time horizon, at each iteration i of the PGD, given the current estimation $\mathbf{z}^{i-1} = \{\mathbf{z}_1^{i-1}, \mathbf{z}_2^{i-1}, \ldots, \mathbf{z}_T^{i-1}\}$, the gradient update step on attack variables is determined as follows:

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

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 $\mathbf{Z} = \{\mathbf{z} : \sum_n z_{t,n} \leq K, z_{t,n} \geq 0, \forall t, n\}$. Note that $\mathbf{z} \in \mathbf{Z}$ is a vector of length $T \times N$ where T is the number of steps and N is the number of targets in the game. This projection step is done by finding the closest feasible point in $\mathbf{Z}$ to the current value $\mathbf{z}^i$, which is formulated as follows:

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

which is a convex optimization problem and thus can be solved optimally using standard solvers. This update process continues until convergence, where convergence means that the update does not improve the attacker utility in Eq. (3). Once converged, we obtain a local optimal solution of (3–6). By running the PGD 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 attacker accumulated utility among the local optimal ones.

Technical Challenge. The main technical issue of apply PGD is computing the gradients required for PGD. Essentially, the core of PGD is to compute the gradient of the attacker utility at a value $\mathbf{z}$ of the attack variables:

$$\frac{dF}{d\mathbf{z}} = \sum_t \frac{dU^a(\mathbf{x}_t, \mathbf{z}_t)}{d\mathbf{z}}$$
$$= \sum_t \sum_n \frac{dz_{t,n}}{d\mathbf{z}} U_n^a(x_{t,n}) + z_{t,n}(P_n^a - R_n^a)\frac{dx_{t,n}}{d\mathbf{z}}$$

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, $\frac{dz_{t,n}}{d\mathbf{z}} = \left\{\frac{\partial z_{t,n}}{\partial z_{t',n'}}\right\}_{(t',n') \in [T] \times [N]}$, which is straightforwardly determined as follows:

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

The second component is $\frac{dx_{t,n}}{d\mathbf{z}} = \left\{\frac{\partial x_{t,n}}{\partial z_{t',n'}}\right\}_{(t',n') \in [T] \times [N]}$, 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 as:

$$\frac{\partial x_{t,n}}{\partial z_{t',n'}} = \sum_j \frac{\partial x_{t,n}}{\partial \theta_{t,j}} \cdot \frac{\partial \theta_{t,j}}{\partial z_{t',n'}}, \forall t' < t \text{ and } \frac{\partial x_{t,n}}{\partial z_{t',n'}} = 0, \forall t' \geq t$$

where $\theta_{t,j}$ is the j^{th} component of the parameter vector θ_t at step t. Next, the challenge is that, even though $x_{t,n}$ depends on θ_t and θ_t depends on $z_{t',n'}$, we do not have a closed form of $x_{t,n}$ and θ_t as a function of θ_t and $z_{t',n'}$, respectively.

To address this, we take inspiration from hyper-parameter learning (Maclaurin, Duvenaud, and Adams 2015); for utilizing hyper-parameter learning, the attacker has to assume knowledge of the computations steps and model used by defender to solve the problem in Eq. (4) and (5). The computational steps can be any differentiable steps that leads to the optimally solving problem in Eq. (4) and (5). Here we take the computation by the defender to be a projected gradient descent approach. We show later in our experiments that even when the attacker's assumption about the computation steps is different from the defender's actual model, the attacker still gains a significant benefit for its manipulation. Next, we elaborate methods to estimate the gradient components $\frac{d\mathbf{x}_t}{d\theta_t} = \left\{\frac{\partial x_{t,n}}{\partial \theta_{t,j}}\right\}$ and $\frac{d\theta_t}{d\mathbf{z}_{t'}} = \left\{\frac{\partial \theta_{t,j}}{\partial z_{t',n'}}\right\}$ for all $t' < t$.[1]

Computing Gradient of Defender Strategy

Overall, computing the gradient $\frac{d\mathbf{x}_t}{d\theta_t}$ is not straightforward since we don't have a closed-form representation of the defender's strategy $\mathbf{x}_t$ as a function of the model parameter θ_t. Essentially, the defender strategy at time step t, $\mathbf{x}_t$, is an optimal solution of the following optimization problem:

$$\max_{\mathbf{x} \in \mathbf{X}} U^d(\mathbf{x}, \theta_t)$$

The above problem is in general a non-convex optimization problem. Our technique of obtaining $\frac{d\mathbf{x}_t}{d\theta_t}$ is to differentiate through the steps of the defender's PGD approach to solve this problem; this approach is related to the hyper or (sometimes) meta gradient approach in literature (Bengio 2000).

Essentially, the defender starts with an initial strategy $\mathbf{x}^{0,\text{proj}} \in \mathbf{X}$ which is randomly generated within the feasible region $\mathbf{X} = \{\mathbf{x} : Ax \leq b\}$ of the defender's patrol strategies. At each iteration i of the defender's PGD, given the current defender strategy $\mathbf{x}^{i-1,\text{proj}}$, the update is as follows:

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

Then the updated (possibly infeasible) strategy is projected back to the feasible region. We obtain a new feasible strategy $\mathbf{x}^{i,\text{proj}}$ which is an optimal solution of:

$$\mathbf{x}^{i,\text{proj}} \in \arg\min_{\mathbf{x} \in \mathbf{X}} ||\mathbf{x} - \mathbf{x}^i||_2 \tag{9}$$

[1] In (Nguyen, Sinha, and He 2020), they propose a different approach to compute these gradient components in one-shot SSGs by exploiting intrinsic properties of the defender's learning. Their approach is applicable only when Quantal Response is used. Our approach can be applied for any differentiable behavior models.

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

Initialize $optU = -\infty$;
for $round = 1 \rightarrow nRound$ **do**
  Initialize $\mathbf{x}^{0,\text{proj}}$; $\delta U = +\infty$; $i = 0$;
  while $\delta U > 0$ **do**
    Update $i = i + 1$;
    Compute $\mathbf{x}^i$ and $\mathbf{x}^{i,\text{proj}}$ according to (8–9);
    Compute $\frac{d\mathbf{x}^{i,\text{proj}}}{d\theta_t}$ based on (10) and Prop. 1;
    Update $\delta U = U^d(\mathbf{x}^{i,\text{proj}}, \theta_t) - U^d(\mathbf{x}^{i-1,\text{proj}}, \theta_t)$;
  if $optU < U^d(\mathbf{x}^{i,proj}, \theta_t)$ **then**
    Update $optU = U^d(\mathbf{x}^{i,\text{proj}}, \theta_t)$; $\frac{d\mathbf{x}_t}{d\theta_t} = \frac{d\mathbf{x}^{i,\text{proj}}}{d\theta_t}$;

The problem (9) is a convex optimization problem, which can be easily solved using any convex solver. Clearly, $\mathbf{x}^{i,\text{proj}}$ is a function of $\mathbf{x}^i$. We thus have the gradient decomposition:

$$\frac{d\mathbf{x}^{i,\text{proj}}}{d\theta_t} = \frac{d\mathbf{x}^{i,\text{proj}}}{d\mathbf{x}^i} \cdot \frac{d\mathbf{x}^i}{d\theta_t} \tag{10}$$

We present Proposition 1 which shows the computation of the two gradient components on the RHS of (10).

Proposition 1. *Denote by* $G(\mathbf{x}^{i-1,proj}, \theta_t) = \frac{\partial U^d(\mathbf{x}^{i-1,proj}, \theta_t)}{\partial \mathbf{x}^{i-1,proj}}$. *Denote by* J_{G,θ_t} *the sub-matrix of the Jacobian* J_G *of* G *that is formed by partial derivatives w.r.t.* $\theta_{t,j}$. *Similarly,* $J_{G,\mathbf{x}^{i-1,proj}}$ *is the sub-matrix of* J_G *that is formed by partial derivatives w.r.t.* $\mathbf{x}^{i-1,proj}$. *Note that then,* $J_G = [J_{G,\mathbf{x}^{i-1,proj}} \mid J_{G,\theta_t}]$. *Then, the gradient components in (10) are computed as follows:*

$$\frac{d\mathbf{x}^i}{d\theta_t} = \alpha J_{G,\theta_t} + \left[\alpha J_{G,\mathbf{x}^{i-1,proj}} + diag(\vec{1})\right] \cdot \frac{d\mathbf{x}^{i-1,proj}}{d\theta_t} \tag{11}$$

$$\begin{bmatrix} \frac{d\mathbf{x}^{i,proj}}{d\mathbf{x}^i} \\ \frac{d\eta}{d\mathbf{x}^i} \end{bmatrix} = -\begin{bmatrix} \nabla^2_{\mathbf{x}^{i,proj}} ||\mathbf{x}^i - \mathbf{x}^{i,proj}||_2 & \mathbf{A}^T \\ diag(\eta)\mathbf{A} & diag(\mathbf{A}\mathbf{x}^{i,proj} - b) \end{bmatrix}^{-1} \tag{12}$$
$$\cdot \begin{bmatrix} \frac{d\nabla_{\mathbf{x}^{i,proj}} ||\mathbf{x}^i - \mathbf{x}^{i,proj}||_2}{d\mathbf{x}^i} \\ 0 \end{bmatrix}$$

where η *is the dual variable with respect to the projected strategy* $\mathbf{x}^{i,proj}$ *in the optimization problem (9).*

All of our proofs are in the appendix. We now present Algorithm 1 which computes $\frac{d\mathbf{x}_t}{d\theta_t}$. Overall, we run $nRound$, each round finds a local optimal strategy solution and its gradient w.r.t θ_t. At each round, Algorithm 1 starts by initializing a defender strategy $\mathbf{x}^{0,\text{proj}}$. Then at each iteration i, the algorithm updates the defender's strategy and its corresponding gradient. This 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 his maximum utility over all rounds (line (10)).

Compute Gradient of Learning Outcome

In general, computing the gradient $\frac{d\theta_t}{d\mathbf{z}_{t'}}$ is challenging since the learning outcome θ_t depends on the entire attack history before t. More specifically, the learning outcome, θ_t, is an optimal solution of the following optimization problem:

$$\min_{\theta \in \Theta} \; L(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta) \tag{13}$$

which minimizes the defender's learning loss. $\mathbf{X}_{t-1} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}\}$ 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. We consider the set parameter feasible region Θ to be represented by a set of linear constraints $\Theta = \{\theta : \mathbf{C} \cdot \theta \leq \mathbf{D}\}$. This optimization problem is generally known to be non-convex. Similar to the computation of the gradient of the defender strategy, we also apply the a recursive approach to solve this problem and differentiate through the gradient steps. That is, we start with some initial value of θ, denoted by $\theta^{0,\text{proj}}$, which is randomly generated within the feasible region Θ. Then at each iteration i, given the current value θ^{i-1}, we update:

$$\theta^i = \theta^{i-1,\text{proj}} - \alpha \frac{dL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,\text{proj}})}{d\theta^{i-1,\text{proj}}} \tag{14}$$

Then the updated (possibly infeasible) learning outcome θ^i is projected back to the feasible region Θ. We obtain a new feasible learning outcome $\theta^{i,\text{proj}}$ which is the optimal solution of the following minimization problem:

$$\theta^{i,\text{proj}} \in \arg\min_{\theta \in \Theta} ||\theta - \theta^i||_2 \tag{15}$$

We thus have the following gradient decomposition:

$$\frac{d\theta^{i,\text{proj}}}{d\mathbf{z}_{t'}} = \frac{d\theta^{i,\text{proj}}}{d\theta^i} \cdot \frac{d\theta^i}{d\mathbf{z}_{t'}} \tag{16}$$

Observing that the problem (15) is similar to problem Eq. (9), we can thus compute the gradient $\frac{d\theta^{i,\text{proj}}}{d\theta^i}$ similarly to $\frac{d\mathbf{x}^{i,\text{proj}}}{d\mathbf{x}^i}$. On the other hand, the gradient $\frac{d\theta^i}{d\mathbf{z}_{t'}}$ is challenging to compute since θ^i depends on the entire history $(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1})$. We present our Proposition 2 which shows that this gradient component can be computed recursively according to time steps.

Proposition 2. *Let* $H(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,proj}) = \frac{dL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,proj})}{d\theta^{i-1,proj}}$. *Let* J_H *be the Jacobian of* H. *Let* $J_{H,x_{t''}}$ *be the part of the Jacobian restricted to partial derivatives w.r.t.* $x_{t''}$ *(and similar for other variables* $z_{t'}, \theta^{i-1,proj}$*). The gradient component,* $\frac{d\theta^i}{d\mathbf{z}_{t'}}$, *can be computed recursively, as follows:*

$$\frac{d\theta^i}{d\mathbf{z}_{t'}} = \frac{d\theta^{i-1,proj}}{d\mathbf{z}_{t'}} \tag{17}$$
$$- \alpha \left[\sum_{t''=t'+1}^{t-1} J_{H,x_{t''}} \cdot \frac{d\mathbf{x}_{t''}}{d\mathbf{z}_{t'}} + J_{H,z_{t'}} + J_{H,\theta^{i-1,proj}} \cdot \frac{d\theta^{i-1,proj}}{d\mathbf{z}_{t'}} \right]$$

$$\text{where } \frac{d\mathbf{x}_{t''}}{d\mathbf{z}_{t'}} = \frac{d\mathbf{x}_{t''}}{d\theta_{t''}} \cdot \frac{d\theta_{t''}}{d\mathbf{z}_{t'}}, t' + 1 \leq t'' \leq t - 1 \tag{18}$$

Therefore, in order to compute all the derivatives $\frac{d\theta_t}{d\mathbf{z}_{t'}}$ for all $t' < t$, we can recursively call a modified version of Algorithm 1 for $t = 2, \ldots, T$ and $t' = 1, \ldots, t-1$. The

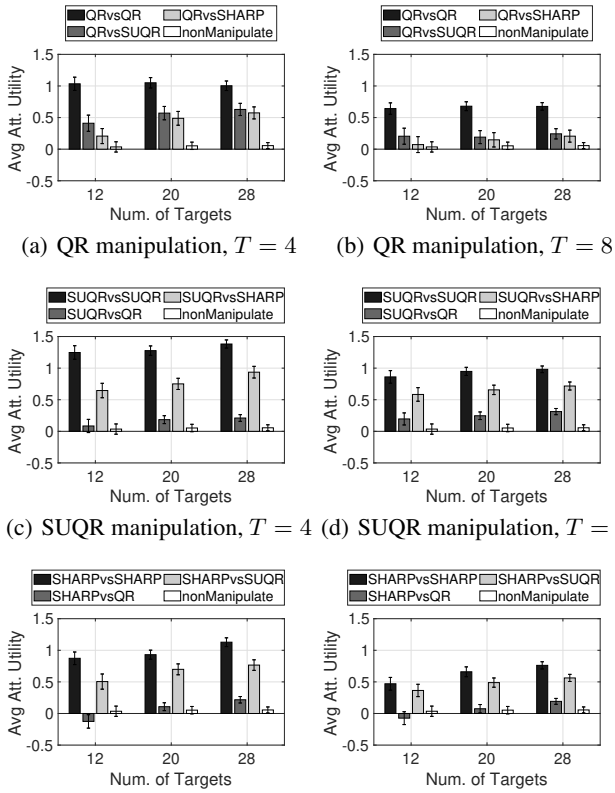
inputs of this modified algorithm include: (i) previous defense strategies $\mathbf{X}_{t-1}$; (ii) previous attacks $\mathbf{Z}_{t-1}$; and (iii) the derivatives $\left\{\frac{d\theta_{t''}}{d\mathbf{z}_{t'}}\right\}$ and $\left\{\frac{d\mathbf{x}_{t''}}{d\theta_{t''}}\right\}$, for all $t'' > t'$ and $t'' < t$. Note that the derivative $\left\{\frac{d\mathbf{x}_{t''}}{d\theta_{t''}}\right\}$ is computed based on Algorithm 1. The full algorithm is stated in the appendix.

6 Experiments

Our experiments are conducted on a High Performance Computing (HPC) cluster, with dual E5-2690v4 (28 cores) processors and 128 GB memory. We use Matlab to implement our algorithms. While the attacker assumes the defender uses PGD for his computation, the defender in our experiments actually uses the interior-point method, a well known optimization method. Our goal is to show that even when the defender uses a different optimization technique (i.e., interior-point method), the attack data manipulation computed based on the gradient descent assumption remains highly effective. Further, we investigate the impact of the attacker's incorrect knowledge of the defender's learning models on its manipulation outcomes. We analyze the impact of the attacker's manipulation on both players' utility (which is their ultimate goal) over the entire time horizon.

Security Game Generation. To analyze average performance, we generate multiple games with payoffs randomly within $[0, 10]$ for the rewards and $[-10, 0]$ for the penalties of players at each target, using covariance games from Gambit (McKelvey, McLennan, and Turocy 2014). This is the commonly-used approach for generating games in security game literature (Tambe 2011). Our work aims at examining the impact of attack manipulation in various security settings. The covariance value r in game simulation of Gambit allows us to govern the correlation between the attacker and defender's payoffs. In particular, when $r = -1$, the games are zero-sum which means the players' utility objectives are completely opposite. When $r = 0$, the players' payoffs are uncorrelated. We use 10 game instances for each value of $r \in \{-1.0, -0.8, -0.6, -0.4, -0.2, 0\}$ (60 games in total). In our games, the maximum number of attacks at each time step is limited to $K = 50$. Each of data points is averaged over 60 games. Next, we highlight important results.

Attacker Behavior Models. We consider three different behavior models: QR (Yang et al. 2011), SUQR (Nguyen et al. 2013), and SHARP (Kar et al. 2015), with an increasing order of model complexity (i.e., QR is the simplest model while SHARP is the most complex among the three). These three models have been validated and applied in both human subject experiments and real-world domains such as ferry protection and wildlife security. In addition, these three models are generic, which are suitable for various security settings, including the standard security game setting used in our evaluation. Note that, beside these three models, there are other models which were proposed to predict poacher behavior in the wildlife setting. However, these specific models exploit intrinsic properties of the wildlife domain. We thus don't include these models in our experiments.



(a) QR manipulation, $T = 4$ (b) QR manipulation, $T = 8$

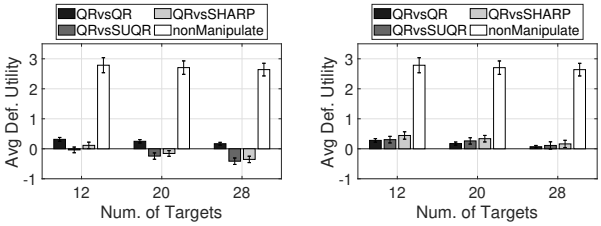
(c) SUQR manipulation, $T = 4$ (d) SUQR manipulation, $T = 8$

(e) SHARP manipulation, $T = 4$ (f) SHARP manipulation, $T = 8$

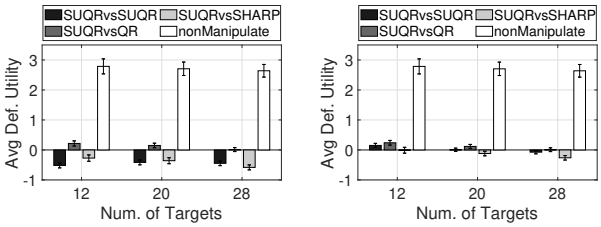
Figure 1: Attacker Utility Evaluation

Evaluation Results As mentioned previously, the impact of the attacker's manipulative attacks depends on the attacker's knowledge of the defender's learning model. In our experiments, we analyze nine difference manipulation scenarios; each scenario corresponds to a different pair of learning models (a combination of the attacker's assumption of the defender's learning model and the actual model used by the defender). For example, `QRvsSUQR` refers to the scenario in which (i) the attacker assumes the defender uses Quantal Response; and (ii) the defender actually chooses SUQR. The other scenarios are interpreted similarly. Our results on players' utility are shown in Figures 4 and 2. In each figure, the y-axis is either the attacker or the defender's utility outcomes which are averaged over 60 different games. The x-axis is the number of targets in the games. In addition, `nonManipulate` refers to the case when the attacker is not manipulative at all (i.e., both players plays the SSE strategies). For a fair comparison between different number of time steps, we show the averaged utility return per step.

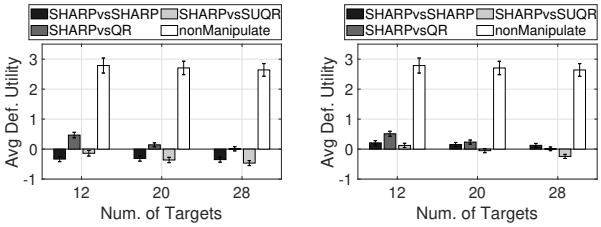
Overall, Figure 4 shows that the attacker gains a significantly higher utility for manipulating its attacks, especially when the attacker knows the defender's behavior model (`QRvsQR`, `SUQRvsSUQR` and `SHARPvsSHARP` versus `nonManipulate`). When the attacker optimizes its manipulative attacks w.r.t a behavior model not actually used by the defender, the impact of the attacker's manipulation is much



(a) QR manipulation, $T = 4$ (b) QR manipulation, $T = 8$



(c) SUQR manipulation, $T = 4$ (d) SUQR manipulation, $T = 8$



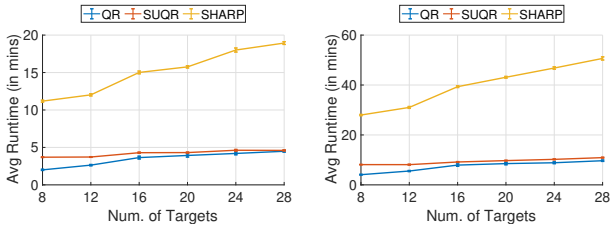
(e) SHARP manipulation, $T = 4$ (f) SHARP manipulation, $T = 8$

Figure 2: Defender Utility Evaluation

less severe compared to the case of a known behavior model. But in most cases, the attacker's utility for playing manipulatively is still substantially higher than `nonManipulate`. For example, in Figure 4(a), when the number of targets is 12, the attacker's average utility is 1.03 in `QRvsQR` while its utility reduces to 0.41 and 0.20 in `QRvsSUQR` and `QRvsSHARP`, respectively. Yet, it is still significantly higher than `nonManipulate` with the utility of 0.03.

There is a notable case in which the attacker suffers loss when it assumes the SHARP model (the most complex one among the three model) while the defender actually uses QR (the simplest model) (Figure 4(e)(f) with 12-target games and `SHARPvsQR` versus `nonManipulate`). Indeed, an over-parameterized complex model generally tends to be more sensitive towards noise and has been shown to a poor choice for a surrogate model in transferability of attacks in machine learning (Demontis et al. 2019). Thus, when the learning model is unknown, it would be beneficial for the attacker to use a simple behavior model.

Next, we examine the long-term benefit of the attacker's deception by comparing its utility between the 4-step games and the 8-step games (Figures 4(a)(c)(e) versus Figures 4(b)(d)(f)). For a fair comparison, we show results of the average utility return per step of the attacker. In these figures, we observe that when the number of time steps increases (8 versus 4 steps), the average utility return per step



(a) 4 steps (b) 8 steps

Figure 3: Runtime Performance

for the attacker is reduced. This result indicates that the attacker's deception has less impact on its benefit in a longer-term. This result is reasonable since the attacker has to trade off between the deception benefit (for misleading the defender's learning and patrolling) and the immediate utility loss for playing boundedly rational at every time step. And, in a long run, manipulative attacks at later time steps would have less impact on the defender's learning outcome.

Regarding the defender's utility (Figure 2), regardless of whether the attacker knows the defender's choice of a learning behavior model or not, the defender suffers a significant loss in utility in the presence of the attacker's manipulation (compared to the `nonManipulate` case). This result clearly shows that when the defender relies on attacker behavior learning, the quality of his strategy outcome is extremely vulnerable to the manipulative attacks of a clever attacker.

Finally, we examine the runtime performance of our proposed algorithm. The result is shown in Figure 3 in which the y-axis is the average runtime in minutes. Each of our data points is computed based on aggregated 5 rounds of the PGD process; each round consists of approximately 30 iterations of gradient descent update until reaching a local optimal solution. Note that each iteration involves multiple optimization components (i.e., training the behavior model and computing an optimal strategy at each time step). Despite the complex computation, Figure 3 shows that the runtime increases linearly in the number of targets, suggesting our method can be scaled up for large games.

7 Summary

This work investigates sequential attack manipulation in multi-step security games. We formulate new gradient based algorithms to compute an optimal attack plan for the attacker, tackling the computational challenge due to multiple inter-connected optimization components across the entire time horizon. Our experiments in various game settings show that the defender's data-based patrol strategies become extremely vulnerable to the attacker's manipulation of attack data, regardless of whether the attacker knows the defender's learning model or not. Our future research goal is to design effective learning-planning solutions for the defender in security games that are resilient to such attacker manipulation.

Acknowledgement. Dr. Nguyen was supported by grant W911NF-20-1-0344 from the US Army Research Office.

References

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

Biggio, B.; and Roli, F. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84: 317–331.

Boyd, S.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

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

Demontis, A.; Melis, M.; Pintor, M.; Jagielski, M.; Biggio, B.; Oprea, A.; Nita-Rotaru, C.; and Roli, F. 2019. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 321–338.

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

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

Gan, J.; Xu, H.; Guo, Q.; Tran-Thanh, L.; Rabinovich, Z.; and Wooldridge, M. 2019. Imitative follower deception in stackelberg games. In *ACM EC*, 639–657.

Gholami, S.; Yadav, A.; Tran-Thanh, L.; Dilkina, B.; and Tambe, M. 2019. Don't Put All Your Strategies in One Basket: Playing Green Security Games with Imperfect Prior Knowledge. In *AAMAS '19*, 395–403.

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

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

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

Kar, D.; Fang, F.; Delle Fave, F.; Sintov, N.; and Tambe, M. 2015. " A Game of Thrones" When Human Behavior Models Compete in Repeated Stackelberg Security Games. In *AAMAS-15*, 1381–1390.

Kar, D.; Ford, B.; Gholami, S.; Fang, F.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; and Nsubaga, M. 2017. Cloudy with a Chance of Poaching: Adversary Behavior Modeling and Forecasting with Real-World Poaching Data. In *AAMAS '17*.

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

Lowd, D.; and Meek, C. 2005. Adversarial learning. In *ACM SIGKDD*.

Maclaurin, D.; Duvenaud, D.; and Adams, R. 2015. Gradient-based hyperparameter optimization through reversible learning. In *ICML-15*, 2113–2122. PMLR.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083.

MCFADDEN, D. 1973. Conditional logit analysis of qualitative choice behavior. *Frontiers in Econometrics*, 105–142.

McKelvey, R. D.; McLennan, A. M.; and Turocy, T. L. 2014. Gambit: Software Tools for Game Theory, Version 15.1.1. http://www.gambit-project.org.

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

Nguyen, T.; Yang, R.; Azaria, A.; Kraus, S.; and Tambe, M. 2013. Analyzing the effectiveness of adversary modeling in security games. In *AAAI-13*, volume 27.

Nguyen, T. H.; Sinha, A.; and He, H. 2020. Partial Adversarial Behavior Deception in Security Games. In *IJCAI*.

Nguyen, T. H.; Vu, N.; Yadav, A.; and Nguyen, U. 2020. Decoding the Imitation Security Game: Handling Attacker Imitative Behavior Deception. In *ECAI-20*.

Nguyen, T. H.; Wang, Y.; Sinha, A.; and Wellman, M. P. 2019. Deception in Finitely Repeated Security Games. In *AAAI-19*.

Papernot, N.; McDaniel, P.; Sinha, A.; and Wellman, M. P. 2018. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, 399–414. IEEE.

Pita, J.; Jain, M.; Marecki, J.; Ordóñez, F.; Portway, C.; Tambe, M.; Western, C.; Paruchuri, P.; and Kraus, S. 2008. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *AAMAS: industrial track*, 125–132.

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

Shieh, E.; An, B.; Yang, R.; Tambe, M.; Baldwin, C.; DiRenzo, J.; Maule, B.; and Meyer, G. 2012. PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States. In *AAMAS*.

Sinha, A.; Fang, F.; An, B.; Kiekintveld, C.; and Tambe, M. 2018. Stackelberg Security Games: Looking Beyond a Decade of Success. In *IJCAI*, 5494–5501.

Song, Y.; Ma, C.; Wu, X.; Gong, L.; Bao, L.; Zuo, W.; Shen, C.; Lau, R. W.; and Yang, M.-H. 2018. VITAL: VIsual Tracking via Adversarial Learning. In *IEEE CVPR*.

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

Xu, H.; Rabinovich, Z.; Dughmi, S.; and Tambe, M. 2015. Exploring Information Asymmetry in Two-Stage Security Games. In *AAAI-15*, 1057–1063.

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

Zhang, X.; Zhu, X.; and Lessard, L. 2019. Online Data Poisoning Attack. arXiv:1903.01666.

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

Appendix

Proof of Proposition 1

Proof. Essentially, Eq. (11) is derived by differentiating both sides of Eq. (8). And Eq. (12) is the result from applying the Implicit Function Theorem (Krantz and Parks 2012) upon the KKT conditions (Boyd and Vandenberghe 2004) for the convex problem given by Eq. (9). Here, η is the dual variable of $\mathbf{x}^{i,\text{proj}}$. We denote by $G(\mathbf{x}^{i-1,\text{proj}}, \theta_t) = \frac{\partial U^d(\mathbf{x}^{i-1,\text{proj}}, \theta_t)}{\partial \mathbf{x}^{i-1,\text{proj}}}$, which is a function of $(\mathbf{x}^{i-1,\text{proj}}, \theta_t)$. Recall the notation $J_G = [J_{G,\mathbf{x}^{i-1,\text{proj}}} \mid J_{G,\theta_t}]$. By taking the derivative on both side of Eq. (8) with respect to θ_t, we obtain the gradient computation as follows:

$$\begin{aligned}
\frac{d\mathbf{x}^i}{d\theta_t} &= \frac{d\mathbf{x}^{i-1,\text{proj}}}{d\theta_t} + \alpha \frac{dG(\mathbf{x}^{i-1,\text{proj}}, \theta_t)}{d\theta_t} \\
&= \frac{d\mathbf{x}^{i-1,\text{proj}}}{d\theta_t} + \alpha \frac{dG(\mathbf{x}^{i-1,\text{proj}}, \theta_t)}{d(\mathbf{x}^{i-1,\text{proj}}, \theta_t)} \cdot \frac{d(\mathbf{x}^{i-1,\text{proj}}, \theta_t)}{d\theta_t}) \\
&= \frac{d\mathbf{x}^{i-1,\text{proj}}}{d\theta_t} + \alpha \left[J_{G,\theta_t} + J_{G,\mathbf{x}^{i-1,\text{proj}}} \cdot \frac{d\mathbf{x}^{i-1,\text{proj}}}{d\theta_t} \right] \\
\implies \frac{d\mathbf{x}^i}{d\theta_t} &= \alpha J_{G,\theta_t} + \left[\alpha J_{G,\mathbf{x}^{i-1,\text{proj}}} + diag(\vec{1}) \right] \cdot \frac{d\mathbf{x}^{i-1,\text{proj}}}{d\theta_t}
\end{aligned}$$

which show that we can compute the gradient $\frac{d\mathbf{x}^i}{d\theta_t}$ recursively according to the gradient ascent step.

Next, we will describe the computation of $\frac{d\mathbf{x}^{i,\text{proj}}}{d\mathbf{x}^i}$, which is the gradient of the projected strategy $\mathbf{x}^{i,\text{proj}}$ with respect to the gradient-based updated strategy $\mathbf{x}^i$. Note that the projection problem (9) is a convex optimization problem:

$$\min_{\mathbf{x}} ||\mathbf{x} - \mathbf{x}^i||_2 \tag{19}$$

$$\text{s.t. } A\mathbf{x} \leq b \tag{20}$$

By applying the the KKT conditions (Boyd and Vandenberghe 2004) to this optimization problem, we obtain that the projected strategy $\mathbf{x}^{i,\text{proj}}$ satisfies:

$$\begin{aligned}
&\nabla_{\mathbf{x}^{i,\text{proj}}} ||\mathbf{x}^{i,\text{proj}} - \mathbf{x}^i||_2 + \eta \nabla_{\mathbf{x}^{i,\text{proj}}} (\mathbf{A}\mathbf{x}^{i,\text{proj}} - b) = 0 \\
&\eta \left(\mathbf{A}\mathbf{x}^{i,\text{proj}} - b \right) = 0
\end{aligned}$$

where η is the dual variable with respect to $\mathbf{x}^{i,\text{proj}}$. We can thus apply the Implicit Function Theorem (Krantz and Parks 2012) upon these equations, to obtain the gradient $\frac{d\mathbf{x}^{i,\text{proj}}}{d\mathbf{x}^i}$. That is, we can differentiate these two equations with respect to $\mathbf{x}^i$, resulting in:

$$\begin{aligned}
&\begin{bmatrix} \nabla^2_{\mathbf{x}^{i,\text{proj}}} ||\mathbf{x}^{i,\text{proj}} - \mathbf{x}^i||_2 & \mathbf{A}^T \\ diag(\eta)\mathbf{A} & diag(\mathbf{A}\mathbf{x}^{i,\text{proj}} - b) \end{bmatrix} \begin{bmatrix} \frac{d\mathbf{x}^{i,\text{proj}}}{d\mathbf{x}^i} \\ \frac{d\eta}{d\mathbf{x}^i} \end{bmatrix} \\
&= - \begin{bmatrix} \frac{d\nabla_{\mathbf{x}^{i,\text{proj}}} ||\mathbf{x}^{i,\text{proj}} - \mathbf{x}^i||_2}{d\mathbf{x}^i} \\ 0 \end{bmatrix}
\end{aligned} \tag{21}$$

which implies equation (12), concluding our proof. □

Algorithm 2: Compute the gradients $\left\{ \frac{d\theta_t}{d\mathbf{z}_{t'}} (t' < t) \right\}$

for $t = 2 \to N$ **do**
  for $t' = 1 \to t-1$ **do**
    Compute the derivatives $\frac{d\mathbf{x}_{t''}}{d\mathbf{z}_{t'}} = \frac{d\mathbf{x}_{t''}}{d\theta_{t''}} \cdot \frac{d\theta_{t''}}{d\mathbf{z}_{t'}}$ for all $t'+1 \leq t'' \leq t-1$ where the derivative $\frac{d\mathbf{x}_{t''}}{d\theta_{t''}}$ is computed by Algorithm 1;
    Initialize $optL = +\infty$;
    for $round = 1 \to nRound$ **do**
      Initialize $\theta^{0,\text{proj}}$; $\delta L = +\infty$; $i = 0$;
      while $\delta L > 0$ **do**
        Update $i = i + 1$;
        Compute θ^i, $\theta^{i,\text{proj}}$ based on (14–15);
        Compute $\frac{d\theta^{i,\text{proj}}}{d\mathbf{z}_{t'}} = \frac{d\theta^{i,\text{proj}}}{d\theta^i} \cdot \frac{d\theta^i}{d\mathbf{z}_{t'}}$ where $\frac{d\theta^{i,\text{proj}}}{d\theta^i}$ is computed similar to $\frac{d\mathbf{x}^{i,\text{proj}}}{d\mathbf{x}^i}$ and $\frac{d\theta^i}{d\mathbf{z}_{t'}}$ is based on Prop. 2;
        Update $\delta L = L(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i,\text{proj}}) - L(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,\text{proj}})$;
      if $optL < L(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i,proj})$ **then**
        Update $optL = L(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i,\text{proj}})$;
        $\frac{d\theta_t}{d\mathbf{z}_{t'}} = \frac{d\theta^{i,\text{proj}}}{d\mathbf{z}_{t'}}$;

Proof of Proposition 2

Let $H(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,\text{proj}}) = \frac{dL(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,\text{proj}})}{d\theta^{i-1,\text{proj}}}$. By taking the derivatives on both sides of (14), we obtain:

$$\frac{d\theta^i}{d\mathbf{z}_{t'}} = \frac{d\theta^{i-1,\text{proj}}}{d\mathbf{z}_{t'}} - \alpha \frac{dH}{d\mathbf{z}_{t'}}$$

We observe that H is a function of $(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,\text{proj}})$ in which the defender's strategies $\mathbf{x}_{t''} \in \mathbf{X}_{t-1}$ with $t'' \leq t-1$ is a function of attack variables $\mathbf{z}_{t'}$ for all $t'' > t'$. In addition, $\theta^{i-1,\text{proj}}$ is also a function of $\mathbf{z}_{t'}$. Therefore, by applying the chain rule, we obtain:

$$\begin{aligned}
\frac{dH}{d\mathbf{z}_{t'}} &= \frac{dH}{d(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,\text{proj}})} \frac{d(\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}, \theta^{i-1,\text{proj}})}{d\mathbf{z}_{t'}} \\
&= \sum\nolimits_{t''=1}^{t-1} J_{H,x_{t''}} \cdot \frac{d\mathbf{x}_{t''}}{d\mathbf{z}_{t'}} + J_{H,z_{t''}} \cdot \frac{d\mathbf{z}_{t''}}{d\mathbf{z}_{t'}} + J_{H,\theta^{i-1,\text{proj}}} \cdot \frac{d\theta^{i-1,\text{proj}}}{d\mathbf{z}_{t'}}
\end{aligned}$$

Note that $\frac{d\mathbf{z}_{t''}}{d\mathbf{z}_{t'}} = 0$ if $\mathbf{z}_{t''} \neq \mathbf{z}_{t'}$ and is the identity matrix I otherwise. Also, $\frac{d\mathbf{x}_{t''}}{d\mathbf{z}_{t'}} = 0$ for all $t'' \leq t'$. Thus, the above is same as:

$$= \sum\nolimits_{t''=t'+1}^{t-1} J_{H,x_{t''}} \cdot \frac{d\mathbf{x}_{t''}}{d\mathbf{z}_{t'}} + J_{H,z_{t'}} + J_{H,\theta^{i-1,\text{proj}}} \cdot \frac{d\theta^{i-1,\text{proj}}}{d\mathbf{z}_{t'}}$$

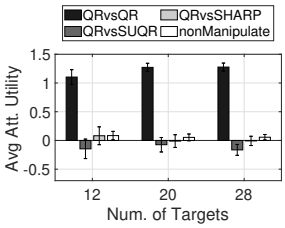
Note that when $t' = t-1$, we have:

$$\frac{dH}{d\mathbf{z}_{t-1}} = J_{H,z_{t'}} + J_{H,\theta^{i-1,\text{proj}}} \cdot \frac{d\theta^{i-1,\text{proj}}}{d\mathbf{z}_{t'}}$$

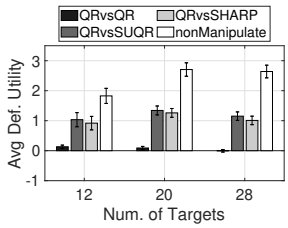
which concludes our proof.

Full details of Algorithm 2

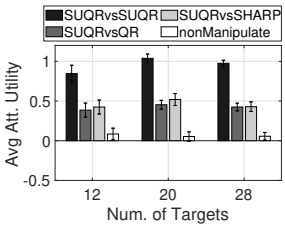
Additional Experiment



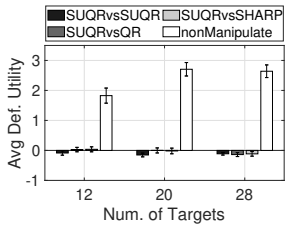
(a) QR manipulation, Attacker Utility



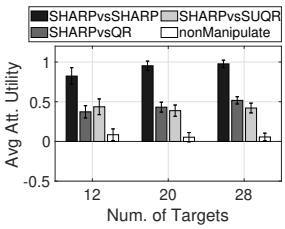
(b) QR manipulation, Defender Utility



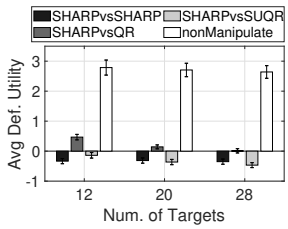
(c) SUQR manipulation, Attacker Utility



(d) SUQR manipulation, Defender Utility



(e) SHARP manipulation, Attacker Utility



(f) SHARP manipulation, Defender Utility

Figure 4: Players' utility evaluation. The defender follows Projected Gradient Descent to compute his strategies.