Incentive-Based Modeling and Inference of Attacker Intent, Objectives, and Strategies

Peng Liu
School of Information Sciences and Technology
Penn State University
University Park, PA 16802
pliu@ist.psu.edu

Wanyu Zang
School of Information Sciences and Technology
Penn State University
University Park, PA 16802
wyzang@psu.edu

ABSTRACT

Although the ability to model and infer Attacker Intent, Objectives and Strategies (AIOS) may dramatically advance the literature of risk assessment, harm prediction, and predictive or proactive cyber defense, existing AIOS inference techniques are ad hoc and system or application specific. In this paper, we present a general incentive-based method to model AIOS and a game theoretic approach to infer AIOS. On one hand, we found that the concept of incentives can unify a large variety of attacker intents; the concept of utilities can integrate incentives and costs in such a way that attacker objectives can be practically modeled. On the other hand, we developed a game theoretic AIOS formalization which can capture the inherent inter-dependency between AIOS and defender objectives and strategies in such a way that AIOS can be automatically inferred. Finally, we use a specific case study to show how AIOS can be inferred in real world attack-defense scenarios.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: Security and protection

General Terms

Security, Theory

Keywords

Attack Prediction, Game Theory, Computer Security

1. INTRODUCTION

The ability to model and infer Attacker Intent, Objectives and Strategies (AIOS) may dramatically advance the state-of-art of computer security for several reasons. First, for many "very difficult to prevent" attacks such as DDoS, given the specification of a system protected by a set of specific security mechanisms, this ability could tell us which kind of strategies are more likely to be taken by the attacker than the others, even before such an attack happens. Such AIOS inferences may lead to more precise risk assessment and harm prediction.

Second, AIOS modeling and inference could be more beneficial during run time. A big security challenge in countering a multi-phase, well planned, carefully hided attack from either malicious insiders or outside attackers is "how to make correct proactive (especially predictive) real-time defense decisions during an earlier stage of the attack in such a way that much less harm will be caused without consuming a lot of resources"? Although many proactive defense techniques are developed such as sandboxing [21] and isolation [18], making the right proactive defense decisions in real-time is very difficult primarily due to the fact that intrusion detection during the early stage of an attack can lead to many false alarms, which could make these proactive defense actions very expensive in terms of both resources and denial-of-service.

Although alert correlation techniques [7, 26] may reduce the number of false alarms by correlating a set of alerts into an attack scenario (i.e., steps involved in an attack) and may even tell which kind of attack actions may follow a given action [8], they are limited in supporting proactive intrusion response in two aspects. (1) When many types of (subsequences of) legitimate actions may follow a given suspicious action, alert correlation can do nothing except waiting until a more complete attack scenario emerges. However, intrusion response at this moment could be "too late". (2) When many types of attack actions may follow a given (preparation) action, alert correlation cannot tell which actions are more likely to be taken by the attacker next. As a result, since taking proactive defense actions for each of the attack actions can be too expensive, the response may have to wait until it is clear what attack actions will happen next - perhaps during a later stage of the attack. However, late intrusion response usually means more harm. By contrast, with the ability to model and infer AIOS, given any suspicious action, we can predict the harm that could be caused; then we can make better and affordable proactive intrusion response decisions based on the corresponding risk, the corresponding cost (e.g., due to the possibility of false alarms), and the attack action inferences. Moreover, the intrusion response time is substantially shortened.

However, with a focus on attack characteristics [17] and attack effects [2, 33], existing AIOS inference techniques are ad hoc and system or application specific [28, 11]. To systematically model and infer AIOS, we need to distinguish AIOS from both attack actions and attack effects. Since the same attack action can be issued by two attackers with very different intents and objectives, AIOS cannot be directly inferred from the characteristics of attacks. Although the attacker achieves his or her intents and objectives through attacks and their effects, the *mapping* from attack actions and/or effects to attacker intents and/or objectives is usually not one-to-one but one-to-many, and more interesting, the (average) *cardinality* of this mapping can be much larger than the mapping from attacker intents and/or objectives to attack actions and/or effects. This asymmetry nature indicates that in many cases using AIOS models to predict attack actions can be more precise than using the set of actions already taken by the attacker based on either their effects or the *causal* relationship between them and some other attack actions*. As a result, although a variety of attack taxonomies and attribute databases have been developed, people's ability to model and infer AIOS, to predict attacks, and to do proactive intrusion response is still very limited. Nevertheless, a good understanding of attacks is the foundation of practical AIOS modeling and inference.

In this paper, we present a systematic incentive-based method to model AIOS and a game theoretic approach to infer AIOS. On one hand, we found that the concept of incentives can unify a large variety of attacker intents; the concept of utilities can integrate incentives and costs in such a way that attacker objectives can be practically modeled. On the other hand, we developed a game theoretic AIOS formalization which can capture the inherent inter-dependency between AIOS and defender objectives and strategies in such a way that AIOS can be automatically inferred. Finally, we use a specific case study to show how AIOS can be inferred in real world attack-defense scenarios. The proposed framework, in some sense, is an *economics-based* framework since it is based on economic incentives, utilities, and payoffs.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. Section 3 presents a conceptual, incentive-based framework for AIOS modeling. In Section 4, we present a game-theoretic formalization of this framework. Section 5 shows how to compute AIOS inferences in real world attack-defense scenarios. In Section 6, we mention several future research issues.

2. RELATED WORK

The use of game theory in modeling attackers and defenders has been addressed in several other research. In [28], Syverson talks about "good" nodes fighting "evil" nodes in a network and suggests using stochastic games for reasoning and analysis. In [20], Lye et al. precisely formalize this idea using a general-sum stochastic game model and give a concrete example in detail where the attacker is attacking a simple enterprise network that provides some Internet services such as web and FTP. A set of specific states regarding this example are identified, state-transition probabilities are assumed, and the Nash equilibrium or best-response strategies for the players are computed.

In [3], Browne describes how static games can be used to analyze attacks involving complicated and heterogeneous military networks. In his example, a defense team has to defend a network of three hosts against an attacking team's *worms*. The defense team can choose either to run a worm detector or not. Depending on the combined attack and defense actions, each outcome has different costs. In [4], Burke studies the use of repeated games with incomplete information to model attackers and defenders in information warfare. In [13], Hespanha and Bohacek discuss zero-sum routing games where an adversary (or attacker) tries to intersect data packets in a computer network. The designer of the network has to find routing policies that avoid links that are under the attacker's surveillance. In [34], Xu and Lee use game-theoretical framework to analyze the performance of their proposed DDoS defense system and to guide its design and performance tuning accordingly.

Our work is different from the above game theoretic attacker modeling works in several aspects. First, these works focus on specific attack-defense scenarios, while our work focuses on general AIOS modeling. Second, these works focus on specific types of game models, e.g., static games, repeated games, or stochastic games, while our work focuses on the fundamental characteristics of AIOS, and game models are only one possible formalization of our AIOS framework. In addition, our AIOS framework shows the inherent relationship between AIOS and the different types of game models, and identifies the conditions under which a specific type of game models will be feasible and desirable. Third, our work systematically identifies the properties of a good AIOS formalization. These properties not only can be used to evaluate the merits and limitations of game theoretic AIOS models, but also can motivate new AIOS models that can improve the above game-theory models or even go beyond standard game-theoretic models.

In [11], information security is used as a response to game theoretic *Competitor Analysis Systems* (CAS) for the purpose of protecting a firm's valuable business data from its competitors. Although understanding and predicting the behavior of competitors are key aspects of competitor analysis, the behaviors CAS want to predict are not cyber attacks. Moreover, security is what our game theoretic system wants to model while security is used in [11] to protect a game theoretic system.

The computational complexity of game theoretic analysis is investigated in several research. For example, [6] shows that both determining whether a pure-strategy Bayes-Nash equilibrium exists and determining whether a pure-strategy Nash equilibrium exists in a stochastic (Markov) game are NP-hard. Moreover, [16] shows that some specific knowledge representations, in certain settings, can dramatically speed up equilibrium finding.

The marriage of economics and information security has attracted a lot of interests recently (a lot of related work can be found at the economics and security resource page maintained by Ross Anderson at http://www.cl.cam.ac.uk/∼rja14 /econsec.html). However, these work focuses on the economics perspective of security (e.g., security market, security insurance), while our approach is to apply economics concepts to model and infer AIOS.

In recent years, it is found that economic *mechanism de-*

*To illustrate, consider a large space of strategies the attacker may take according to his or her intent and objectives where each strategy is simply a sequence of actions. An attack action may belong to many strategies, and the *consequences* of the action could satisfy the *preconditions* of many other actions, but each strategy usually contains only a small number of actions.

sign theory [30, 5, 12] can be very valuable in solving a variety of Internet computing problems such as routing, packet scheduling, and web caching [9, 32, 27]. Although when market-based mechanisms are used to defend against attackers [31], the AIOS are incentive-based, which is consistent with our framework, market-based computing does not imply an in-depth AIOS model.

Finally, it should be noticed that AIOS modeling and inference are very different from intrusion detection[19, 25, 23]. Intrusion detection is based on the characteristics of attacks, while AIOS modeling is based on the characteristics of attackers. Intrusion detection focuses on the attacks that have already happened, while AIOS inference focuses on that attacks that may happen in the future.

3. AN INCENTIVE-BASED FRAMEWORK FOR AIOS MODELING

We build our AIOS models on top of the relationships between the attacker and a computer system (i.e., the defender). In our model, the computer system can be be any kind (e.g., a network system, a distributed system, a database system). We call it the *system* for short. The attacker issues *attacks* to the system. Each attack is a sequence of *attack actions* associated with the system. For example, an action can be the sending of a message, the submission of a transaction, the execution of a piece of code, etc. An attack will cause some *effects* on the system, i.e., transforming the system from one *state* to another state. Part of the system is a set of specific security *mechanisms*. A mechanism can be a piece of software or hardware (e.g., a firewall, an access controller, an IDS). A mechanism usually involves a sequence of *defense actions* associated with the system when being activated. A security mechanism is *activated* when an *event* arrives which causes a set of specific *conditions* to be satisfied. Many of these conditions are associated with the effects of an attack action in *reactive defense*, or the *prediction* of an incoming attack action in *proactive defense*. Finally, a *defense posture* of the system is defined by the set of security mechanisms and the ways they are activated.

There are several unique characteristics of the attacker-system relationship which we will exploit shortly.

⊙*Intentional Attack Property.* Attacks are typically not random. They are planned by the attacker based on some intent and objectives.

⊙*Strategy-Interdependency Property.* Whether an attack can succeed is dependent on how the system is protected. Whether a security mechanism is effective is dependent on how the system is attacked. In other words, the capacity of either an attack or a defense posture should be measured in a *relative* way. Note that we will define the notion of strategy shortly.

⊙*Uncertainty Property.* The attacker usually has *incomplete* information or *knowledge* about the system, and vice versa.

3.1 Incentive-Based Attacker Intent Modeling

Different attackers usually have different intents even when they they issue the same attack. For example, some attackers attack the system to show off their hacking capacity, some hackers attack the system to remind the administrator of a security flaw, cyber terrorists attack our cyberspace for creating terrors, business competitors may attack each other's information systems to increase their market shares, just to name a few It is clear that investigating the characteristics of each kind of intents involves a lot of effort and complexity, and such complexity actually prevents us from building a general, robust connection between attacker intents and attack actions. This connection is necessary to do almost every kind of attacker behavior inference.

We focus on building general, simple intent models. In particular, we believe that the concept of economic "incentive" can be used to model attacker intent in a general way. In our model, the attacker's *intent* is simply to maximize his or her *incentives*. In other words, the attacker is motivated by the possibility of gaining some incentives. Most, if not all, kinds of intents can be modeled as incentives such as the amount of profit earned, the amount of terror caused, and the amount of satisfaction because of a nice show-off. We may use economics theory to classify incentives into such categories as money, emotional reward and fame.

To infer attacker intents, we need to be able to compare one incentive with another. Incentives can be compared with each other either qualitatively or quantitatively. Incentives can be *quantified* in several ways. For example, profits can be quantified by such monetary units as dollars. One critical issue in measuring and comparing incentives is that under different *value systems*, different comparison results may be obtained. For example, different types of people value such incentives as time, fame and faith differently. As a result, very misleading attacker intent inferences could be produced if we use our value system to evaluate the attacker's incentives.

After an attack is enforced, the incentives (e.g., money, fame) earned by the attacker are dependent on the effects of the attack, which are typically captured by the degradation of a specific set of security measurements that the system cares. Each such measurement is associated with a specific *security metric*. Some widely used categories of security metrics include but not limited to confidentiality, integrity, availability (against denial-of-service), non-repudiation, and authentication. In our model, we call the set of security metrics that a system wants to protect the *metric vector* of the system. (Note that different systems may have different metric vectors.) At time t, the measurements associated with the system's metric vector are called the *security vector* of the system at time t, denoted V_t^s. As a result, assume an attack starts at time t_1 and ends at t_2, then the incentives earned by the attacker (via the attack) can be measured by $degradation(V_{t_1}^s, V_{t_2}^s)$, which basically computes the *distance* between the two security vectors.

The above discussion indicates the following property of AIOS inference.

⊙ *Attack effect property.* Effects of attacks usually yield more insights about attacker intent and objectives than attack actions.

3.2 Incentive-Based Attacker Objective Modeling

In real world, many attackers face a set of constraints when issuing an attack, for example, an attacker may have limited resources; a bad insider may worry about the risk of being arrested and put into jail. However, our intent model assumes no constraints. To model attacker motivations in a more realistic way, we incorporate constraints in our attack objective model. In particular, we classify *constraints* into two categories: *cost constraints* and *non-cost* *constraints.* (a) Cost constraints are constraints on things that the attacker can "buy" or "trade" such as hardware, software, Internet connection, and time. Such things are typically used to measure the cost of an attack. (b) Non-cost constraints are constraints on things that the attacker cannot buy such as faith-related constraints and top secret attacking tools.

The *cost* of an attack is not only dependent on the resources needed to enforce the attack, but also dependent on the *risk* for the attacker to be traced-back, arrested, and published. Based on the relationship between incentives and costs, we classify attackers into two categories: (a) *rational attackers* have concerns about the costs associated with their attacks. That is, when the same incentive can be obtained by two attacks with different costs, rational attackers will pick the one with a lower cost. (b) *Irrational attackers* have no concerns about the costs associated with their attacks. They only want to maximize the incentives.

Given a set of (cost) constraints, inferring the attack actions of an irrational attacker is not so difficult a task since we need only to find out "what are the most rewarding attack actions in the eyes of the attacker without violating the constraints?" By contrast, we found that inferring the attack actions of a rational attacker is more challenging. In this paper, we will focus on how to model and infer the IOS of rational attackers.

In our model, an attacker's *objective* is to maximize his or her *utilities* through an attack without violating the set of cost and non-cost constraints associated with the attacker. The utilities earned by an attacker indicate a distance between the incentives earned by the attacker and the cost of the attack. The distance can be defined in several ways, for example, $utilities = incentives - cost$; $utilities = \frac{incentives}{cost}$. Note that the cost of an attack can be measured by a set of cost metrics which capture both attacking resources and risk.

3.3 Incentive-Based Attacker Strategy Modeling

Strategies are taken to achieve objectives. The strategy-interdependency property indicates that part of a good attacker strategy model should be the defense strategy model because otherwise we will build our AIOS models on top of the assumption that the system never changes its defense posture, which is too restrictive. See that whenever the system's defense posture is changed, the defense strategy is changed.

In our model, attack strategies are defined based on the "battles" between the attacker and the system. Each attack triggers a *battle* which usually involves multiple *phases*. (For example, many worm-based attacks involve such phases as reconnaissance, probe and attack, toehold, advancement, stealth, and takeover.) In each phase, the attacker may take some attack actions and the system may take some defense actions (automatically). How such attack actions are taken in each phase of the battle defines the attacker's *strategy* for the battle. How such defense actions are taken defines the system's defense strategy. We will show some concrete attack and defense strategies in Section 5. Not that an attack strategy is not simply a sequence of attack actions; it may also include such dynamic, strategic decision making rules as "what action should be taken under what state or condition". Hence, during two different battles with the system, the same attack strategy may result in two different sequences of attack actions. When a battle has multiple phases, we could have two possible types of attack or defense strategies: (1) *static strategies* take exactly the same set of actions in every phase; (2) *dynamic strategies* adjust actions when a new phase arrives based on what has happened.

In our model, each defense posture defines a defense strategy since it specifies how a set of security mechanisms behave in the face of an attack. Some security mechanisms are *adaptive*, but adaptations do not indicate a different defense strategy because the adaptation rules are not changed. The way we define defense postures is general enough to support a variety of defense strategies. The definition allows us to (dynamically) add, activate, de-activate, or remove a security mechanism. It also allows us to *reconfigure* a security mechanism by "replacing" an old mechanism with the reconfigured mechanism.

In our model, an attacker's *strategy space* includes every possible attack strategy of the attacker under the set of constraints associated with the attacker. To infer an attacker's strategy space, a good understanding of the system's vulnerabilities and the attack/threat taxonomy is necessary. Moreover, constraints and costs help infer the boundary of a strategy space, since they imply which kind of attacks will not be enforced. Similarly, the system's *strategy space* is determined by the set of defense postures of the system. Due to the constraints associated with the system and the cost of security[†], the system's strategy space is usually bounded.

A key issue in modeling attacker strategies is to help compare two attack strategies in terms of "which one is better (for the attacker)?" Based on why attack strategies are taken, the answer is dependent on the degree to which the attacker objectives can be achieved with a strategy. Based on the definition of attacker objectives, the answer is then dependent on "which strategy can bring in more utilities to the attacker?" Based on the definition of utilities, if we assume that the costs for these two strategies are the same, the answer is then dependent on "which strategy can bring in more incentives to the attacker?" Since attacker incentives are determined by $degradation(V^s_{t_1}, V^s_{t_2})$, the answer is then dependent on "which strategy can cause more degradation to the system's security vector?" However, the answer to this question is in general determined by "which defense strategy will be taken by the system?", since different battles may lead to different amount of security degradation. Therefore, the overall answer is that "which one is a better attack strategy?" is dependent on "what are the two defense strategies taken by the system respectively?" This answer confirms the strategy-interdependency property.

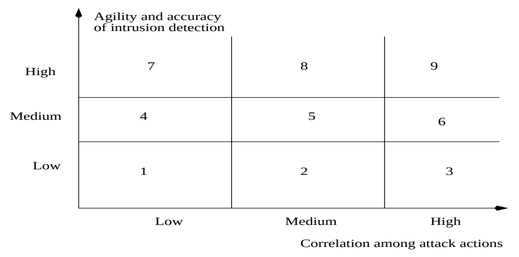
The above discussion implies the following property of AIOS inference.

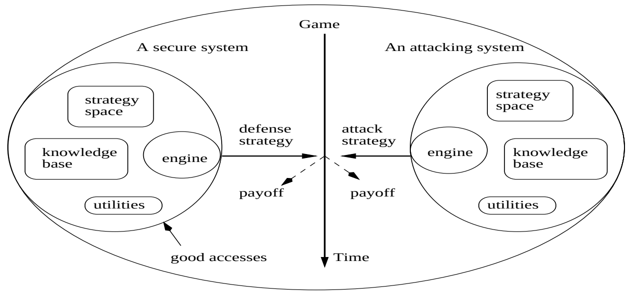
$\odot$ *Dual property.* (a) "Which one is a better attack (defense) strategy?" is dependent on "what are the defense (attack) strategies taken?" (b) Each type of information useful for the attacker (system) to choose a good attack (defense) strategy will be useful for the system (attacker) to choose a good defense (attack) strategy.

4. A GAME-THEORETIC FORMALIZATION

[†]Security mechanisms not only consume resources but also can have a negative impact on the system's functionality and performance.

Our goal is to formalize the AIOS models developed in the previous section in such a way that good inferences of AIOS can be automatically computed. For this purpose, we first propose a game-theoretic AIOS formalization, then we show why it is a good formalization.
Our game-theoretic AIOS formalization is shown in Figure 1(b), where
⊙ Instead of neglecting the attacker and viewing attacks as part of the system's environment, we model the attacker as a "peer" of the system, namely the *attacking system*.
⊙ The *environment* only includes the set of good accesses by a legitimate user.
⊙ We further split the system into two parts: the *service part* includes all and only the components that provide computing services to users; and the *protection part* includes the set of security mechanisms.
⊙ Instead of passively monitoring, detecting, and reacting to attacks, the relation between the system and the attacker is modeled as a *game* (or battle) across the time dimension where the system may actively take defense actions.
⊙ The game is a 6-tuple. (1) The two *players*, namely the system and the attacking system. Note that the "real" player for the system is the set of security mechanisms.
(2) The *game type* (e.g., a Bayesian game or a stochastic game) and the set of type-specific parameters of the game.
(3) The two strategy spaces of the two players, defined in the same say as in Section 3. The attacker's strategy space is denoted as $S^a = \{s_1^a, ..., s_m^a\}$ where s_i^a is an attack strategy. The system's strategy space is denoted as $S^d = \{s_1^d, ..., s_m^d\}$ where s_i^d is a defense strategy. Note that the constraints associated with the attacker and the cost of each attack imply the boundary of S^a. A more detailed formalization of attack strategies is described in Section 5.
(4) A set of game *plays*. A play is a function $pl_i : S^a \times S^d \to O$ where O is the set of *outcomes* which indicate the effects of an attack. Each play involves one battle due to an attack. Each play may have several phases. We assume each player uses a *game engine* to determine which strategy should be taken in a specific play.
(5) The two *utility (or payoff) functions* which calculate the utilities earn by the two players out of each play. The attacker's utility function is $u^a : S^a \times S^d \to R$ where R is the set of utility measurements. Given a play (s_i^a, s_i^d), the attack cost is an attribute of s_i^a, denoted $cost(s_i^a)$. The attacker's incentives are determined by $degradation(V_{t_1}^s, V_{t_2}^s)$ where t_1 is the time when the play starts; t_2 is the time when the play ends; and security vector $V_{t_2}^s$ is dependent on the outcome of the play, namely $pl_i(s_i^a, s_i^d)$. And $u^a(s_i^a, s_i^d)$ is a distance between $cost(s_i^a)$ and the attacker's incentives. By contrast, the system's utility function is $u^d : S^a \times S^d \to R$. Given a play (s_i^a, s_i^d), the system's cost is $cost(s_i^d)$. The system's incentives are determined by $improvement(V_\emptyset^s, V_{s_i^d}^s)$ where $V_\emptyset^s$ is the security vector resulted after the attack when no security mechanisms are deployed; $V_{s_i^d}^s$ is the vector resulted after the attack when strategy s^d is taken. And $u^d(s_i^a, s_i^d)$ is still a distance between the system's incentives and cost.
(6) A *knowledge base* maintained by each player. The attacker's (system's) knowledge base maintains the attacker's (system's) knowledge about the system's (attacker's) strategy space (including the system's (attacker's) cost and constraints), the system's (attacker's) value system, the system's metric and security vectors. Note that the attacker's (system's) knowledge may not always be true; it in fact captures the attacker's (system's) *beliefs*.
⊙ Note that for clarify, only the game-relevant components are shown in Figure 1(b). Note also that the game model can be extended to cover multiple attackers who are either cooperating with other attackers (i.e., cooperative) or not (i.e., non-cooperative). This extension is out of the scope of this paper.
Discussion. We believe a game theoretic formalization can be very valuable for AIOS modeling and inference because (1) such a formalization shifts the focus of traditional AIOS modeling from attacks to attackers; (2) such a formalization captures every key property of the attacker-system relation such as the Intentional Attack Property and the Strategy Interdependency Property; (3) such a formalization captures every key elements of our incentive-based AIOS modeling framework such as incentives, utilities, costs, risks, constraints, strategies, security mechanisms, security metrics, defense postures, vulnerabilities, attacks, threats, knowledge, and uncertainty; (4) such a formalization can be used to infer AIOS. The rationale is that (a) non-cooperative game theory is the primary tool to handle *strategic interdependence*[22], which is the fundamental property of the attacker-system relation; (4b) game-theoretic models have been successfully used to predict *rational* behaviors in many applications such as auctions and their *rationality* notion (that each player plays an expected-utility maximizing best-response to every other player) is consistent with the goals of many, if not most, attackers and systems; (4c) Nash equilibria of attacker-system games can lead to good AIOS inferences since Nash equilibria indicate the "best" rational behaviors of a player, and when the system always takes a Nash equilibrium defense strategy, only a Nash equilibrium attack strategy can maximize the attacker's utilities.


Agility and accuracy of intrusion detection
High
Medium
Low
7
8
9
4
5
6
1
2
3
Low
Medium
High
Correlation among attack actions
(a) A Taxonomy of Game Theoretic AIOS Models



(b) A Game Theoretic Formalization

Figure 1:

5. GAME THEORETIC AIOS INFERENCE

In real world, how to model and infer AIOS? The previous presentation implies the following pipeline:
⊙ (1) Make assumptions about the system and the (types of) attacks that concern the system. Note that practical AIOS inferences may only be able to be computed within some domain or scope (due to the complexity).
⊙ (2) Model the attacker intent, objectives and strategies (conceptually). Specify the attacker's utility function and strategy space. Estimate the attacker's knowledge base.
⊙ (3) Specify the system's metric vector and security vector. Specify the system's utility function and strategy space. Build the system's knowledge base.
⊙ (4) Determine the game type of the game theoretic AIOS inference model that will be developed, then develop the model accordingly.
⊙ (5) Compute the set of Nash equilibrium strategies of the AIOS inference game model developed in Step 4. A key task is to handle the computation complexity. If the *complexity* is too much, we need to do (inference) precision-performance tradeoffs properly using some (semantics-based) approximate algorithms.
⊙ (6) Validate the inferences generated in Step 5. The relevant tasks include but not limited to *accuracy analysis* (i.e., how accurate are the inferences?) and *sensitivity analysis* (i.e., are the inferences sensitive to some specific model parameters?). The relevant validation techniques include but not limited to (a) investigating the degree to which the inferences match the real world intrusions; (b) extracting a set of high-level properties or features from the set of inferences and asking security experts to evaluate if the set of properties match their experiences, beliefs, or intuitions.
⊙ (7) If the validation results are not satisfactory, go back to Step 1 to rebuild or improve the inference model.

In the following, before we do the case study to show how AIOS can be inferred in real world attack-defense scenarios, we would first show how to choose the right game type for a real world AIOS inference task.

5.1 How to Choose the Right Game Theoretic AIOS Model?

A good AIOS inference model must be built on top of the real characteristics of the attack-defense (A-D) scenario. Different A-D scenarios may require different inference models. Hence, to develop a taxonomy of game theoretic AIOS inference models, we need a general, simple model to classify the characteristics of A-D scenarios. For this purpose, we will start with two critical factors of the attacker-system relation, namely state and time. In our model, there are two categories of states:
⊙ *System state:* At one point of time, the *state* of a system is determined by the state of each component of the system's service part. A component of the system's service part can be a piece of data or a piece of code. Note that sometimes a piece of code can be handled as a piece of data. It should be noticed that the system's state has nothing to do with the system's defense posture, which is determined by the state of each component of the system's protection part.
⊙ *Attack state:* Attack states classify system states from the attack-defense perspective. Every attack action, if successfully taken, will have some *effects* on the system state. Such effects are usually determined by the characteristics of the attack. After a specific attack happens, the resulted effects are specified as an attack state. For example, all the possible states of a web server system after its *Ftpd* service is hacked can be denoted as the *Ftpd_hacked* attack state. Hence each attack state usually covers a cluster of system states.

It is clear that the attacker is always clear about the current attack state, but the defender (i.e., the system) is usually not. The system uses an intrusion detector to learn the current attack state. Due to the latency of intrusion detection, the system may know an attack state with some delay. Due to the false alarms, the system may have wrong belief about the current attack state.

The relation between states and times is simple. At one point of time, the system must be associated with a specific system state and attack state. Good accesses, attack actions, and defense actions can all change the system state, however, only attacks and defense operations can change attack states. Changes of both system states and attack states indicate changes of time. An interesting question here is: when should we terminate an attack state? One way to solve this problem is to give each attack a life time. When the life time of an attack is over, we make the corresponding attack state part of the history. The life time of an attack should involve some defense actions or operations, since when the life of the attack is over, the system should have already been recovered from the attack in many possible ways, e.g., replacing the system with a new system, repairing the damaged part of the system, etc..

We model the battles between the attacker and the system as follows.

Definition 1. **(General Model)** A *battle* between the attacker and the system is an interleaved sequence of system states and actions such that
⊙ Each action belongs to one of three possible types: (a) the action can be an attack action which is part of an attack strategy, (b) the action can be an action or operation taken by a legitimate user which indicates a good access, (c) the action can be a defense action which is part of a defense strategy. We denote an attack action as o_b^i. We denote a good access action as o_g^i. We denote a defense action as o_d^i.
⊙ There must be either one attack action or one good access action between two adjacent states. No more than one attack action can happen between two adjacent states. No more than one good access action can happen between two adjacent states either.
⊙ There is exactly one defense action between two adjacent states. However, a defense action can be a *null* action, but neither an attack action nor a good access action can be a null action.

Under some specific conditions, the above model can be further simplified. In particular, when every attack action can be detected instantly after it happens with accuracy, the fights between the attacker and the system can be modeled as follows. Here, we model an intrusion as a sequence of attack actions, namely $I_j = \{o_b^1, o_b^2, ..., o_b^n\}$. Note that here since we can distinguish bad actions from good ones, a set of system states can be clustered into a specific attack state, and good actions need not be explicitly modeled.

Definition 2. **(Under Instant Accurate Intrusion Detection)** A *battle* between the attacker and the system is an interleaved sequence of attack states and actions such that

⊙ Each action belongs to one of two possible types: (a) an attack action; or (b) a defense action.
⊙ There is exactly one attack action between two adjacent states. No more than one attack action can happen between two adjacent states.
⊙ There is exactly one defense action between two adjacent states. A defense action can be a null action.
⊙ A *fight* is composed of two adjacent attack states and the pair of actions between them. It is clear that every pair of attack and defense actions (o_b^i, o_d^i) can transform the system from one attack state to another.

When intrusions can be instantly detected with accuracy, it is clear that both the system and the attacker know the current attack state for sure. The system's utility earned after each fight, denoted $u_d(o_b^i, o_d^i)$, is computable if we know which good actions are involved in the fight, so is $u_a(o_b^i, o_d^i)$. Note that the system is clear about the set of good actions involved in each fight, but the attacker could have some uncertainty about the good actions.

However, when intrusion detection has delay or when the detection is not 100% accurate, the simplified model cannot realistically model the fights between the attacker and the system, and the general model is the model we should use. Why? When the accuracy is low, even if you can instantly raise alarms, the simplified model still has too much uncertainty which makes the inferences generated by the model difficult to be validated. See that because of the inaccuracy, the system is actually not sure about the current attack state, and taking the defense action as if the raised alarm is true is not only not secure but also very expensive. When the detection latency is long, after an attack action is detected, several attack states may have already been bypassed, and as a result, the system can only take a null defense action for every bypassed state. This indicates that the attacker can take a lot of advantage if the simplified model is used to guide the defense.

The above discussion shows that (a) if the game model is not properly chosen and followed, the system can lose a lot of security and assurance, and that (b) the agility and accuracy of intrusion detection play a critical role in finding optimal AIOS game models. In addition, we found that the correlation among attack actions also plays a critical role in finding optimal AIOS game models. Based on these two factors, the taxonomy of AIOS models can follow the regions shown in Figure 1(a), and the taxonomy can be simply summarized as follows:
⊙ In region 9, *stochastic games* should be used together with reactive defense. When intrusion detection is highly effective, stochastic games become feasible. See that not also that each attack state can be accurately identified by the system with agility, which enables effective reactive defense, but only that the transition probability among attack states can be estimated with good accuracy. When there is strong correlation among attack actions, stochastic game models are better than repeated game models, since they can model the correlation relation among attack actions, but repeated game models cannot.
⊙ In region 1, *Bayesian multistage games* should be used together with proactive defense. When the intrusion detection effectiveness is poor, the system can have substantial uncertainty about the current attack state, and such uncertainty usually makes stochastic game models infeasible, since the utility of stochastic game models is dependent on the assumption that each attack state can be instantly identified by each player. In this case, Bayesian game models are a robust, realistic solution, since they do not require accuracy detection, and they do not require instant detection either.
⊙ In Region 7, Bayesian multistage games should be used.
⊙ In region 3, normal *dynamic multistage games* should be used, and sub-game perfect strategies should be taken by the players. In this case, compared with the combination of probabilistic "attack states" and stochastic game models, simple dynamic multistage games are easier, cheaper, having a smaller search space, more accurate, and having no need to know all the attack states.
⊙ Finally, the gray areas usually need a tradeoff between the extreme cases when we need to build a good game theoretic AIOS model for such a region. The tradeoffs are dependent on many factors, such as the amount of uncertainty, accuracy, and sensitivity, as we will discuss shortly.
⊙ Note that every type of AIOS inference games can support both *pure* strategies and *mixed* strategies.

5.2 Bayesian Game-Theoretic AIOS Models

In this section, we present a concrete Bayesian game-theoretic AIOS model, which can be used to handle regions 1 and 7. This model will be used shortly to do the case study.

A Bayesian game-theoretic AIOS inference model is composed of two parts: a Bayesian *game model* that characterizes the attacker-system relation, and a set of *AIOS inferences* generated by the game model. In particular, the game model is a specific 2-player finitely repeated Bayesian game between the system and a *subject*, where (a) there can be multiple *types* of subjects. And the *type space* is denoted $T^{sub} = \{good, bad\}$. A subject's type is privately known by the subject. (b) A^{sys} is the *action space* of the system, and A^{sub} is the action space of the subject. One or more actions can build a *strategy*. (c) The game has a finite number of plays (or stages) and each play include a pair of simultaneous actions (a^{sys}, a^{sub}). And each play will have an outcome denoted $o(a^{sys}, a^{sub})$. (d) The system is uncertain about the type of the subject. This uncertainty is measured by the system's *type belief*, denoted p_{sys}^{type}. For example, $p_{sys}^{type}(bad)$, a probability, denotes the system's belief about the statement that the subject is a bad guy. (e) For each outcome o, the system's utility function is $u_{sys}(o) = p_{sys}^{type}(good)u_{sys}^{good}(o) + p_{sys}^{type}(bad)u_{sys}^{bad}(o)$. If the subject is a good guy, his or her utilities are determined by $u_{sub}(o; good)$, otherwise, his or her utilities are determined by $u_{sub}(o; bad)$.

On the other hand, the set of AIOS inferences are determined by the Nash equilibria of the game model based on the rationality notion of an expected-utility maximizer[‡]. In particular, for each Nash equilibrium of the game, denoted $(a_{sys}^*, a_{bad}^*, a_{good}^*)$, the game model will output $\{a_{sys}^*, a_{bad}^*, u_{sys}(a_{sys}^*, a_{bad}^*), u_{sub}(a_{sys}^*, a_{bad}^*; bad)\}$ as the AIOS inferences, where $u_{sub}(a_{sys}^*, a_{bad}^*; bad)$ can be mapped to the amount of security vector degradation caused by the attack, which clearly indicates the attacker intent and objectives; a_{bad}^* indicates the kind of strategies that are more likely to be taken by the attacker. Moreover, as side benefits, a_{sys}^*

[‡]The Nash equilibrium theory can be found in [24]. Note that mixed strategy Nash equilibria exist for every Bayesian game, although sometimes no pure strategy Nash equilibrium exists. Also a game may have multiple Nash equilibria.

indicates a better defense posture and $u_{sys}(a^*_{sys}, a^*_{bad})$ indicates the overall resilience of the system.

Discussion. Bayesian AIOS inference models are simple, robust and may work well even when a very little amount of information is available. For example, in Region 1, although neither the intrusion detector nor the previous actions (of a subject) can provide hints, timely inferences could still be generated based on a probabilistic estimation of how intense the attacks are. Since a small number of disturbing attacks will not affect the estimated intensity degree much, Bayesian AIOS inference models are very robust to disturbing alerts.

5.3 Case Study : Inferring the AIOS of DDoS Attackers

In this study, we want to infer the AIOS of the attackers that enforce *brute-force* DDoS attacks. (Although DDoS attacks with clear signatures, such as SYN flooding, can be effectively countered, DDoS attacks without clear signatures, such as brute-force DDoS attacks, are very difficult to defend against since the system is not clear which packets are DDoS packets and which are not.) An example scenario is shown in Figure 2 where many zombies (i.e., a subset of source hosts $\{S_0, ..., S_{64}\}$) are flooding a couple of web sites (i.e., the *victims*) using normal HTTP requests. Note that the web sites may stay on different subnets.

Although our AIOS inferring technique can handle almost every DDoS defense mechanism, to make this case study more tangible, we select *pushback* [15], a popular technique, as the security mechanism. Pushback uses *aggregates*, i.e., a collection of packets from one or more flows that have some property in common, to identify and *rate-limit* the packets that are most likely to cause congestion or DoS. Pushback is a coordinated defense mechanism which typically involves multiple routers. To illustrate, consider Figure 2 again, when router $R1.0$ detects a congestion caused by a set of aggregates, $R1.0$ will not only rate-limit these aggregates, but also request adjacent upstream routers (e.g., $R2.1$) to rate-limit the corresponding aggregates via some *pushback messages*.

Now, we are ready to present the specific Bayesian game-theoretic AIOS inference model. To save space, we only mention the differences from the generic model proposed in the previous section. In this model,

$\odot$ (a) $V = \{v_1, ..., v_l\}$ is the set of victims of the DDoS attack (note that our inference model only handles a single DDoS attack). Note that the victims may stay on different subnets. The system is composed of every router that is part of the pushback defense, denoted $\{R_1, ..., R_n\}$. The subject is the set of hosts that send packets to V. The type of a host is either *good* or *bad* (i.e., a zombie), but not both.

$\odot$ (b) $A^{sub} = \{T_1, ..., T_m\}$, where T_i is a communication *task*, e.g., visiting a web site, transferring a file, sending out a set of DDoS packets, etc.

$\odot$ (c) A^{sys} is determined by the pushback postures of each router in the system. In particular, the pushback behavior of a router is determined by the following configurable parameters: $[p^1_{sys}]$ *congestion checking time* (default value: 2s) is the interval time that the router checks congestion. When serious congestion is detected, the router will identify (and rate-limit) the aggregate(s) responsible for the congestion and send out some pushback messages. Note that in this study the *thresholds* for "reporting" serious congestion and for determining who should receive pushback messages are fixed. Note also that how the rate limits (for each aggregate) are set up is also fixed. $[p^2_{sys}]$ *Cycle time* (default value: 5s) is the interval time that the router reviews the limits imposed on its aggregates and sends refresh messages to the adjacent upstream routers to update their rate limits. Note that how such rate limits are updated is fixed in this study. $[p^3_{sys}]$ *Target drop rate* (default value: 5%) determines the upper-bound drop rate of the router's output queue. To achieve this upper-bound, the rate limiter should make the bit rate towards the output queue less than $B/(1 - target\_drop\_rate)$, where B is the bandwidth of the output link. $[p^4_{sys}]$ *Free time* (default value: 20s) is the earliest time to release a rate-limited aggregate after it goes below the limit imposed on it. $[p^5_{sys}]$ *Rate limit time* (default value: 30s) determines how long a newly identified aggregate must be rate-limited. After the period, the router may release an aggregate. $[p^6_{sys}]$ *Maximum number of sessions* (default value: 3) determines the maximum number of aggregates the rate limiter can control. $[p^7_{sys}]$ *Aggregate pattern* (default value: destination address prefix) determines which kinds of properties will be used to identify aggregates.

$\odot$(d) A^{bad}_{sub} is determined by the following parameters: (d.1) The set of victims, i.e., V. (d.2) the *number of zombies*. For simplicity, we assume each zombie does the same thing. (d.3) The *location* of each zombie. (d.4) The *traffic volume* generated by each zombie. (d.5) The *traffic pattern* used by each zombie.

$\odot$(e) $p^{type}_{sys}(bad) = \theta$. In this paper, we set $\theta = 0.01$.

$\odot$(f) For each outcome o of a game play, $u_{sub}(o; good) = B_{lo}/B_{lw}$ and $u_{sub}(o; bad) = \alpha B_{ao}/B_N + (1-\alpha)(1-B_{lo}/B_{lw})$, where B_{lo} is the bandwidth occupied by the legitimate users; B_{lw} is the bandwidth that legitimate users want to occupy; B_{ao} is the bandwidth occupied by the attacker; B_N is the bandwidth capacity. For simplicity, B_{lo}, B_{lw}, B_{ao}, and B_N are all measured based on the incoming links to the victims, as shown in Figure 2. Note that B_{ao}/B_N indicates the absolute impact of the attack on the (whole) network, while $1 - B_{lo}/B_{lw}$ indicates the relative impact of the attack on legitimate users. α is the weight that balances these two aspects. In this study we let $\alpha = 0.5$.

$\odot$ (g) $u_{sub}(o)$ is defined in the standard way.

Simulation. In order to obtain concrete AIOS inferences of real world DDoS attackers, we have done extensive simulations on the game plays specified above using ns-2 [1]. The network topology of our experiments is shown in Figure 2, which is the same as the topology used in [15]. There are 64 source hosts and 4 levels of routers. Except the routers at the lowest level, each router has a fan-in of 4. The link bandwidths are shown in the figure. Each router uses a ns-2 pushback-module to enforce the pushback mechanism. It should be noticed that although there can be multiple victims staying on different subnets, we assume all the victims share the same incoming link, namely $R1.0 - R0.0$.

In our experiments, A^{sys} is materialized as follows. A^{sys} includes 10 defense strategies. The default value combination of $\{p^1_{sys}, ..., p^7_{sys}\}$ is the 7th defense strategy. The 1st strategy is the same as the 7th except that $p^1_{sys} = 4s$. The 2nd is the same as the 7th except that the cycle time is 10s. The 3rd is different in that that the target drop rate is 0.03. The difference of the 4th is that the target drop rate is 0.07. The 5th is different in that the free time is 10s. The 6th is different in that the free time is 30s. The 8th is different in that the rate limit time is 15s. The 9th is different in that

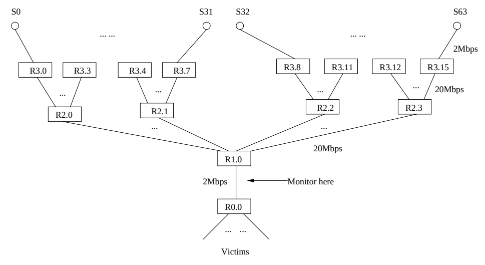
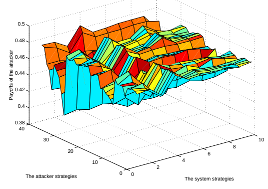


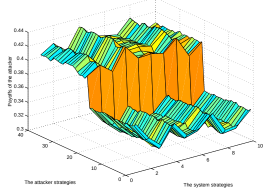
Figure 2: Network topology

the rate limit time is 50s. The 10th is different in that the maximum number of sessions is 5.

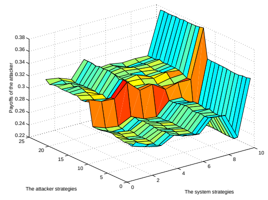
In our experiments, A^{bad}_{sub} is materialized as follows. (a) We set the number of zombies as 12 (FewBad) or 32 (ManyBad). (b) The zombies are randomly chosen from the 64 hosts. (c) The traffic volume and pattern are determined based on several real world Internet traces posted at *http://ita.ee.lbl.gov/html/traces.html*. These traces show three typical volume patterns when there are no attacks: RATE1 = 67.1kbps (the rates to a web site during the rush hour); RATE2 = 290kbps (the average rates from an Intranet to the Internet); RATE3 = 532kbps (the rates from an Intranet to the Internet during the rush hour). Based on these statistics, we let the total traffic volume of the good source hosts to the victims be 67.1kbps, 290kbps, or 532kbps. (Note that here we do not count the packets that go from a good host to a destination that is not a victim.) Since the aggregate pattern is "destination address prefix", every good packet to the victims will be put into the same set of rate-limiting aggregates (if any) as the DDoS packets. Hence, such good packets are called *poor* packets. To illustrate, when the poor traffic volume is 290kbps and when there are 12 zombies, each good host will send out 290/52 bps traffic to the victims besides the good traffic sent to other destinations. (d) We determine the total attack traffic volume based on a parameter called the *bad-to-poor ratio*. For example, when the ratio is 30 and the poor traffic volume is 290kbps, the total attack traffic volume is 30*290 bps. Moreover, if there are 32 zombies, each zombie will send out 30*290/32 bps traffic to the victims. (e) When the poor traffic volume is 67.1kbps or 290kbps, we let the ratio be 30, 35, 40, 45, or 50. When the poor volume is 532kbps, we let the ratio be 30, 35, or 40. In this way, we totally get 13 possible attack traffic volumes. (f) The traces also show 4 kinds of traffic patterns. They are: Constant bits rate (CBR), Exponential (EXP), ICMP, and Mixed (i.e., half CBR and half ICMP). We let the attack traffic patterns be of these four types. (g) If we count the number of value combinations of these attack strategy parameters, we can know that there are 40 possible strategies under RATE1 or RATE2, and they are 24 possible strategies under RATE3. We number the attack strategies as follows. In the first 20 (12) strategies of the 40 (24) strategies, the number of zombies is FEW. In the second 20 (12) strategies, the number of zombies is MANY. Within each 20 (12) strategy group, the first 5 (3) strategies use CBR traffic, the 2nd use Exponential traffic, the 3rd use ICMP traffic, and the 4th use Mixed traffic. Within each such 5 (3) strategy group, the strategies are ordered according to the bad-to-poor ratio, and the order is 30, 35, 40, 45 and 50



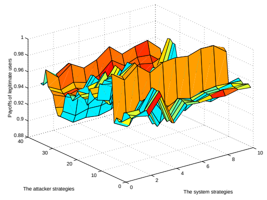
(a) rate1



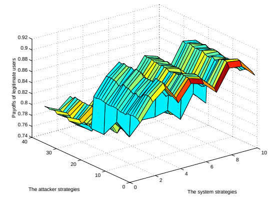
(b) rate2



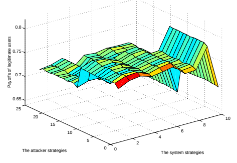
(c) rate3



(d) rate1



(e) rate2



(f) rate3

Figure 3: The attacker's and legitimate users' payoffs under different defense and attack strategies

(30, 35 and 40). Finally, it should be noticed that when the system takes strategy 10, the attacker will target 4 victims in each of the 40 (24) strategies, although in every other case the attacker will target only one victim.

Moreover, we assume the good traffics that do not go to the victims will not cause any congestion by themselves. Hence, they will not be involved in any aggregate in our experiments and their influence can be neglected.

Payoffs and their AIOS implications. Figure 3 shows the attacker's and legitimate users' payoffs under different network scenarios (i.e., poor traffic volumes). We found that $\odot$(a) The attacker's payoffs are dependent upon not only attack strategies, but also network scenarios and defense postures. For example, when the poor traffic volume is low and the target drop rate is 0.07, the attacker prefers using many zombies, while in some other situations the attacker prefers using few zombies.

$\odot$(b) From Figure 3, we found that the bad-to-poor ratio does not affect the attacker's payoffs much in each network scenario. It seems when the attacker sends more packets to the victims, the attacker should occupy more bandwidth and get more payoffs. However, based on the results, the attack traffic volume does not affect the payoffs much. This indicates that pushback will still be effective even under intense DDoS attacks.

$\odot$(c) Regarding the traffic pattern, the attacker earns significantly less payoffs when the traffic pattern is ICMP, there are many zombies, and the poor traffic rate is 67.1kbps (see Figure 3(a)). Under other situations, traffic patterns have little impact on the attacker's payoffs.

$\odot$(d) Except the case when the traffic pattern is ICMP and the poor traffic rate is low, the attacker gets high pay-

Table 1: Nash equilibria Strategies

system strategy	LS	AS
dp0.03	rate1, MF	MANY, 40, CBR, OA
dp0.03	rate1, MM	MANY, 45, CBR, OA
sess5	rate1, MM	FEW, 50, EXP, MA
sess5	rate1, MF	MANY, 45, CBR, MA
sess5	rate1, MM	MANY, 50, CBR, MA
sess5	rate1, MM	MANY, 35, ICMP, MA
sess5	rate1, MM	MANY, 30, EXP, MA
sess5	rate2, MM	MANY, 50, CBR, MA
dp0.03	rate3, MM	MANY, 35, EXP, OA
dp0.03	rate3, MM	MANY, 30, Mixed, OA
sess5	rate3, MM	MANY, 40, EXP, MA

offs when using many zombies. When using few zombies, some poor traffic may not share the same route with the attack traffic and the poor traffic may be "protected" by the routers in such a way that minimum dropping is suffered. Hence, when using fewer zombies, more attack packets can be dropped. Therefore, the attacker should intend to use many zombies in most cases.

$\odot$(e) The payoffs earned by the system and the legitimate users are dependent not only on the defense strategies and the legitimate users' strategies, but also on the attack strategies. For example, legitimate users earn higher payoffs when there are few zombies and many good hosts, and the drop rate is 0.03.

Nash equilibria and their AIOS implications. We get 48 Nash equilibria based on the payoffs we got when the relative error is 0.005[§]. Some of them are shown in Table 1, where "LS" means the legitimate users' strategy; "AS" means the attacker's strategy; "OA" means the attacking traffic has one aggregate; "MA" means the attacking traffic has multiple aggregates; "FM" means FEWGOODMANYPOOR, that is, there are many poor hosts (i.e., hosts that send packets to the victims) and few unaffected hosts (i.e., hosts that send packets to another destination); "dp0.03" means that the target-drop-rate is 0.03; and "sess5" means that the maximum number of sessions is 5.

We found that several interesting AIOS inferences can be obtained from the distributions of the Equilibria. In particular,

$\odot$ (a) in terms of the traffic pattern, the distribution is {0.22(CBR), 0.32(EXP), 0.19(ICMP), 0.27(Mixed)}. This indicates that the attacker would most likely use EXP traffic, since most Nash equilibria occur when the attack traffic pattern is EXP (based on UDP).

$\odot$ (b) The distribution under bad-to-poor ratio is {0.25(30), 0.31(35), 0.07(40), 0.07(45), 0.30(50)}, that is, the most unlikely used ratio is 40.

$\odot$ (c) The distribution under different combinations of the number of zombies, poor hosts and unaffected hosts is {0(FFF), 0(FFM), 0.01(FMF), 0(FMM), 0.12(MFF), 0(MFM), 0.24(MMF), 0.63(MMM)}. This indicates that the attacker will typically use many zombies.

$\odot$ (d) The distribution under different defense strategies indicates that most Nash equilibria occur when the target-drop-rate is 0.03 (22%); or when the max-number-of-sessions is 5 (57%). This indicates that to be more resilient, the system can increase the number of sessions and decrease the target-drop-rate.

[§]When we reduce the computing error, we get fewer Nash equilibria but the distributions of the Nash equilibria are still almost the same.

6. STATUS AND FUTURE WORK

In this paper, we have developed an incentive-based conceptual framework for AIOS modeling. We have developed a game theoretic formalization of the conceptual framework. We have investigated how to practically model and infer AIOS. And we have done a real world case study to gain more insights about how to infer AIOS.

Nevertheless, our work in inferring AIOS is still preliminary and several important research issues need to be further explored in order to get better AIOS inferences. In particular, (a) model level inference accuracy analysis and sensitivity analysis that can model and predict the influence of incomplete information, asymmetric information (between the attacker and the system), and uncertainty; (b) approximate algorithms that can do optimal, quantitative tradeoffs between inference precision and efficiency during Nash equilibria estimation; (c) AIOS inference models beyond Bayesian games (i.e., the ones identified by our taxonomy).

Acknowledgement. This work was supported by DARPA and AFRL, AFMC, USAF, under award number F20602-02-1-0216, and by Department of Energy Early Career PI Award.

7. REFERENCES

[1] The network simulator ns-2. http://www.isi.edu/nsnam/ns/.

[2] H. Browne, W. A. Arbaugh, J. McHugh, and W. L. Fithen. A trend analysis of exploitations. In *Proc. 2001 IEEE Symposium on Security and Privacy*, pages 214–229, May 2001.

[3] R. Browne. C4i defensive infrastructure for survivability against multi-mode attacks. In *Proc. 21st Century Military Communications - Architectures and Technologies for Information Superiority*, 2000.

[4] D. Buike. Towards a game theory model of information warfare. Technical report, Airforce Institute of Technology, 1999. Master's Thesis.

[5] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[6] V. Conitzer and T. Sandholm. Complexity results about nash equilibria. Technical report, Carnegie Mellon University, 2002. CMU-CS-02-135.

[7] F. Cuppens and A. Miege. Alert correlation in a cooperative intrusion detection framework. In *Proc. IEEE Symposium on Security and Privacy*, 2002.

[8] H. Debar and A. Wespi. Aggregation and correlation of intrusion detection alerts. In *Recent Advances in Intrusion Detection, LNCS 2212*, pages 85–103. 2001.

[9] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A bgp-based mechanism for lowest-cost routing. In *Proc. 21st ACM Symposium on Principles of Distributed Computing*, 2002.

[10] A. M. Fink. Equilibrium in a stochastic n-person game. *Journal of Science in Hiroshima University, Series A-I*, (28):89–93, 1964.

[11] L. A. Gordon and M. P. Loeb. Using information security as a response to competitor analysis systems. *Communications of the ACM*, 44(9):70–75, 2001.

[12] T. Groves. Incentives in teams. *Econometrica*, 41:617–663, 1973.

[13] J. P. Hespanha and S. Bohacek. Preliminary results in routing games. In *Proc. 2001 American Control Conference*, 2001.

[14] J. Nash. Equilibrium Points in n-Person Games *Proceedings of the National Academy of Sciences*, 36, 1950.

[15] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against ddos attacks. In *Proc. 2002 Network and Distributed Systems Security*, 2002.

[16] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *Proc. 17th International Joint Conference on Artificial Intelligence*, 2001.

[17] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys*, 26(3), 1994.

[18] P. Liu, S. Jajodia, and C.D. McCollum. Intrusion confinement by isolation in information systems. *Journal of Computer Security*, 8(4):243–279, 2000.

[19] T.F. Lunt. A Survey of Intrusion Detection Techniques. *Computers & Security*, 12(4):405–418, June 1993.

[20] K. Lye and J. M. Wing. Game strategies in network security. In *Proc. 15th IEEE Computer Security Foundations Workshop*, 2002.

[21] D. Malkhi and M. K. Reiter. Secure execution of java applets using a remote playground. *IEEE Transactions on Software Engineering*, 26(12), 2000.

[22] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory.* Oxford University Press, 1 edition, 1995.

[23] J. McHugh. Intrusion and intrusion detection. *International Journal of Information Security*, (1):14–35, 2001.

[24] M. Mesterton-Gibbons. *An Introduction to Game-Theoretic Modeling.* Addison-Wesley Publishing Company, 1992.

[25] B. Mukherjee, L. T. Heberlein, and K.N. Levitt. Network intrusion detection. *IEEE Network*, pages 26–41, June 1994.

[26] P. Ning, Y. Cui, and D. S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *ACM Int'l Conf. on Computer and Communications Security*, 2002.

[27] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35, 2001.

[28] P. F. Syverson. A different look at secure distributed computation. In *Proc. 10th IEEE Computer Security Foundations Workshop*, 1997.

[29] F. Thusijsman. *Optimality and Equilibria in Stochastic Games.* Gentrum voor Wiskunde en Information, Amsterdam, 1992.

[30] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[31] X. Wang and M. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *IEEE Symposium on Security and Privacy*, 2003.

[32] M. P. Wellman and W. E. Walsh. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35, 2001.

[33] C. Zou, W. Gong, and D. Towsley. Code red worm propagation modeling and analysis. In *Proc. ACM Conference on Computer and Communication Security*, 2002.

[34] J. Xu and W. Lee. Sustaining availability of web services under distributed denial of service attacks. In *IEEE Transactions on Computer*, 52(4):195–208, February 2003.

APPENDIX

A. A SIMPLE REVIEW OF GAME THEORY

The *normal-form representation* of an n-player game specifies the players' *strategy* spaces S_1, ..., S_n and their *payoff* functions u_1, ..., u_n. We denote this game by $G = \{S_1, ..., S_n; u_1, ..., u_n\}$. In this game, the strategies $(s_1^*, ..., s_n^*)$ are a *Nash equilibrium* if, for each player i, s_i^* is (at least tied for) player i's best response to the strategies specified for the n-1 other players, (s_1^*, ..., s_{i-1}^*, s_{i+1}^*, ...,s_n^*). That is, s_i^* solves $\max_{s_i \in S_i}$ $u_i(s_1^*$, ..., s_{i-1}^*, $s_i, s_{i+1}^*, ..., s_n^*)$.

A *pure strategy* for player i is an element of set S_i. Suppose $S_i = \{s_{i1}, ..., s_{ik}\}$, then a *mixed strategy* for player i is a probability distribution $p_i = (p_{i1}, ..., p_{ik})$, where $o \leq p_{ik} \leq 1$ for $k = 1, ..., K$ and $p_{i1} + ... + p_{ik} = 1$. Although a game does not always have a pure strategy Nash equilibrium, Nash [14] proved that a game always has at least one mixed strategy Nash equilibrium.

The static Bayesian game theory is mentioned in Section 5.2. Note that a Bayesian Nash equilibrium can be defined in a way very similar to a normal Nash equilibrium.

Given a stage game G (e.g., a static (Bayesian) game), let $G(T)$ denote the *finitely repeated game* in which G is played T times, with the outcomes of all preceding plays observed before the next play begins. The payoffs for $G(T)$ are simply the sum of the payoffs from the T stage games. If the stage game G has a unique Nash equilibrium then, for any finite T, the repeated game $G(T)$ has a unique subgame-perfect outcome: the Nash equilibrium of G is played in every stage.

Moreover, in a finitely repeated game $G(T)$, a player's *multi-stage strategy* specifies the action the player will take in each stage, for each possible history of play through the previous stage. In $G(T)$, a *subgame* beginning at stage $t+1$ is the repeated game in which G is played $T-t$ times, denoted $G(T-t)$. There are many subgames that begin at stage $t+1$, one for each of the possible histories of play through stage t. A Nash equilibrium is *subgame-perfect* if the player's strategies constitute a Nash equilibrium in every subgame.

Finally, a standard formal definition of stochastic games is as follows. An n-player stochastic game Γ is a tuple $\langle$ $S, A^1, ..., A^n, r^1$, ..., r^n, p $\rangle$, where S is the state space, A^i is the action space of player i for $k = 1, ..., n$, $r^i : S \times A^1 \times ... \times A^n \rightarrow R$ is the payoff function for player i, $p : S \times A^1 \times ... \times A^n \rightarrow \nabla$ is the transition probability map, where ∇ is the set of probability distributions over state space S [29]. In Γ, a *strategy* $\pi = (\pi_0, ..., \pi_t, ...)$ is defined over the entire course of the game, where π_i is called the *decision rule* at time t. A decision rule is a function $\pi_t : \mathbf{H}_t \rightarrow \sigma(A^i)$, where $\mathbf{H}_t$ is the space of possible histories at time t, with each $H_t \in \mathbf{H}_t$, $H_t = (s_0, a_0^1, ..., a_0^n, ..., s_{t-1}, a_{t-1}^1, ..., a_{t-1}^n, s_t)$, and $\sigma(A^i)$ is the space of probability distributions over agent i's actions. π is called a *stationary strategy* if $\pi_t = \overline{\pi}$ for all t, that is, the decision rule is independent of time. Otherwise, π is called a *behavior strategy.* In Γ, a Nash equilibrium point is tuple of n strategies $(\pi_*^1, ..., \pi_*^n)$ such that for all $s \in S$ and $i = 1, ..., n$, $v^i(s, \pi_*^1, ..., \pi_*^n) \geq v^i(s, \pi_*^1, ..., \pi_*^{i-1}, \pi^i, \pi_*^{i+1}, ..., \pi_*^n)$ for all $\pi^i \in Pi^i$, where where Π_i is the set of strategies available to agent i.