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

PENG LIU and WANYU ZANG
Pennsylvania State University
and
MENG YU
Monmouth University

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 inferring 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 interdependency 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 attack strategies 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

Additional Key Words and Phrases: Attacker intent and strategy modeling, attack strategy inference, game theory

1. INTRODUCTION

The ability to model and infer attacker intent, objectives, and strategies (AIOS) may dramatically advance the state of the 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 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.
Authors' addresses: P. Liu and W. Zang, School of Information Science, Pennsylvania State University, University Park, PA 16802; email: pliu@ist.psu.edu; M. Yu, Department of Computer Science, Monmouth University, West Long Branch, NJ 07764.

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 multiphase, well-planned, carefully hidden 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 [Malkhi and Reiter 2000] and isolation [Liu et al. 2000], 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 [Cuppens and Miege 2002; Ning et al. 2002] 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 [Debar and Wespi 2001], 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 for 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 [Landwehr et al. 1994] and attack effects [Browne et al. 2001; Zou et al. 2002], existing AIOS inference techniques are ad hoc and system or application specific [Gordon and Loeb 2001; Syverson 1997]. 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 interestingly, 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.[1] 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 inferring 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 interdependency 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 attack strategies 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 addresses show to infer AIOS. In Section 6, we use a specific case study to show how attack strategies can be inferred in real-world attack–defense scenarios. In Section 7, 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 Syverson [1997], Syverson talks about "good" nodes fighting "evil" nodes in a network and suggests using stochastic games for reasoning and analysis. In Lye and Wing [2002], Lye and Wing 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 Browne [2000], 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

[1]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.

detector or not. Depending on the combined attack and defense actions, each outcome has different costs. In Burke [1999], Burke studies the use of repeated games with incomplete information to model attackers and defenders in information warfare. In Hespanha and Bohacek [2001], 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 Xu and Lee [2003], 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, for example, 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 Gordon and Loeb [2001], 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 Gordon and Loeb [2001] to protect a game-theoretic system.

The computational complexity of game-theoretic analysis is investigated in several research. For example, Conitzer and Sandholm [2002] show 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, Koller and Milch [2001] show 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 design* theory [Clarke 1971; Groves 1973; Vickrey 1961] can be very valuable in solving a variety of Internet computing problems such as routing, packet scheduling, and web

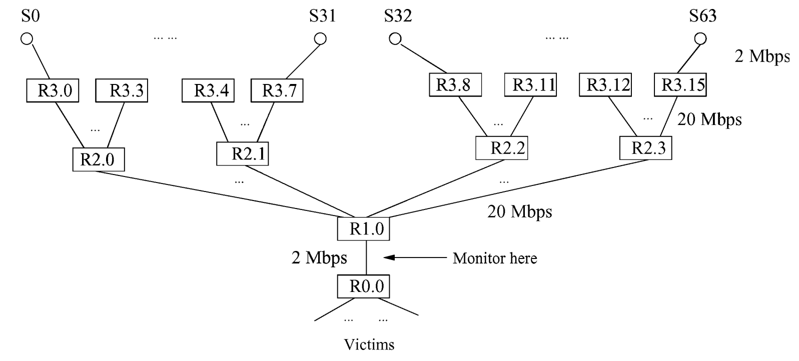


Fig. 1. Network topology.

caching [Feigenbaum et al. 2002; Nisan and Ronan 2001; Wellman and Walsh 2001]. Although when market-based mechanisms are used to defend against attackers [Wang and Reiter 2003], 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 [Lunt 1993; McHugh 2001; Mukherjee et al. 1994]. 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 the attacks that may happen in the future.

3. AN INCENTIVE-BASED FRAMEWORK FOR AIOS MODELING

In this section, we present an incentive-based conceptual model of attacker intent, objectives, and strategies. Our model is quite abstract. To make our presentation more tangible, we will first present the following example, which will be used throughout the paper to illustrate our concepts.

Example 1. In recent years, Internet distributed denial-of-service (DDoS) attacks have increased in frequency, severity, and sophistication and become a major security threat. When a DDoS attack is launched, a large number of hosts (called *zombies*) "controlled" by the attacker flood a high volume of packets toward the target (called the *victim*) to downgrade its service performance significantly or make it unable to deliver any service.

In this example, we would model the intent and objectives and infer the strategies of the attackers that enforce *brute-force* DDoS attacks. (Although some DDoS attacks with clear signatures, such as SYN flooding, can be effectively countered, most DDoS attacks without clear signatures, such as brute-force DDoS attacks, are very difficult to defend against since it is not clear which packets are DDoS packets and which are not.) An example scenario is shown in Figure 1 where many zombies (i.e., a subset of source hosts $\{S_0, \ldots, S_{64}\}$) are flooding a couple of web sites (i.e., the *victims*) using normal HTTP requests. Here, $Rx.y$ denotes a router; the bandwidth of each type of links is marked; and the web sites may stay on different subnets.

Although our modeling and inference framework can handle almost every DDoS defense mechanism, to make this example more tangible, we select *pushback* [Ioannidis and Bellovin 2002], a popular technique, as the security mechanism. Pushback uses *aggregates*, that is, a collection of packets from one or more flows that have some properties in common, to identify and *rate limit* the packets that are most likely to cause congestion or DoS. Pushback is a coordinated defense mechanism that typically involves multiple routers. To illustrate, consider Figure 1 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*.

The effectiveness of pushback can be largely captured by four bandwidth parameters associated with the incoming link to the victims (i.e., the link that connects $R1.0$ and $R0.0$): (a) B_N, the total bandwidth of this link; (b) B_{ao}, the (amount of) bandwidth occupied by the DoS packets; (c) B_{lo}, the bandwidth occupied by the legitimate packets; (d) B_{lw}, the bandwidth that the legitimate users would occupy if there are no attacks. For example, pushback is effective if after being enforced B_{ao} can become smaller and B_{lo} can become larger.

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 any kind (e.g., a network system, a distributed system, a database system). We call it the *system* for short. For example, in Example 1 the system consists of every router on a path from a zombie to a victim. 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, and so on. An attack will cause some *effects* on the system, that is, transforming the system from one *state* to another state. For example, in Example 1 the main attack effects are that many legitimate packets could not reach the victims.

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. For example, in Example 1 a router sending out a pushback message is a defense action, and this action can trigger the receiving router(s) to take further defense actions. 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*. For example, in Example 1 a packet arriving at a router is an event. When there is no congestion at the router, this event will not activate any security mechanism. However, when this event leads to "the detection of a congestion" (i.e., the condition), pushback will be activated. And it is clear that whether this condition can be satisfied is dependent upon the accumulated effects of the previous DoS packets arriving at the router. Finally, a *defense posture* of the system is defined by the set of security mechanisms and the ways they are activated. For example, in Example 1, pushback may be configured to stay at various defense postures based on such parameters as congestion thresholds and target drop rate, which we will explain in Section 3.3 shortly.

The attacker-system relation has several unique characteristics (or properties) that are important in illustrating the principles of our attack strategy inference framework. These properties are as follows.

—*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. We will define the notion of strategy shortly. And we will use concrete attack and defense strategies derived from Example 1 to illustrate this property shortly in Section 3.3.

—*Uncertainty Property.* The attacker usually has *incomplete* information or *knowledge* about the system, and vice versa. For example, in Example 1 the attacker usually has uncertainty about how Pushback is configured when he or she enforces a DDoS attack.

3.1 Incentive-Based Attacker Intent Modeling

Different attackers usually have different intents even when 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 damage, 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 yet simple intent models. In particular, we believe that the concept of economic "incentives" 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. For an example, in Example 1 the incentives for the attacker can be the amount of DoS suffered by the legitimate users. For another example, the incentives for an attacker that enforces a worm attack can be the amount of network resources consumed by the worm's scanning packets plus the amount of DoS caused on certain type of services. 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. For another example, in Example 1, the attacker's incentives can be quantified by two metrics: (a) B_{ao}/B_N, which indicates the absolute impact of the DDoS attack; and (b) B_{lo}/B_{lw}, which indicates the relative availability impact of the attack. Accordingly, the attacker's intent is to maximize B_{ao}/B_N but minimize B_{lo}/B_{lw}. 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 differently. As a result, very misleading attacker strategy 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 about. 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), nonrepudiation, and authentication. For example, in Example 1 the major security metrics of the system are (a) B_{lo}, which indicates the absolute availability provided by the system; and (b) B_{lo}/B_{lw}, which indicates the relative availability provided by the system. 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.) For example, the metric vector for the system in Example 1 can be simply defined as $\langle B_{lo}, B_{lo}/B_{lw}\rangle$. At time t, the measurements associated with the system's metric vector are called the *security vector* of the system at time t, denoted by 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) may be measured by $degradation(V_{t_1}^s, V_{t_2}^s)$, which basically computes the *distance* between the two security vectors. For example, in Example 1 assume the security vector is $V_{t_1}^s = \langle 1000\,\text{Mbps}, 100\%\rangle$ before the attack and $V_{t_2}^s = \langle 50\,\text{Mbps}, 5\%\rangle$ after the attack, then $degradation\,(V_{t_1}^s, V_{t_2}^s) = \langle -950\,\text{Mbps}, -95\%\rangle$.

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. For example, in Example 1, a DoS packet indicates almost nothing about the attacker's intent which can only be seen after some DoS effects are caused.

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 malicious 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 *noncost 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. In addition, risk is typically a cost constraint. (b) Noncost constraints are constraints on things that the attacker cannot buy such as religion-based constraints and top secret attacking tools that the attacker may never be able to "buy."

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 punished. Based on the relationship between incentives and costs, we classify attackers into two categories: (a) *rational attackers* have concerns about the costs (and risk) 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 noncost 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 values which captures both attacking resources and risk.

To illustrate, let us revisit Example 1. The attacker's total incentives may be measured by $\alpha B_{ao}/B_N + (1-\alpha)\,(1 - B_{lo}/B_{lw})$, where α determines how the attacker weighs the two aspects of the impact of the DDoS attack. The attack's costs in this example are not much, though the attacker needs a computer and Internet access to "prepare" the zombies and the needed controls. The cost will become larger when the risk of being traced back is included. Let us assume the cost is a constant number η. Then the attacker's utilities can be measured by $\alpha B_{ao}/B_N + (1-\alpha)\,(1-B_{lo}/B_{lw}) - \eta$, and the attacker's objective can be quantified as Max $\alpha B_{ao}/B_N + (1-\alpha)\,(1 - B_{lo}/B_{lw})$.

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

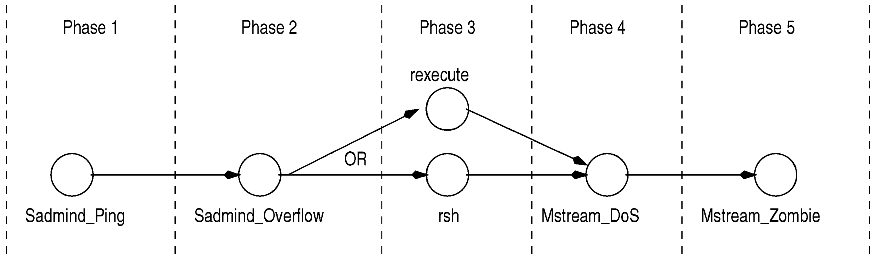


Fig. 2. Phases in *Mstream* DDoS attacks.

as reconnaissance, probe and attack, toehold, advancement, stealth (spreading quietly), 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.

Example 2. *Mstream* DDoS attacks, a common type of DDoS attacks, have several phases, which are shown in Figure 2. In phase 1, the attacker uses *Sadmind_Ping* (attacking tools or commands) to find out the vulnerable *Sadmind* services. In phase 2, the attacker uses *Sadmind_Over flow* to break into the hosts that run these vulnerable *Sadmind* services. In phase 3, the attacker uses *rsh* or *rexecute* to install the *mstream* daemon and master programs and the needed backdoors. In phase 4, the attacker uses *Mstream_DoS* to send commands to activate the master programs (via the backdoors). In phase 5, the DDoS master programs communicate with the daemon or zombie programs to ask them to send out a large number of DoS packets.

Let us assume that the attack scenario is shown in Figure 1 and that all of the 64 source hosts run vulnerable *Sadmind* services. Then two simple attack strategies can be specified as follows: (A1) *Sadmind_Ping* and *Sadmind_Over flow* the 64 hosts in phase 1 and phase 2, respectively; use *rsh* in phase 3 and install a master program on S_1 but a daemon program on each of the 64 hosts; use *Mstream_DoS* in phase 4; ask only 10 zombies to send CBR (constant bit rate) DoS traffic to a single web site (the victim) in the speed of 201.3 kbps per zombie. (A2) The same as A1 except for using *rexecute* in phase 3 and asking 30 zombies to send CBR DoS traffic to the victim in the speed of 67.1 kbps.

Similarly, two simple defense strategies can be specified as follows: (D1) Take no defense action in the first four phases. Enforce pushback (shown in Example 1) in phase 5 and set the target drop rate for each router (i.e., the upper-bound drop rate of the router's output queue) as 7%, while keeping all the other configuration parameters in default values. (D2) The same as D1 except that setting the target drop rate as 3%.

Note 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." For example, in Example 2 an attack strategy may include the following rule: when *rsh* is disabled by the system, use *rexecute*. 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. For example, in Example 2 both A1 and A2 are static attack strategies.

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, deactivate, 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,[2] the system's strategy space is usually bounded.

A key issue in modeling attacker strategies is how to compare two attack strategies and tell which one is better (for the attacker). Based on the purpose of attack strategies, 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 determining which strategy can yield 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 determining which strategy can yield 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 determining which strategy can cause more degradation to the system's security vector. However, the answer to this question is in general determined by the defense strategy that the system will take, since different battles may lead to different amount of security degradation. To illustrate, let us revisit Example 2. Based on our simulations (done in Section 6), we found that when D1 is taken by the system, A2 is better than A1 (i.e., causing more security degradation); however, when D2 is taken by the system, A1 is better than A2. More interestingly, we found that when A1 is taken by the attacker, D2 is better than D1; however, when A2 is taken, D1 is better than D2.

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

P1: IAZ
ACMJ030-01 ACM-TRANSACTION February 7, 2005 23:34

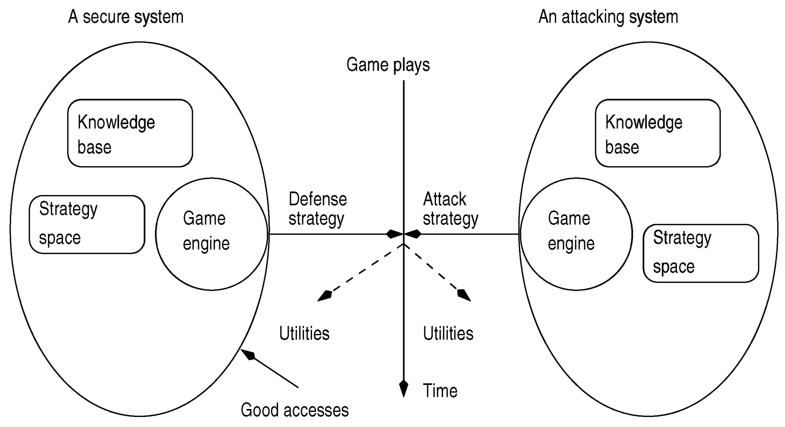


Fig. 3. A game-theoretic formalization.

The above discussion not only confirms the strategy-interdependency property, but also implies the following property of attack strategy inference:

—*Dual Property.* (a) Given two attack (defense) strategies, determining which one is a better attack (defense) strategy is dependent on the defense (attack) strategies the system (attacker) is going to take. (b) Each type of information useful for the attacker (system) to choose a good attack (defense) strategy will also be useful for the system (attacker) to choose a good defense (attack) strategy.

4. A GAME-THEORETIC FORMALIZATION

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 3, 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. For example, in Example 1 the service part mainly includes the hardware and software components (within the routers) that route packets; the pushback components belong to the protection part.
—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.
  —The two *players*, namely the (secure) system and the attacking system. Note that the "real" player for the system is the set of security mechanisms.
  —The *game type* (e.g., a Bayesian game or a stochastic game) and the set of type-specific parameters of the game.
  —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 $S^a = \{s_1^a, \ldots, s_m^a\}$, where s_i^a is an attack strategy. The system's strategy space is denoted as $S^d = \{s_1^d, \ldots, s_m^d\}$, 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.
  —A set of game *plays*. A play is a function $pl_i : S^a \times S^d \rightarrow O$, 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. For example, in Example 2 a game play between the DDoS attacker and the network may involve attack strategy A1 and defense strategy D2.
  —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 $u^a : S^a \times S^d \rightarrow R$, 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 as $cost(s_i^a)$. 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 $u^d : S^a \times S^d \rightarrow R$. 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.
  —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 3. 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., noncooperative). 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 element 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) noncooperative game theory is the primary tool to handle *strategic interdependence* [Mas-Colell et al. 1995], which is the fundamental property of the attacker-system relation; (b) 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; (c) 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.

5. GAME-THEORETIC AIOS INFERENCE

As we mentioned in previous sections, the ability to infer attacker intent, objectives, and strategies (in information warfare) may dramatically advance the literature of risk assessment, harm prediction, and proactive cyber defense. In the previous section, we show how to model AIOS via a game-theoretic formalization. In this section, we address how to exploit such formalizations to infer AIOS. In particular, we tackle two types of AIOS inference problems, which are illustrated below.

Type A—Infer Attack Strategies. Given a specific model of attack intent and objectives, infer which attack strategies are more likely to be taken by the attacker. The previous presentation implies the following pipeline in inferring attack strategies:

(1) Make assumptions about the system and the (types of) attacks that concern the system. Note that practical attack strategies 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 attack strategy inference model that will be developed, then develop the model accordingly.
(5) Compute the set of Nash equilibrium strategies of the attack strategy 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 trade-offs properly using some (semantics-based) approximate algorithms.

(6) Validate the inferences generated in step 5. The relevant tasks include but are not limited to *accuracy analysis* (i.e., how accurate the inferences are) and *sensitivity analysis* (i.e., how sensitive the inferences are to some specific model parameters). The relevant validation techniques include but are 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.

Type B—Infer Attacker Intent and Objectives. Based on the attack actions observed, infer the intent and objectives of the attacker in enforcing the corresponding attack. To a large degree, the pipeline for inferring attacker intent and objectives is the reverse of that for inferring attack strategies. In particular, the pipeline has two phases: the *learning* phase and the *detection* phase, which are as follows.

—In the learning phase, do the same thing in step 1 as the previous pipeline. In step 2, identify and classify the possible models of attacker intent and objectives into a set of representative attacker intent and objectives models. Then model the attack strategies for each of the representative models. In steps 3–5, do the same thing as the previous pipeline. As a result, a (separate) set of attack strategy inferences will be generated for each of the representative AIOS models built in step 2.

—In the detection phase, once an attack strategy is observed, match the observed attack strategy against the inferred attack strategies generated in the learning phase. Once an inferred attack strategy is matched, the corresponding attacker intent and objective model(s) will be the inference(s) of the real attacker's intent and objectives. (Note that sometimes an observed attack strategy may "match" more than one attacker intent and objective models.) Nevertheless, when none of the inferred attack strategies can be matched, go back to the learning phase and do more learning.

In summary, both type A and type B inference problems need a game-theoretic inference model, and we call such inference models *AIOS inference models* in general. As we will show shortly in Section 6, given a specific attack–defense scenario, once we have a good understanding of the attack, the defense, the attacker and the system, most steps of the two pipelines are fairly easy to follow, but the steps of determining the game type of the AIOS inference model are not naive and require substantial research. Therefore, in the following, before we show how an AIOS inference pipeline can be implemented in a real-world attack–defense scenario in Section 6, 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. For example, in Example 1 a system state captures such state information as the length of the packet queue(s) in each router. It should be noted that the system's state has nothing to do with the system's defense posture, which is determined by the configuration 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 resulting 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 can know the current attack state by analyzing the defense system and his attack strategies even before the attacks, 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 (i.e., false positives), the system may have wrong belief about the current attack state.

The relation between states and time 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 lifetime. When the lifetime of an attack is over, we make the corresponding attack state part of the history. The lifetime of an attack should involve some defense actions or operations, since when the life of the attack is over, the system should have already recovered from the attack in many possible ways, for example, replacing the system with a new system, repairing the damaged part of the system, and so on.

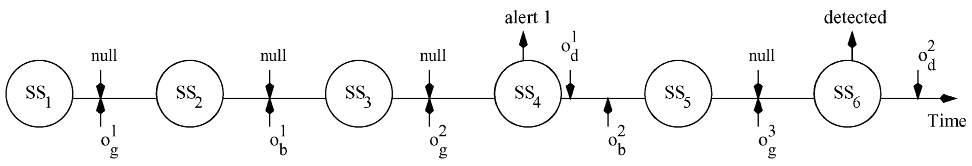


Fig. 4. An example battle between the attacker and system.

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. We use i as the action index, for example, the o_b^i is the ith action taken by the attacker.

—There must be either one attack action or one good access action between two adjacent states. (Two system states SS_i and SS_j are *adjacent* if there does not exist a third state that stays between SS_i and SS_j in the interleaved sequence of system states and actions.) 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.

Example 3. Consider the battle shown in Figure 4. Good access action o_g^1 transforms the system state from SS_1 to SS_2. Then attack action o_b^1 transforms SS_2 to SS_3 which is part of an attack state. We assume there is some latency in detection, so no defense action is taken until alert 1 is raised in system state SS_4. Hence, the effects of attack action o_b^1 are initially embodied in SS_3, and o_g^2, though a good action, may further spread the damage accidentally. When alert 1 is raised, since it may not be able to confirm an intrusion and since it may be a false alarm, the system is unsure whether SS_4 belongs to an attack state, and the system needs to determine whether a proactive defense action should be taken or the system should just wait until an intrusion is detected.[3] Suppose proactive defense is done here. After defense action o_d^1 is taken, o_b^1's direct and indirect effects may be removed, at least to some extent. However, o_b^2, a new attack action, may be simultaneously taken together with o_d^1, and a

[3]Note that false negatives may also affect the battle. In particular, the detection of an attack usually involves a sequence of alerts. When there are false negatives, some alerts will not be raised. As a result, the system can have more uncertainty about whether there is an intrusion and the response can become even less proactive.

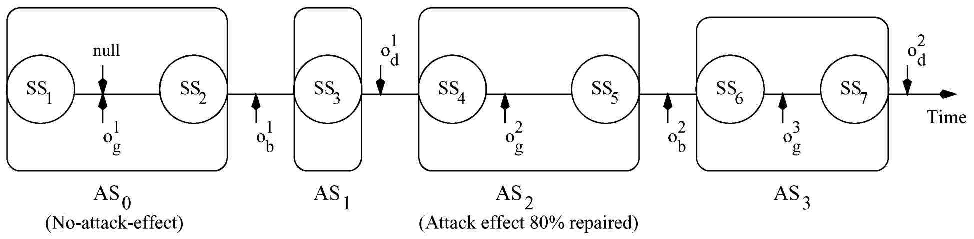


Fig. 5. A battle between the attacker and system under instant accurate intrusion detection.

new attack state is reached. Finally, after the intrusion is detected in SS_6 and o_d^2 is taken, the attack state may terminate.

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 battles between the attacker and the system can be modeled as follows.[4] Here, we model an intrusion as a sequence of attack actions, namely $I_j = \{o_b^1, o_b^2, \ldots, o_b^n\}$. 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.
—Between any two adjacent attack states, attack and/or defense actions can only be taken in three possible ways: (a) a single attack action is taken; (b) a single defense action is taken; or (c) a pair of attack and defense actions (o_b^i, o_d^i) are taken simultaneously by the attacker and the system, respectively. Two attack states AS_i and AS_j are adjacent if there does not exist a third attack state that stays between them in the interleaved sequence.
—The battle can involve several fights. A *fight* is composed of two adjacent attack states and a pair of attack and defense actions between them, denoted by (o_b^i, o_d^i). It is clear that (o_b^i, o_d^i) can transform the system from one attack state to another.

Example 4. Consider the battle shown in Figure 5, where we assume intrusion detection can be instantly done with accuracy. Within the initial attack state AS_0 which consists of two system states (i.e., SS_1 and SS_2), there is no attack effects, and a good action such as o_g^1 can transform one system state (e.g., SS_1) to another (e.g., SS_1). When attack action o_b^1 is taken, the attack state transits to AS_1, where some attack effects are caused. Since AS_1 can be instantly detected, the time interval between o_b^1 and defense action o_d^1 can be very short. After o_d^1 is taken, suppose 80% of the attack effects caused by o_b^1 is

[4]Note that since Definition 2 is mainly for theoretic analysis and meant to show how simple a battle model can theoretically be, it is OK to make this not very practical assumption.

repaired,[5] then a new attack state, that is, AS_2, is reached. Finally, note that after o_b^2 is taken, within the new attack state AS_3, a system state transition is possible before o_d^2 is taken, since it may take a while for the system to determine the proper defense action(s) to take.

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 battle, denoted by $u_d(o_b^i, o_d^i)$, is computable if we know which good actions are involved in the battle, 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 battle, 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 battles 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 that 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 6, and the taxonomy can be simply summarized as follows:

—In *region* 9, two types of *dynamic games* can be used together with primarily reactive defense, which are illustrated below. In both scenarios, the attack is composed of a sequence of highly correlated attack actions that are complementary to each other; and each attack action can be detected with agility. And it should be noticed that the goal of both the attacker and the system is to win the battle in a long run after a sequence of fights are finished.
  — If the attacker can clearly recognize each defense action and wants to see the effects of the current defense action against his latest attack action before choosing a new action, a dynamic observe-then-act game (with perfect information) can be used. In this game, the attacker and system take actions in turn, and at each move the player with the move knows the full history of the game play thus far. The theory of *backwards induction* [Mesterton-Gibbons 1992] can be used to compute the optimal

[5]Note that being able to accurately detect intrusions does not always mean being able to accurately identify all the attack effects and repair them.

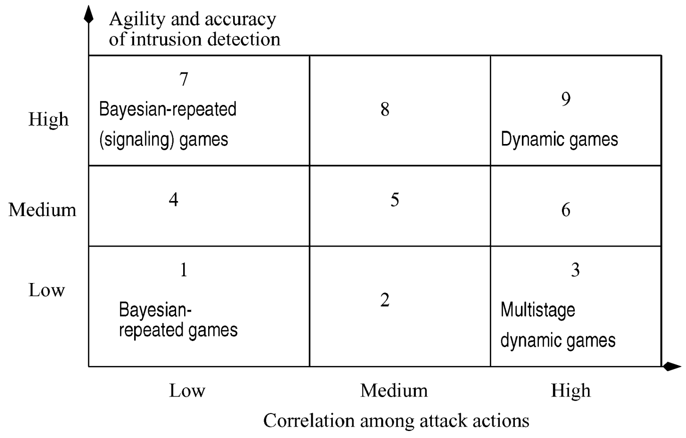

Agility and accuracy of intrusion detection
High
Medium
Low
7
Bayesian-repeated (signaling) games
8
9
Dynamic games
4
5
6
1
Bayesian-repeated games
2
3
Multistage dynamic games
Low
Medium
High
Correlation among attack actions

Fig. 6. A taxonomy of game-theoretic AIOS models.

attack/defense strategies. And the idea is that when an attack action o_b is followed by a defense action, the attacker will take the system's best response to o_b into account when he or she chooses the "best" o_b. Figure 5 shows an example play of this type of game. Finally, the defense should be primarily reactive, since each attack action can be detected with agility but it can be fairly difficult to predict the next action the attacker will take.

—If the attacker has substantial uncertainty in recognizing a defense action but is good at identifying an attack state, *multistage dynamic games* with *simultaneous moves* can be used. In this game, the first attack action and a *null* defense action are simultaneously taken in stage 1, the first defense action and the second attack action are simultaneously taken in stage 2, and so forth. In this scenario, observe-then-act games are not very suitable because the attacker could not identify the exact defense action and thus waiting for the current defense action to end will not help the attacker much in choosing the next attack action. Moreover, in general the optimal attack/defense strategies of this game can be captured by *subgame-perfect Nash equilibrium* (see Appendix). Finally, if the attack state transitions are probabilistic, *stochastic games*, a special type of multi stage dynamic games, should be used. When intrusion detection is highly effective, stochastic games become feasible. See that not only that each attack state can be accurately identified by the system with agility, which enables effective reactive defense, but also 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-repeated 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 with accuracy. In this case, Bayesian game models, which we will discuss shortly in Section 5.2, are a robust realistic solution, since they do not require accurate detection, and they do not require instant detection either. Finally, since the detection latency is not short, proactive defense can be more beneficial than reactive defense, and Bayesian-repeated game theory may guide us to find the "optimal" proactive defense strategies.

—In *region* 7, Bayesian-repeated (signaling) games can be used. First, repeated games can be used since the correlation degree among attack actions is low. As a result, we can assume that in each stage game both the attacker's and system's action spaces are the same. Second, although the detection accuracy is very good, 100% accuracy is usually not achievable, so Bayesian games are still beneficial. Third, since intrusion detection is both accurate and agile, the system can gain much better knowledge about the attacker in this region than in region 1. And such knowledge should be fully exploited to do better defense than in region 1 where simple Bayesian-repeated games are played. In particular, we believe that in each stage a *signaling game* can be played, where the system observes-then-act and exploits its knowledge about attack actions to make its (type) belief about the observed action more precise. Fourth, in this region effective detection favors reactive defense mechanisms, and doing proactive defense may not be cost effective, since substantial denial-of-service can be caused.

—In *region* 3, normal multistage dynamic games should be used, and subgame perfect Nash equilibrium strategies should be taken by the players. Specifically, since the detection latency is long, reactive defense can be very limited. When defense actions are not taken until an intrusion is detected, the effects of the intrusion can seriously spread throughout the system (via both attack and good actions) during the detection latency. Hence, proactive defense can be more beneficial. To support proactive defense, a simple multistage dynamic game can be used, where each stage is associated with (a) a good or bad action, but not both; and (b) a defense action which could be "null." Note that these two actions can be simultaneous or the system can observe-then-act. Since the detection accuracy is poor, in each stage the system has uncertainty about the other action's type. Such uncertainty can be handled by Bayesian type belief and expected payoffs. And in many cases, such uncertainty can be substantially reduced by the alerts raised and the alert correlation results, especially when the detection accuracy is not so bad (e.g., in region 6).

Compared with the combination of probabilistic "attack states" and stochastic game models, simple multistage dynamic games are easier, cheaper, having a smaller search space, more accurate, and having no need to know all the attack states. Finally, note that Bayesian-repeated games cannot be directly applied here because the attack actions are highly correlated.

So in each stage, the action spaces for both the attacker and the system are typically different and a different game is played.

—Finally, the "gray" areas (i.e., *regionsh* 2,4,5,6, *and* 8 usually need a trade-off between the extreme cases we discussed above when we need to build a good game-theoretic AIOS model for such a region. For example, a good AIOS game model for *region* 4 should be a trade-off between Bayesian-repeated (signaling) games (which are used in *region* 7) and Bayesian-repeated games (which are used in *region* 1). These trade-offs 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 in Section 6.

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 two-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 as $T^{sub} = \{good, bad\}$. 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 includes a pair of simultaneous actions (a^{sys}, a^{sub}). And each play will have an outcome denoted by $o(a^{sys}, a^{sub})$. (d) The system is uncertain about the type of the subject. This uncertainty is measured by the system's *type belief*, denoted as p_{sys}^{type}. For example, $p_{sys}^{type}(bad)$, a probability, denotes the system's belief about the statement that the subject is an attacker. (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 legitimate user, 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.[6] In particular, for each Nash equilibrium of the game, denoted as $(a_{sys}^*, a_{bad}^*, a_{good}^*)$, the game model will output a_{bad}^* as the attack strategy inferences (i.e., a_{bad}^* indicates the kind of strategies that are more likely to be taken by the attacker); output $u_{sub}(o; bad)$ (i.e., the utility function) and $u_{sub}(a_{sys}^*, a_{bad}^*; bad)$ as the attacker intent and objectives inferences, where $u_{sub}(a_{sys}^*, a_{bad}^*; bad)$ can be mapped to the amount of security vector degradation

[6]The Nash equilibrium theory can be found in Appendix and Mesterton-Gibbons [1992]. 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.

caused by the attack. Moreover, as side benefits, a^*_{sys} indicates a better defense posture and $u_{sys}(a^*_{sys}, a^*_{bad})$ indicate 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.

6. CASE STUDY: INFERRING THE ATTACK STRATEGIES OF DDOS ATTACKERS

In this case study, we want to infer the strategies of the attackers that enforce brute-force DDoS attacks. Regarding the network topology, the attack model, the system model, and the defense mechanism, we make exactly the same assumptions as in Example 1. In particular, we assume pushback is deployed by the system. Based on the aggregates and the corresponding traffic volume, pushback classifies both the traffic and the users into three categories: *good*, *poor*, and *bad*. The bad traffic is sent by a bad user (attacker) and is responsible for the congestion. The poor and good traffics are legitimate traffics, and are sent by the poor and good users (both legitimate), respectively. However, the poor traffic has the same aggregate properties as the bad traffic, but the good traffic has not, though the good traffic may share some paths with the bad traffic. To illustrate, in Figure 1. we assume the attacker compromises S_0 and sends "bad" packets to a victim denoted by d_0. Simultaneously, S_{31} sends legitimate packets to d_0. If router $R1.0$ uses destination address to identify the congestion aggregate, the poor packets sent from S_{31} to d_0 may be viewed as "bad" packets since they have the same destination address as the bad traffic, and dropped by the defense system. In summary, if the aggregates are destination-address based in Figure 1, then all packets sent to the same destination will belong to the same aggregate. Accordingly, when the attacker floods DoS packets to a set of victims, all the legitimate packets sent to the victims are poor traffic and would be rate limited together with the bad traffic. Nevertheless, the legitimate packets sent to other hosts are good traffic, such as the traffic between hosts in $\{S_0, \ldots, S_{64}\}$.

6.1 The Game-Theoretic AIOS Model

Now, we are ready to present the specific Bayesian game-theoretic AIOS model for DDoS attack/defense, which is specified as follows. Without losing generality, we assume that in each DDoS attack, there is one attacker and multiple legitimate users. (Nevertheless, it should be noticed that our AIOS model can be easily extended to handle collusive attackers.) For concision, we only mention the differences from the generic Bayesian game model proposed in Section 5.2.

$$\begin{aligned}\text{DDoSGM} = \{&A_{att}, A^1_{leg}, \ldots, A^i_{leg}, A_{sys}, T_{att}, T^1_{leg}, \ldots, T^i_{leg}, T_{sys},\\ &p_{att}, p^1_{leg}, \ldots, p^i_{leg}, p_{sys}, u_{att}, u^1_{leg}, \ldots, u^i_{leg}, u_{sys}\},\end{aligned}$$

where

(1) The players are the attacker, the system, and several legitimate users. It should be noticed that we cannot model this game as a two-player game and we must extend the two-player Bayesian game model proposed in Section 5.2, since zombies and legitimate hosts are sending packets to the victim(s) simultaneously and neither zombies nor legitimate hosts can "control" the actions taken by the other side. Also note that our game model can be easily extended to model collusive DDoS attacks among multiple attackers.
(2) The attacker's action space is $A_{att} = \{A_1, \ldots, A_m\}$, where A_i is a DDoS attack launched by the attacker. No matter which kind of DDoS attacks A_i belongs to, there are typically some common properties among the attacking packets involved in A_i. For example, they may have the same destination address, or they may use the same protocol. In this model, we use $\{Sour, V, AttTraf, Config\}$ to specify a DDoS attack. In particular, *Sour* is the set of zombies "selected" by this attack. $V = \{v_1, \ldots, v_l\}$ is the set of victims. Note that the victims may stay on different subnets. *AttTraf* specifies the attacking traffic patterns, for example, the protocol and transmission rate patterns between *Sour* and *V*. *Config* specifies the adaptable or reconfigurable aspects of the zombies. For example, the zombies may adjust their sending rate or traffic patterns according to the response of the defense system. In this study, we assume $Config = Null$, that is, the zombies will not adjust their behaviors during this attack and all of them will have simply the same behavior.
(3) The action space for legitimate user k is $A^k_{leg} = \{T_1, \ldots, T_m; 1 \leq k \leq i\}$, where T_i is a specific network application (or service). In the model, we use $\{Sour, Dest, Traffic, Config\}$ to specify such an application. Each network application may involve multiple hosts that transmit packets simultaneously in a coordinated or interactive way to execute a business process. Within a network application, *Sour* is the set of source hosts (or initiators) involved; and $Dest = \{d_1, \ldots, d_k\}$ is the set of destinations (or responders) involved. Moreover, *Traffic* captures the traffic patterns between *Sour* and *Dest*. *Config* specifies the adaptable or reconfigurable aspects of the application. In this study, we assume $Config = Null$.
(4) The system's action space A_{sys} is determined by the pushback postures of each router in the system. The system is composed of every router that is part of the pushback defense, denoted as $\{R_1, \ldots, R_n\}$. In particular, the pushback behavior of a router is determined by the following configurable parameters:
—*Congestion checking time*, denoted as p^1_{sys} (default value: 2 s). The router checks if the network is congested every p^1_{sys} seconds. 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* are fixed for "reporting"

serious congestion and for determining who should receive pushback messages. Note also that how the rate limits (for each aggregate) are set up is also fixed.

—*Cycle time*, denoted as p^2_{sys} (default value: 5 s) 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.

—*Target drop rate*, p^3_{sys} (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 toward the output queue less than $B/(1 - target\_drop\_rate)$, where B is the bandwidth of the output link.

—*Free time*, denoted as p^4_{sys} (default value: 20 s), is the earliest time to release a rate-limited aggregate after it goes below the limit imposed on it.

—*Rate-limit time*, denoted as p^5_{sys} (default value: 30 s), determines how long a newly identified aggregate must be rate limited. After the period, the router may release an aggregate.

—*Maximum number of sessions*, p^6_{sys} (default value: 3), determines the maximum number of aggregates the rate limiter can control.

—*Aggregate pattern*, denoted as p^7_{sys} (default value: "destination address prefix") determines which kinds of properties will be used to identify aggregates.

(5) The attacker's type space is $T_{att} = \{bad, good\}$. Legitimate user i's type space is also $T^i_{leg} = \{bad, good\}$. The system's type space is $T_{sys} = sys$.

(6) Regarding the system's type belief, since when a packet arrives at a router, the router cannot tell whether the sender of the packet is a zombie or not, the system's belief (or uncertainty) about every other player's type is the same, that is, $p^{good}_{sys} = \theta$, and $p^{bad}_{sys} = 1 - \theta$. In our simulation, for simplicity, we assume there are one attacker and one legitimate user. Accordingly, $\theta = 0.5$. In real world, the value of θ can be estimated based on some specific statistics of the DDoS attacks that have happened toward the system.

(7) Regarding the attacker and legitimate users' type belief, since both the attacker and the legitimate users know the system's type, $p^{type}_{att}(sys) = p^{type}_{leg}(sys) = 1$. Since the attacker knows who are zombies and who are legitimate nodes, the attacker's uncertainty about a legitimate user's type is $p^{type}_{bad}(good) = 1$. However, a legitimate user typically has uncertainty about the type of a node that is not involved in his application, since he is not sure whether the node is a zombie or not. So a legitimate user's uncertainty about the attacker's type and another legitimate user's type are the same, namely $p^{type}_{leg}(bad) = \beta$ and $p^{type}_{leg}(good) = 1 - \beta$.

(8) For each outcome o of a game play, the attacker's utility is $u_{att}(o) = \alpha u^{sys}_{att}(o) + (1-\alpha)\sum^i_{k=1} u^{leg_k}_{att}(o)$, where $u^{sys}_{att}(o)$ measures the attack's impact on the network, while $u^{leg_k}_{att}(o)$ measures the attack's impact on legitimate user k. In particular, $u_{att}(o) = \alpha B_{ao}/B_N + (1-\alpha)(1 - B_{lo}/B_{lw})$, where B_{ao}

is the bandwidth occupied by the attacker; B_N is the bandwidth capacity; B_{lo} is the bandwidth occupied by the legitimate user (note that we assume there is only one legitimate user); B_{lw} is the bandwidth that the legitimate user wants to occupy. For simplicity, B_{ao}, B_N, B^i_{lo}, and B^i_{lw} are all measured based on the incoming link to the edge router of the victim(s), as shown in Figure 1. Note that B_{ao}/B_N indicates the absolute impact of the attack on the (whole) network, while $1 - B^k_{lo}/B^k_{lw}$ indicates the relative availability impact of the attack on legitimate user k. α is the weight that balances these two aspects. Usually the attacker is mainly concerned with the attack's impact on legitimate users, so in this study we let $\alpha = 0.2$.

(9) The legitimate user's utility is $u_{leg}(o) = u^{sys}_{leg}(o) + p^{type}_{leg}(bad)u^{bad}_{leg}(o)$. Since the system controls both the legitimate and the bad traffic, and the attacker does not control the legitimate traffic directly, we simply let $u^{bad}_{leg}(o) = 0$. Therefore, $u_{leg}(o) = B_{lo}/B_{lw}$.

(10) The system's utility function is $u_{sys}(o) = w\theta B_{lo}/B_{lw} + (1 - w)(1 - \theta)(-B_{ao}/B_N)$; and it is defined in the standard way. w is the weight that helps the system make the trade-off between throttling the attacker and providing more bandwidth to legitimate users, and we set it as 0.8 in the simulations.

Although in this case study several specific parameter values are set, the above DDoS attack strategy inference model is a general model and can handle a variety of other DDoS attack scenarios beyond the case study. For example, our model can handle the scenario where the zombies adjust their strategies (e.g., attacking rate, traffic pattern) according to the response of the defense system. Moreover, although in our model the system's action space is pushback specific, our model can be extended to support other DDoS defense mechanisms such as traceback.

6.2 Simulation

In order to obtain concrete attack strategy inferences of real-world DDoS attackers, we have done extensive simulations on the game plays specified above using ns-2 [ns2]. The network topology of our experiments is shown in Figure 1, which is the same as the topology used in pushback evaluation [Ioannidis and Bellovin 2002]. There are 64 source hosts and four 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 11 defense strategies, as shown in Table I. The default value combination of $\{p^1_{sys}, \ldots, p^7_{sys}\}$ is the 7th defense strategy, which is the default defense strategy. In the experiment, we only change one parameter each time and compare the results with those under the default 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 10 s.

Table I. The Eleven System Strategies

System Strategy	p^1_{sys} (s)	p^2_{sys} (s)	p^3_{sys}	p^4_{sys} (s)	p^5_{sys} (s)	p^6_{sys}	p^7_{sys}
1st	4	5	0.05	20	30	3	Destination Address Prefix
2nd	2	10	0.05	20	30	3	Destination Address Prefix
3rd	2	5	0.03	20	30	3	Destination Address Prefix
4th	2	5	0.07	20	30	3	Destination Address Prefix
5th	2	5	0.05	10	30	3	Destination Address Prefix
6th	2	5	0.05	30	30	3	Destination Address Prefix
7th	2	5	0.05	20	30	3	Destination Address Prefix
8th	2	5	0.05	20	15	3	Destination Address Prefix
9th	2	5	0.05	20	50	3	Destination Address Prefix
10th	2	5	0.05	20	30	5	Destination Address Prefix
11th	2	5	0.05	20	30	3	Destination Address Prefix plus traffic pattern

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 10 s. The 6th is different in that the free time is 30 s. The 8th is different in that the rate limit time is 15 s. The 9th is different in that the rate limit time is 50 s. The 10th is different in that the maximum number of sessions is 5. The 11th is that the aggregate property is destination address prefix plus traffic pattern.

In our experiments, A_{leg} is materialized as follows:

—The poor traffic volume is 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 = 67.1 kbps (the average rate to a web site during the rush hour); RATE2 = 290 kbps (the average rate from an Intranet to the Internet); RATE3 = 532 kbps (the average rate from an Intranet to the Internet during the rush hour). Based on these statistics, we let the total poor traffic volume be 67.1 kbps, 290 kbps, or 532 kbps.

—The traffic pattern of the good and poor traffic is CBR (constant bits rate).

—There is only one legitimate user in the system. In each DDoS experiment, the legitimate user selects 2 (FEWPOOR) or 4 (MANYPOOR) hosts to send packets to the victim. When the poor traffic volume is 290 kbps and there are four poor hosts, each host will send out 290/4 bps traffic to the victims. Since the traffic pattern for poor traffic is fixed, the poor traffic belongs to a single aggregate.

—Moreover, in each DDoS experiment, the legitimate user selects 5 or 10 hosts to sends packets to other destinations. We assume the good traffic flows 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.

—The poor and good hosts are randomly chosen from the 64 hosts.

—In summary, for each poor traffic volume, there are four legitimate strategies corresponding to different numbers of poor hosts and good hosts. So in total there are 12 legitimate strategies.

A_{att} is materialized as follows:

—We set the number of zombies as 12 (FewBad) or 32 (ManyBad). The zombies are randomly chosen from the 64 hosts.

—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 290 kbps, 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 victim(s).

—When the poor traffic volume is 67.1 kbps or 290 kbps, we let the bad-to-poor ratio be 30, 35, 40, 45, or 50. When the poor volume is 532 kbps, we let the ratio be 30, 35, or 40. In this way, we totally get 13 possible attack traffic volumes.

—The traces also show four 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.

—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 there 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 (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.

6.3 Payoffs and Their Attack Strategy Implications

Figure 7 shows the attacker's, legitimate user' and defense system's payoffs under different network scenarios (i.e., poor traffic volumes), attacking strategies, and defense strategies when the aggregate property is destination address prefix. Figure 8 is different in that the aggregate property is destination address prefix plus traffic pattern. That is, two traffic flows sent to the same host or subnet may not always belong to the same aggregate because their traffic patterns may be different from each other. Note that for clarity, we show the legitimate strategies' effects in a special way. Since our results show that poor traffic volumes can have a significant impact on the players' payoffs while the numbers of poor or good hosts have almost no impact, we break down the 12 legitimate strategies into three groups based on the three different poor traffic volumes. And within each group, for each pair of attack and defense strategies, first the players' payoffs are measured based on the four legitimate strategies in that group, then an average payoff will be calculated for each player. Hence, each payoff shown in Figures 7 and 8 is an average payoff.

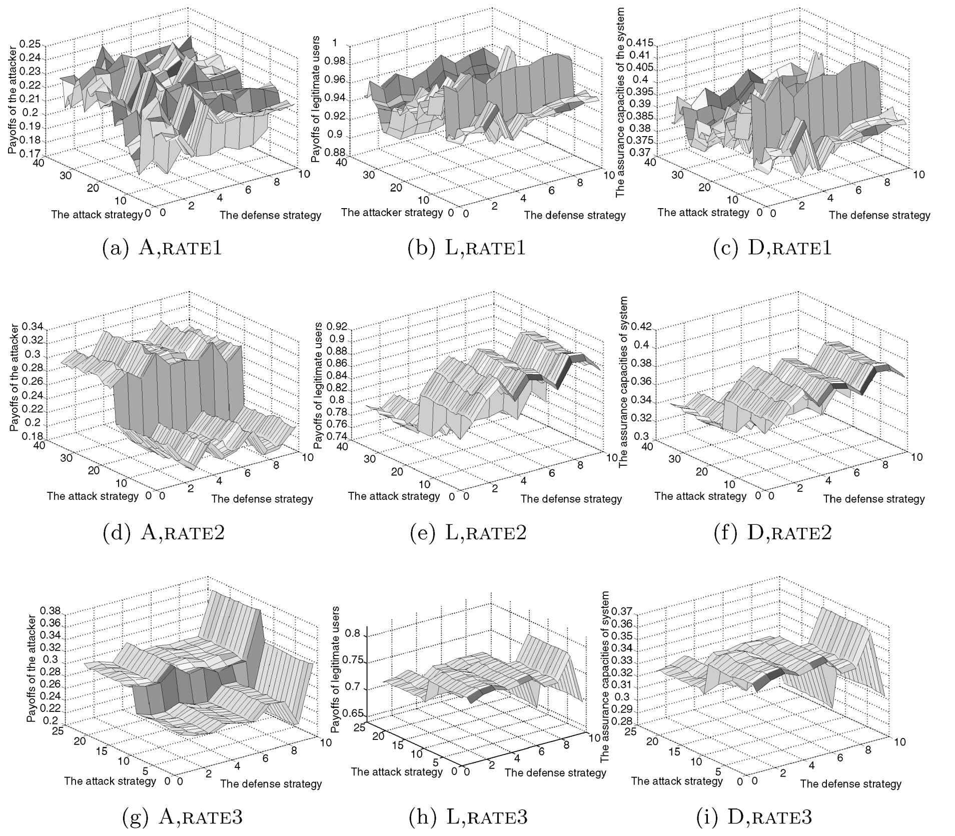
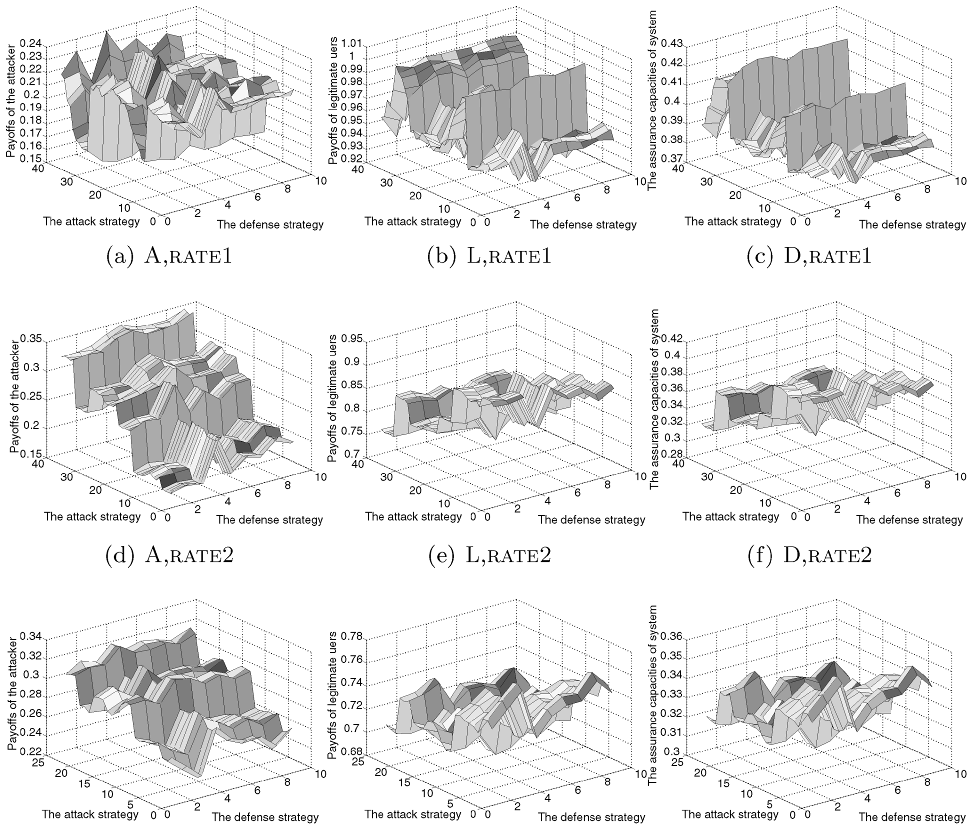


Fig. 7. The attacker's, legitimate user's and defense system's payoffs under different defense and attack strategies.

Based on the simulation results summarized in these two figures, where axis X represents the 10-defense strategies and axis Y represents the attacking strategies, we found that

(1) The attacker's payoffs are dependent upon not only attack strategies, but also network scenarios and defense postures, which well justifies the strategy interdependency property of our AIOS model.
(2) Our experiments confirm many well-known observations about DDoS attack and defense. For example, the attacker prefers more zombies and the defense system prefers lower drop rate. Nevertheless, our experiments give more insights on DDoS attack and defense. For example, many people believe that the attacker's and defense system's payoffs are mainly determined by the attack and defense strategies, but Figure 9 shows that the ratio between the poor traffic volume and the total bandwidth is a very important factor, and this ratio may greatly affect the attacker's and defense system's payoffs.
(3) Our experiments also yield several surprising observations. (a) Many people may believe that the more packets the zombies send out to the victims,



(a) A,RATE1 (b) L,RATE1 (c) D,RATE1

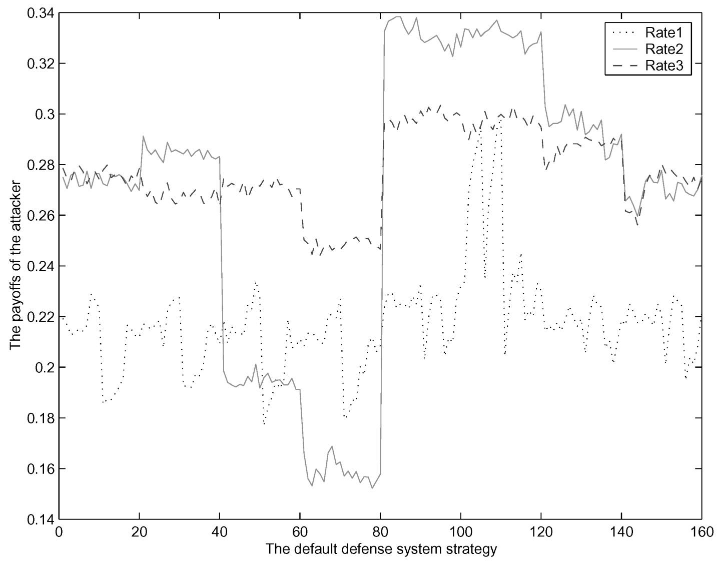
(d) A,RATE2 (e) L,RATE2 (f) D,RATE2

(g) A,RATE3 (h) L,RATE3 (i) D,RATE3

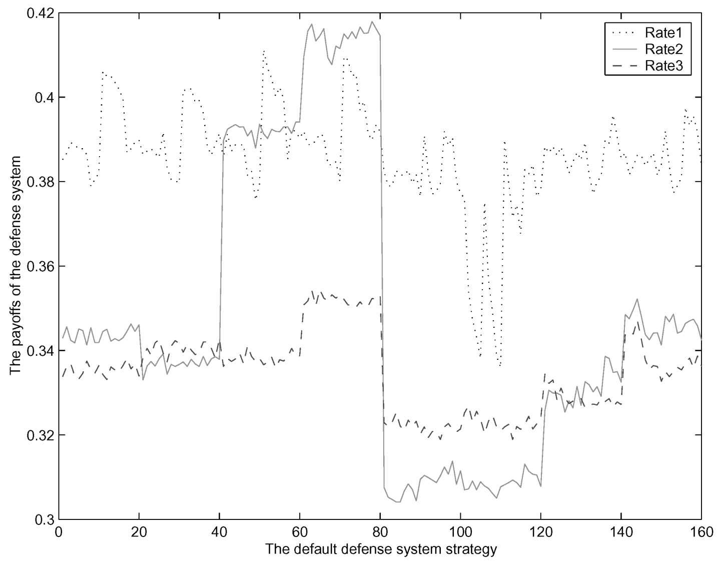
Fig. 8. The attacker's, legitimate user's and defense system's payoffs under different defense and attack strategies.

the more bandwidth and payoffs the attacker should earn. (b) Many people may believe that using different traffic patterns should be more effective in attacking than a single traffic pattern. (c) Many people may believe that exponential bit rate should be more effective in attacking than constant bit rate. (d) Many people may believe that using UDP should be more effective in attacking than TCP or ICMP. However, our results show that neither the attacking rate nor the traffic pattern matters, and different bad-to-poor ratios (30, 35, or 50) or different traffic patterns (UDP or ICMP) give the attacker similar amounts of payoffs. Actually, Figure 7 shows that among all the attacking strategies, only the number of zombies and the traffic aggregate properties can substantially affect the payoffs of the attacker.

(4) For the system, to obtain higher resilience against DDoS attacks, it needs only be concerned with three specific pushback parameters, namely target drop rate (p_{sys}^3), maximum number of sessions (p_{sys}^6), and aggregate pattern (p_{sys}^7). The other parameters do not affect the results much.



(a) The attacker's payoffs



(b) The defense system's payoffs

Fig. 9. The attacker's and defense system's payoffs under different poor traffic.

6.4 Nash Equilibrium Calculation

Figures 7 and 8 show the attacking capacity of the attacker, the survivability of the legitimate user, and the resilience of the defense system under different defense and attacking strategies. In the real world, the legitimate user, attacker, and defense system will only choose optimal strategies from their action spaces to maximize their payoffs. Hence, to know what the attacker, legitimate user, and defense system will do when a DDoS attack really happens, we need to know the Nash equilibrium strategies of the players. Nash equilibria specify the expected payoff maximizing best response of one player to every other player.

For each game play that involves a legitimate strategy, an attacking strategy, and a defense strategy, we can get three payoffs for the legitimate user, the attacker, and the system, respectively. We denote the payoffs by a 3-tuple $\langle L, A, D\rangle$. "L" is the legitimate user's payoff, "A" is the attacker's payoff, and "D" is the system's payoff. Note that each payoff 3-tuple is associated with a strategy 3-tuple which records the corresponding legitimate, attack, and defense strategies. Based on the experimental results of multiple game plays, we can get a set of payoff 3-tuples, which is called the *payoff list*. In accordance with the definition of Nash equilibria, we use the following steps to calculate the Nash equilibria:

(1) In the payoff list, for each legitimate strategy and attack strategy combination, we look for the defense strategies that give the highest payoff to the system. The resulting strategy 3-tuples forms strategy sublist 1.
(2) In the payoff list, for each legitimate strategy and defense strategy combination, we look for the attack strategies that give the highest payoff to the attacker. The resulting strategy 3-tuples forms strategy sublist 2.
(3) In the payoff list, for each defense strategy and attack strategy combination, we look for the legitimate strategies that give the highest payoff to the legitimate user. The resulting strategy 3-tuples forms strategy sublist 3.
(4) Every strategy 3-tuple in the intersection of sublist 1, sublist 2, and sublist 3 is a Nash Equilibrium.

It should be noticed that even in the same experimental environment, we may get different results in each experiment. Due to experiment errors, when we repeat an experiment, the payoffs of the three players may not be exactly the same as those produced by the original experiment. For example, in ns2, when a host is set up to send packets at rate 10 kbps, we cannot guarantee that the host will send exactly 10 kbps in every experiment. Some minimum errors usually exist, and the host may send 10.2 kbps or 9.8 kbps. Therefore, when we calculate the Nash equilibria, we set a relative measurement error. If the difference between two payoffs is less than the given measurement error, we view them equivalent to each other.

6.5 Nash Equilibria and Their Attack Strategy Implications

We get 42 Nash equilibria in the experiments when the relative measurement error is 0.005.[7] And some of them are shown in Table II.

We found that several interesting and fresh attack strategy inferences can be obtained from the distributions of the set of equilibria. In particular,

(1) In terms of the traffic pattern, the distribution is shown in Table III. "Dest" means the aggregate property is destination address prefix. "DestPatt"

[7]When we reduce the relative measurement error, we get fewer Nash equilibria but the distributions of the Nash equilibria are almost the same.

Table II. Nash Equilibrium Strategies

System Strategy	Legitimate Strategy	Attacking Strategy
Target-drop-rate = 0.03	RATE1, ManygoodFewpoor	Few, 35, CBR, One aggregate
Target-drop-rate = 0.03	RATE1, ManygoodManypoor	Many, 30, CBR, One aggregates
Target-drop-rate = 0.03	RATE1, ManygoodFewpoor	Many, 30, CBR, One aggregate
Maximum session = 5	RATE2, ManygoodManypoor	Many, 35, CBR, Multiple aggregates
Maximum session = 5	RATE2, ManygoodManypoor	Many, 30, EXP, Multiple aggregates
Maximum session = 5	RATE2, ManygoodManypoor	Many, 35, EXP, Multiple aggregates
Maximum session = 5	RATE2, ManygoodManypoor	Many, 45, EXP, Multiple aggregates

Table III. Nash Equilibrium Distribution under Different Attacking Patterns

Aggregate Property	CBR	EXP	ICMP	Mixed
DEST	0.09	0.50	0.27	0.14
DESTPATT	0.38	0.25	0	0.38

means that the aggregate property is destination address prefix plus traffic pattern. Table III shows that the attacker is more likely to use EXP traffic. In this way, he has more chances to stay at a Nash equilibrium since 50% Nash equilibria occur when the traffic pattern is EXP. When the aggregate is DESTPATT, obviously, the attacker prefers to use the same traffic patterns as those used by poor and good users.

(2) The distribution under bad-to-poor ratio is shown in Table IV. Surprisingly, the distribution shows that the attacker is most unlikely to use a high ratio. Some people may believe that higher bad-to-poor ratio should make the attack more successful since more packets will be flooded to the victim(s). However, our analysis of Nash equilibria distributions shows that the attacker has better opportunities to converge to a Nash equilibrium strategy with low bad-to-poor ratio. We believe that an important reason for this phenomenon is because our DDoS game is not a zero-sum game.

(3) The distribution under different combinations of the number of zombies, poor hosts and good hosts is shown in Table V. In the table, “F” means “Few” and “M” means “Many”. “FMF” means “FewgoodManypoorFewbad”. The table indicates that the attacker prefers to use as many zombies as possible, which is consistent with the common sense of DDoS attacks.

(4) The distribution under different defense strategies indicates that most Nash equilibria occur when the target-drop-rate is 0.03 or when the max-number-of-sessions is 5. The probability that Nash equilibria occur under target-drop-rate 0.03 is 0.45, and under max-number-of-session 5 is 0.36. Hence, to be more resilient, the system can increase the number of sessions and decrease the target-drop-rate. Our analysis also shows that the impact of other defense strategy parameters on this distribution is minimum.

(5) Based on the set of Nash equilibria calculated and Figure 7, we can get the upper bounds of the attacking capacity of the attacker and the upper bounds of the assurance capacity of the defense system under different network scenarios. These upper bounds are shown in Table VI. In particular,

Table IV. Nash Equilibrium Distribution under Different Attacking Ratios

Aggregate Property	30	35	40	45	50
DEST	0.23	0.32	0.14	0.09	0.23
DESTPATT	0.50	0	0.13	0.25	0.13

Table V. Nash Equilibrium Distribution under Different Number of Users

Aggregate Property	FFF	FFM	FMF	FMM	MFF	MFM	MMF	MMM
DEST	0	0	0	0	0	0.09	0.55	0.36
DESTPATT	0.12	0	0.13	0.13	0	0	0.38	0.25

Table VI. Upper Bounds of the Assurance Capacity and Attacking Capacity

Network Scenario	Attacking Capacity	Assurance Capacity
RATE1	0.2076	0.3941
RATE2	0.2668	0.3820
RATE3	0.2862	0.3320

the upper bounds of the assurance capacity tell us how well the system (i.e., pushback) is resilient to DDoS attacks. The upper bounds of the attacking capacity tell us how serious the damage could be in the worst case. According to the definition of payoff function, the highest attacking capacity is 1 and the highest defense capacity is θ, which is 0.5 in the paper. We use different normal traffic to the victim to represent different network scenarios in the paper. When the traffic rate is very low, such as RATE1, no matter how hard the attacker tries, the highest attacking capacity he could get is only 0.2076. And the highest assurance capacity of the defense system is 0.3941. When the traffic rate is high, such as RATE3, the highest attacking capacity is 0.2862.

6.6 Converging to Nash Equilibria

If the attacker, legitimate user, and defense system are rational, they should take a Nash equilibrium (optimal) strategy. Even if they did not choose a Nash equilibrium strategy at the very beginning, incentives may automatically "guide" them to converge their ad hoc strategies to a Nash equilibrium strategy at a later stage. In this section, we give a simple example in Figure 10 to explain how a Nash equilibrium can be dynamically converged.

We assume that the legitimate user, attacker, and system start from state 1. The strategies and payoffs are listed in the sequence of legitimate user, attacker, and defense system in box 1, where ⟨MF⟩ means MANYPOOR-FEWGOOD, ⟨F,CBR,35⟩ means FEWBAD, CBR traffic and *ratio* = 35. Moreover, we assume that each player may change his strategy, while strategy changes are not simultaneously done, and the outcome of each strategy change can be observed by the other players before another strategy change is performed.

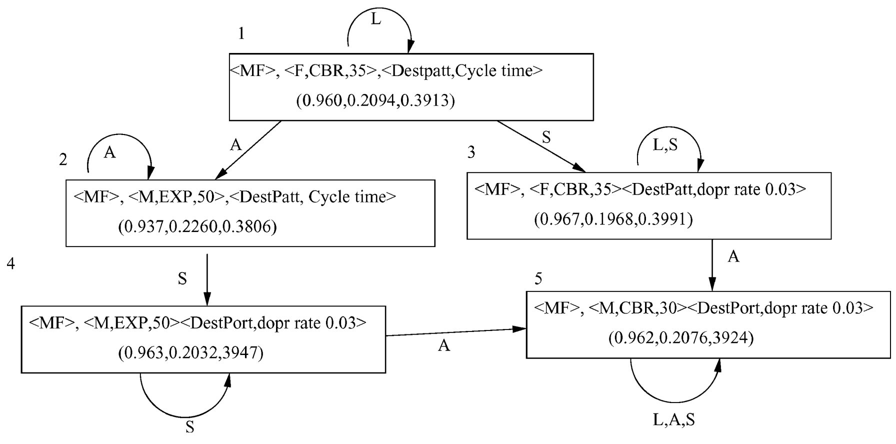

1
L
<MF>, <F,CBR,35>,<Destpatt,Cycle time>
(0.960,0.2094,0.3913)
2
A
A
<MF>, <M,EXP,50>,<DestPatt, Cycle time>
(0.937,0.2260,0.3806)
3
S
L,S
<MF>, <F,CBR,35><DestPatt,dopr rate 0.03>
(0.967,0.1968,0.3991)
4
S
A
5
<MF>, <M,EXP,50><DestPort,dopr rate 0.03>
(0.963,0.2032,3947)
A
<MF>, <M,CBR,30><DestPort,dopr rate 0.03>
(0.962,0.2076,3924)
S
L,A,S

Fig. 10. Converging to Nash equilibrium strategies.

In state 1, based on who is going to perform the next strategy change, the state may transit to different states. In particular, (a) if it is the attacker's turn, since he is not satisfied with his attack effects and since he finds that if he changes his strategy to ⟨M,EXP,50⟩, he can maximize his attack effects, he may take this strategy and transform the state to state 2. (b) If it is the system's turn, since the system is not satisfied with its resilience either and changing its strategy to ⟨DestPatt, target-drop-rate 0.03⟩ can maximize its resilience, the system may take this strategy and transform the state from state 1 to state 3. (c) Finally, if it is the legitimate user's turn, he will probably stay in state 1 without changing his strategy since his payoffs will decrease if he changes his strategy unilaterally.

In state 2, the system finds it can maximize its resilience if it changes its strategy to ⟨target-drop-rate 0.03⟩, so the system may take this strategy and transform the state to state 4. Similarly, in states 3 and 4, the attacker wants to change his strategy to ⟨M,CBR,30⟩ to maximize his attack effects, and the state finally transits to 5.

In state 5, everyone finds that if he changes his strategy unilaterally, his payoffs will decrease. Therefore, no one wants to change his strategy. Not surprisingly, this strategy 3-tuple in state 5 is a Nash equilibrium, as shown in Table II. The example shows that no matter what the start state is, the three players can ultimately "agree" on a Nash equilibrium to maximize their own incentives. Note that if there are more than one Nash equilibrium points, the convergence state would be dependent on the starting state.

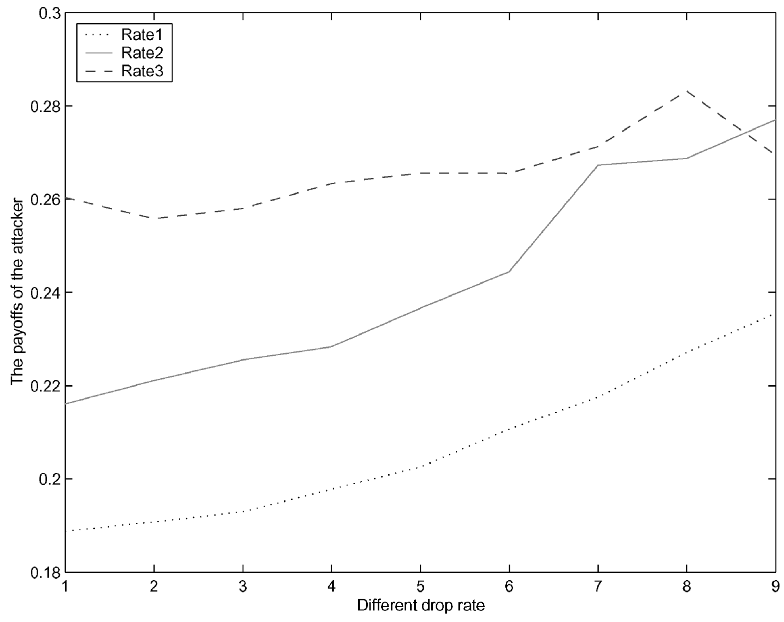
6.7 Using Attack Strategy Inferences to Improve the Network's Resilience

One side benefit of AIOS modeling and inferring is that the distribution of Nash equilibria and the payoff results could be used to optimize the system's defense posture for more resilience. In particular, we found that to better defend against DDoS attacks, the following issues need to be concerned by the network in its pushback defense:

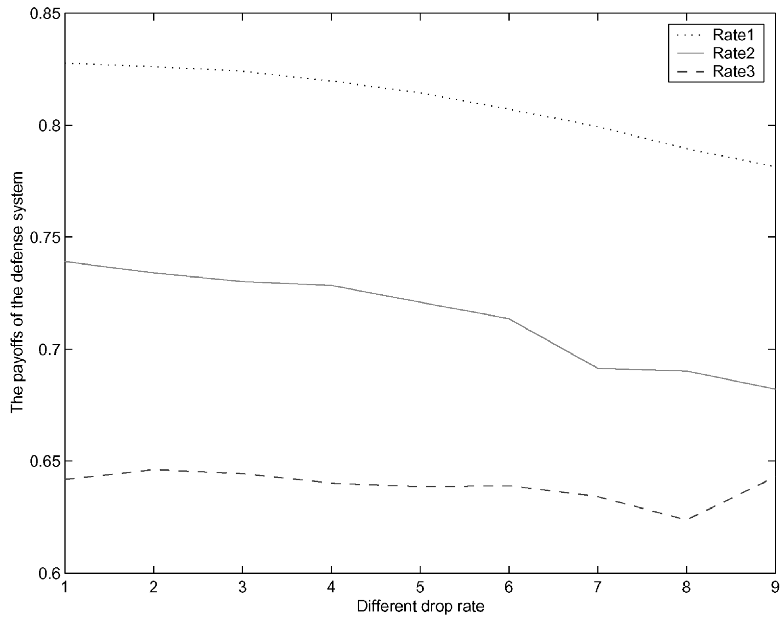
(a) *Normal Bandwidth Usage Planning.* We measure the degree of normal bandwidth usage by the ratio between the bandwidth occupied by the legitimate traffic and the total network bandwidth. Our results show that the higher the usage degree is, the lower the network's resilience will be. When the usage degree is high, more legitimate packets would be considered as malicious packets. When the usage degree is 0.05, less than 5% legitimate packets will be dropped no matter how the attacker changes his strategies. However, when the usage degree is 0.25, about 30% legitimate packets will be dropped. Hence, the degree of normal bandwidth usage should be carefully planed for good resilience. In practice, based on the profile of the legitimate traffics and their availability requirements, the system should be able to figure out the suitable degree of normal bandwidth usage.

(b) *Target Drop Rate Selection.* The lower the target drop rate is, the less packets will be sent to the output queue when a router is doing pushback, and more packets will be dropped by the defense system. From Figure 7, we found a lower drop rate is better than a higher drop rate. But it is hard to say if a lower drop rate is always better than a higher drop rate, since lower drop rates may cause more legitimate packets to be dropped by the system, especially when the legitimate traffic volume is high. To find the best drop rate, we give the simulation results of the assurance capacity of the system and the attacking capacity under different drop rate $\{0, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08\}$ in Figure 11. We found when the percentage of poor traffic in the whole bandwidth is low, such as 15%, a lower drop rate is always better for the system even when the legitimate traffic rate is high. When the percentage of poor traffic goes up to 25%, the system gets the best payoffs at the highest target drop rate. Hence, the target drop rate is dependent upon the percentage of poor traffic. When the percentage of poor traffic is low, the system should use a low drop rate, and vice versa. In practice, to find out the suitable target drop rate, the system needs to analyze the legitimate traffic when there are no attacks to get a profile of the legitimate traffic.

(c) *Configuration of the Number of Rate-Limited Sessions.* When the number of rate-limited sessions is less than the number of the attacking aggregates, there will be some malicious traffic not rate limited and the system will be jeopardized by these attack traffic. Hence, we need to make the number of rate limited sessions larger than the number of aggregates of the attacking traffic. Some people may believe that having too many rate-limited sessions is not good, since the legitimate traffic may be considered as malicious traffic and get rate limited. However, since the volume of malicious traffic is much larger than the normal traffic, our experiment results show that a larger number of rate-limited sessions do not affect the system's resilience seriously. In the real world, it is usually hard to predict accurately the number of attacking aggregates, therefore, we suggest the system just set a large number of rate-limited session for better resilience.



(a) The attacker's payoffs



(b) The defense system's payoffs

Fig. 11. The attacker's and defense system's payoffs under different drop rates.

Table VII. The Distribution of Nash Equilibria under Different Topologies

Topology	cont4	cyct10	dr0.03	dr0.07	ft10	ft30	default	sess5	rt15	rt50
Pushback	0	0	0.45	0	0	0.14	0	0.36	0	0.05
Britetopo	0.02	0	0.38	0	0	0	0.08	0.39	0.06	0.06

6.8 Experiments with a Larger Network Topology

So far, our simulations are based on the original pushback topology composed of 64 hosts and 22 routers. To see whether the characteristics of the simulated DDoS attack/defense game (e.g., characteristics of the payoffs and Nash equilibria) and the corresponding conclusions we have drawn can still hold in a large scale DDoS attack/defense game, we have done some experiments with a larger network topology. In particular, we use *Brite* [Medina et al. 2001], a popular topology generator, to create a network with 101 routers and more than 1000 hosts. We randomly select 200 hosts as zombies. To compare the experiment results with those generated with the pushback topology, we let the attacking bit rate (of each zombie) be the same as before, and we also let the legitimate bit rate be the same as before.

We found that with the Brite topology, the legitimate user's and system's payoffs are slightly smaller than those with the pushback topology, but the attacker's payoffs are slightly larger than those with the pushback topology. We believe the reason is mainly due to the fact that the pushback mechanism works in slightly different ways under different topologies. Nevertheless, the absolute values of payoffs are not very important for this comparison, and we are mainly concerned with the impact of the game parameters on the players' payoffs and the distributions of the Nash Equilibria.

Through a comparison study, we found that compared with the DDoS attack/defense game with the pushback topology, the impact of the game parameters on the players' payoffs is of almost the same set of properties, and the distributions of the Nash equilibria, as shown in Table VII, are very similar. For example, with the Brite topology, (a) the legitimate user always gets the highest payoffs when the target-drop-rate is 0.03 and gets the lowest payoffs when the target-drop-rate is 0.07; and (b) most Nash equilibria occur when the target-drop-rate is 0.03 or when the max-number-of-sessions is 5. These encouraging results, though still preliminary, show that the set of attack strategy characteristics (inferences) we have identified (computed) about DDoS attackers should hold in a large network and can be fairly consistent with the IOS of real-world DDoS attackers against the Internet.

7. CONCLUSION AND FUTURE WORK

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 interdependency between

AIOS and defender objectives and strategies in such a way that AIOS can be automatically inferred. Moreover, we developed a taxonomy of game-theoretic AIOS inference models. Finally, we use a specific case study on DDoS attack and defense to show how attack strategies can be inferred in real-world attack–defense scenarios.

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, in our future work (a) we would investigate model level inference accuracy analysis and sensitivity analysis that can model and predict the influence of incomplete information, asymmetric information (between the attacker and system), and uncertainty; (b) we would investigate approximate algorithms that can do optimal, quantitative trade-offs between inference precision and efficiency during Nash equilibria estimation; (c) we would investigate AIOS inference models beyond Bayesian games, that is, the other type of AIOS inference models identified by our taxonomy.

APPENDIX: A SIMPLE REVIEW OF GAME THEORY

The *normal-form representation* of an n-player game specifies the players' *strategy* spaces $S_1, \dots, S_n$ and their *payoff* functions $u_1, \dots, u_n$. We denote this game by $G = \{S_1, \dots, S_n; u_1, \dots, u_n\}$. In this game, the strategies $(s_1^*, \dots, s_n^*)$ 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, $(s_1^*, \dots, s_{i-1}^*, s_{i+1}^*, \dots, s_n^*)$. That is, s_i^* solves $\max_{s_i \in S_i} u_i(s_1^*, \dots, s_{i-1}^*, s_i, s_{i+1}^*, \dots, s_n^*)$.

A *pure strategy* for player i is an element of set S_i. Suppose $S_i = \{s_{i1}, \dots, s_{ik}\}$, then a *mixed strategy* for player i is a probability distribution $p_i = (p_{i1}, \dots, p_{ik})$, where $o \leq p_{ik} \leq 1$ for $k = 1, \dots, K$ and $p_{i1} + \cdots + p_{ik} = 1$. Although a game does not always have a pure strategy Nash equilibrium, Nash [1950] 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 *multistage 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 by $G(T-t)$. 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.

Besides repeated games, there are several types of more general *dynamic games*. For example, in *multistage dynamic games*, a different game can be played in each stage instead of playing the same game repeatedly. In dynamic observe-then-act games, players can take actions in turn instead of taking actions simultaneously. Nevertheless, the definitions of strategies, subgames, and subgame-perfect Nash equilibriums are very similar to those in repeated games.

Finally, a standard formal definition of stochastic games is as follows. An n-player stochastic game Γ is a tuple $\langle\, S, A^1, \ldots, A^n, r^1, \ldots, r^n, p \,\rangle$, where S is the state space, A^i is the action space of player i for $k = 1, \ldots, n$, $r^i : S \times A^1 \times \cdots \times A^n \to R$ is the payoff function for player i, $p : S \times A^1 \times \cdots \times A^n \to \nabla$ is the transition probability map, where ∇ is the set of probability distributions over state space S [Thusijsman 1992]. In Γ, a *strategy* $\pi = (\pi_0, \ldots, \pi_t, \ldots)$ 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 $\pi_t : \mathbf{H}_t \to \sigma(A^i)$, where $\mathbf{H}_t$ is the space of possible histories at time t, with each $H_t \in \mathbf{H}_t$, $H_t = (s_0, a_0^1, \ldots, a_0^n, \ldots, s_{t-1}, a_{t-1}^1, \ldots, a_{t-1}^n, s_t)$, 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 $(\pi_*^1, \ldots, \pi_*^n)$ such that for all $s \in S$ and $i = 1, \ldots, n$, $v^i(s, \pi_*^1, \ldots, \pi_*^n) \geq v^i(s, \pi_*^1, \ldots, \pi_*^{i-1}, \pi^i, \pi_*^{i+1}, \ldots, \pi_*^n)$ for all $\pi^i \in Pi^i$, where Π_i is the set of strategies available to agent i.

REFERENCES

BROWNE, H., ARBAUGH, W. A., MCHUGH, J., AND FITHEN, W. L. 2001. A trend analysis of exploitations. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*. 214–229.

BROWNE, R. 2000. C4i defensive infrastructure for survivability against multi-mode attacks. In *Proceedings of 21st Century Military Communication-Architectures and Technologies for Information Superiority*.

BURKE, D. 1999. *Towards a Game Theory Model of Information Warfare*. Tech. rep., Air force Institute of Technology. Master's Thesis.

CLARKE, E. H. 1971. Multipart pricing of public goods. *Public Choice 11*, 17–33.

CONITZER, V. AND SANDHOLM, T. 2002. *Complexity Results About Nash Equilibria*. Tech. rep., Carnegie Mellon University. CMU-CS-02-135.

CUPPENS, F. AND MIEGE, A. 2002. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*.

DEBAR, H. AND WESPI, A. 2001. Aggregation and correlation of intrusion detection alerts. In *Proceedings of the 2001 International Symposium on Recent Advances in Intrusion Detection*. 85–103.

FEIGENBAUM, J., PAPADIMITRIOU, C., SAMI, R., AND SHENKER, S. 2002. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing*.

GORDON, L. A. AND LOEB, M. P. 2001. Using information security as a response to competitor analysis systems. *Commun. ACM 9*, 44.

GROVES, T. 1973. Incentives in teams. *Econometrica 41*, 617–663.

HESPANHA, J. P. AND BOHACEK, S. 2001. Preliminary results in routing games. In *Proceedings of the 2001 American Control Conference*.

IOANNIDIS, J. AND BELLOVIN, S. M. 2002. Implementing pushback: Router-based defense against ddos attacks. In *Proceedings of the 2002 Annual Network and Distributed System Security Symposium*.

KOLLER, D. AND MILCH, B. 2001. Multi-agent influence diagrams for representing and solving games. In *Proceedings of the 2001 International Joint Conference on Artificial Intelligence*.

LANDWEHR, C. E., BULL, A. R., MCDERMOTT, J. P., AND CHOI, W. S. 1994. A taxonomy of computer program security flaws. *ACM Comput. Surv. 26*, 3.

LIU, P., JAJODIA, S., AND MCCOLLUM, C. D. 2000. Intrusion confinement by isolation in information systems. *J. Comput. Security 8*, 4, 243–279.

LUNT, T. F. 1993. A survey of intrusion detection techniques. *Computers & Security 4*, 12 (June), 405–418.

LYE, K. AND WING, J. M. 2002. Game strategies in network security. In *Proceedings of the 2002 IEEE Computer Security Foundations Workshop*.

MALKHI, D. AND REITER, M. K. 2000. Secure execution of java applets using a remote playground. *IEEE Trans. Software Eng. 26*, 12.

MAS-COLELL, A., WHINSTON, M. D., AND GREEN, J. R. 1995. *Microeconomic Theory*. Oxford University Press, Oxford, UK.

MCHUGH, J. 2001. Intrusion and intrusion detection. *Int. J. Inf. Security 1*, 14–35.

MEDINA, A., LAKHINA, A., MATTA, I., AND BYERS, J. 2001. An approach to universal topology generation. In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*.

MESTERTON-GIBBONS, M. 1992. *An Introduction to Game-Theoretic Modeling*. Addison-Wesley Publishing, Reading, MA.

MUKHERJEE, B., HEBERLEIN, L. T., AND LEVITT, K. N. 1994. Network intrusion detection. *IEEE Network*, 26–41.

NASH, J. 1950. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences*. 48–49.

NING, P., CUI, Y., AND REEVES, D. S. 2002. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 2002 ACM Conference on Computer and Communications Security*.

NISAN, N. AND RONAN, A. 2001. Algorithmic mechanism design. *Games and Economic Behavior 35*.

NS2. The network simulator. http://www.isi.edu/nsnam/ns/.

SYVERSON, P. F. 1997. A different look at secure distributed computation. In *Proceedings of the 1997 IEEE Computer Security Foundations Workshop*.

THUSIJSMAN, F. 1992. *Optimality and Equilibria in Stochastic Games*. Centrum voor Wiskunde en Informatica, Amsterdam.

VICKREY, W. 1961. Counterspeculation, auctions, and competitive sealed tenders. *J. Finance 16*, 8–37.

WANG, X. AND REITER, M. 2003. Defending against denial-of-service attacks with puzzle auctions. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*.

WELLMAN, M. P. AND WALSH, W. E. 2001. Auction protocols for decentralized scheduling. *Games and Economic Behavior 35*.

XU, J. AND LEE, W. 2003. Sustaining availability of web services under distributed denial of service attacks. *IEEE Trans. Comput. 52*, 4 (Feb.), 195–208.

ZOU, C., GONG, W., AND TOWSLEY, D. 2002. Code red worm propagation modeling and analysis. In *Proceedings of the 2002 ACM Conference on Computer and Communications Security*.

Received May 2004; revised September 2004; accepted September 2004