On the Limits of Payload-Oblivious Network Attack Detection

M. Patrick Collins[1] and Michael K. Reiter[2]

[1] RedJack
`michael.collins@redjack.com`*
[2] Department of Computer Science,
University of North Carolina at Chapel Hill
`reiter@cs.unc.edu`

Abstract. We introduce a methodology for evaluating network intrusion detection systems using an *observable attack space*, which is a parameterized representation of a type of attack that can be observed in a particular type of log data. Using the observable attack space for log data that does not include payload (*e.g.*, NetFlow data), we evaluate the effectiveness of five proposed detectors for bot harvesting and scanning attacks, in terms of their ability (even when used in conjunction) to deter the attacker from reaching his goals. We demonstrate the ranges of attack parameter values that would avoid detection, or rather that would require an inordinately high number of false alarms in order to detect them consistently.

Keywords: network intrusion detection, ROC curve, evaluation.

1 Introduction

We address the problem of evaluating network intrusion detection systems, specifically against scan and harvesting attacks. In the context of this work, a harvesting attack is a mass exploitation where an attacker initiates communications with multiple hosts in order to control and reconfigure them. This type of automated exploitation is commonly associated with worms, however, modern bot software often includes automated buffer-overflow and password exploitation attacks against local networks[1]. In contrast, in a scanning attack, the attacker's communication with multiple hosts is an attempt to determine what services they are running; *i.e.*, the intent is reconnaissance.

While harvesting attacks and scanning may represent different forms of attacker *intent* (*i.e.*, reconnaissance vs. host takeover), they can appear to be similar phenomena in traffic logs. More specifically, a single host, whether scanning

* This work was done while the author was affiliated with the CERT/NetSA group at the Software Engineering Institute, Carnegie Mellon University.

[1] A representative example of this class of bot is the Gaobot family, which uses a variety of propagation methods including network shares, buffer overflows and password lists. A full description is available at `http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_AGOBOT.GEN`.

or harvesting, will open communications to an unexpectedly large number of addresses within a limited timeframe. This behavior led to Northcutt's observation that in the absence of payload—either due to the form of log data, encryption or simply a high connection failure rate—methods for detecting these attacks tend to be threshold-based [19]. That is, they raise alarms after identifying some phenomenon that exceeds a threshold for normal behavior.

Historically, such IDS have been evaluated purely as alarms. Lippmann *et al.* [16] established the standard for IDS evaluation in their 1998 work on comparing IDS data. To compare intrusion detectors, they used ROC curves to compare false positive and false negative rates among detectors. Since then, the state of the practice for IDS evaluation and comparison has been to compare IDS' ROC curves [9].

The use of ROC curves for IDS evaluation has been criticized on several grounds. For our purposes, the most relevant is the *base rate fallacy* described by Axelsson [2]. Axelsson observes that a low relative false positive rate can result in a high number of actual false positives when a test is frequently exercised. For NIDS, where the test frequency may be thousands or tens of thousands of per day, a false positive rate as low as 1% may still result in hundreds of alarms.

In this paper, we introduce an alternative method of evaluating IDS that focuses on an IDS' capacity to frustrate an attacker's goals. In order to do so, we develop a model for evaluating IDS that captures the attacker's payoff over an *observable attack space*. The observable attack space represents a set of attacks an attacker can conduct as observed by a particular logging system. The role of logging in the observable attack space is critical; for example, NetFlow, the logging system used in this paper, does not record payload. As such, for this paper, we define an observable attack space that classifies attacks by the attacker's *aggressiveness* (the number of addresses to which they communicate in a sample period) and their *success* (the probability that a communication opened to an address actually contacts something).

To evaluate the payoff, we construct a *detection surface*, which is the probability of detection over the observable attack space, and then apply a *payoff function* to this detection surface. The payoff function is a function representing the rate at which an attacker achieves the strategic goal of that attack, which is either occupying hosts (in a harvesting attack) or scouting network composition (in a scanning attack).

We use the payoff function to evaluate the impact of various IDS on attacker strategy. We can model payoff as a function of the number of viable hosts in a network that an attacking bot communicates with — the more hosts a bot can contact without being detected, the higher his payoff. We show in this paper that several methods which are good at raising alarms primarily identify low-payoff attacks; with these detectors, an attacker can achieve a high payoff simply by limiting his behavior.

By combining detection surfaces with a payoff function, we are able to compare IDS with greater insight about their relative strengths and weaknesses. In particular, we are able to focus on the relationship between detection capacity and attacker payoff. Instead of asking what kind of false positive rate we get for a specific true positive rate, we are able to relate false positive rates to the attacker goals. By doing so, we are able to determine how high a false positive rate we must tolerate in order to prevent an attacker from, say, substantially compromising a network via a harvesting attack. Our work therefore extends the ROC framework into a model of the attacker's own goals. By doing so, we can reframe questions of IDS designs by evaluating their impact on attacker behavior, on the grounds that a rational attacker will attempt to maximize payoff.

Using this approach, we compare the efficacy of five different detection techniques: client degree (*i.e.*, number of addresses contacted); protocol graph size and protocol graph largest component size [6]; server address entropy [15]; and Threshold Random Walk [11]. We train these systems using traffic traces from a large (larger than /8) network. Using this data, we demonstrate the configurations of aggressiveness and success rate with which an attack will go undetected by any of these techniques. Furthermore, we show that when configured to be sufficiently sensitive to counter attackers' goals, these anomaly detection systems will result in more than ten false alarms per hour, even when alarms are limited to occur only once per 30-second interval.

To summarize, the contributions of this paper are the following. First, we introduce a new methodology for evaluating NIDS that do not utilize payload. Second, we apply this methodology to evaluate several attack detection methods previously proposed in the literature, using data from a very large network. And third, we demonstrate via this evaluation the limits that these techniques face in their ability to prevent attackers from reaching harvesting or scanning goals.

The remainder of this paper is structured as follows. §2 is a review of relevant work in IDS evaluation and anomaly detection. §3 describes the IDS that we evaluate in this paper, and how we configure them for analysis. §4 describes the observable attack space and detection surface. §5 describes the first of our two attack scenarios, in this case the acquisition of hosts by an attacker with a hit list. §6 describes the second scenario: reconnaissance by attackers scanning networks. §7 concludes this work.

2 Previous Work

Researchers have conducted comparative IDS evaluations in both the host-based and network-based domains. In the host-based domain, Tan and Maxion [25,17] developed an evaluation methodology for comparing the effectiveness of multiple host-based IDS. Of particular importance in their methodology is the role of the data that an IDS can actually analyze, an idea further extended in Killourhy *et al.*'s work on a defense-centric taxonomy [13]. The methods of Tan and Maxion and of Killourhy *et al.* informed our experimental methodology and the concept of an observable attack space. However, their approach is focused on host-based IDS and they consequently work with a richer dataset then we believe feasible for NIDS.

A general approach to evaluating IDS was proposed by Cárdenas *et al.* [4], who developed a general cost-based model for evaluating IDS based on the work of Gaffney and Ulvila [8] and Stolfo *et al.* [24]. However, these approaches all model cost from a defender-centric viewpoint — the defensive mechanism is assumed to have no impact on the attacker. In contrast, our models treat the attacker as economically rational, meaning that the attacker attempts to maximize payoff within the rules given by the model.

The general problem of NIDS evaluation was first systematically studied by Lippmann *et al.* [16]. Lippmann's comparison first used ROC curves to measure the comparative effectiveness of IDS. The ROC-based approach has been critiqued on multiple grounds [18,9,2]. Our evaluation model is derived from these critiques, specifically Axelsson's [2] observations on the base-rate fallacy. Our work uses a ROC-based approach (specifically, comparing Type I and Type II errors) as a starting point to convert the relative error rates into payoffs.

3 IDS Construction and Training

In the context of this work, an *IDS* is an anomaly detection system that compares the current state of a network against a model of that network's state developed from historical data. In this section, we describe our candidate IDS, and our method for training and configuring them. This section is divided as follows: §3.1 describes the raw data, §3.2 describes the types of IDS used, and §3.3 describes the detection thresholds used for our IDS.

3.1 Raw Data

Every IDS in this paper is trained using a common data source over a common period of time. The source data used in this paper consists of unsampled NetFlow records[2] generated by internal routers in a large (in excess of 16 million distinct IP address) network. For training and evaluation, we use SSH traffic.

NetFlow records approximate TCP sessions by grouping packets into *flows*, sequences of identically addressed packets that occur within a timeout of each other [5]. NetFlow records contain size and timing information, but no payload. For the purposes of this paper, we treat NetFlow records as tuples of the form (clntip, srvip, succ, stime).

The elements of this tuple are derived from the fields available in CISCO NetFlow. The clntip, srvip, succ and stime fields refer, respectively, to the client address, server address, whether a session was successful, and the start time for the session. Since SSH is TCP based, we rely on the port numbers recorded in the original flow record both for protocol identification and classifying the role a particular address played in the flow. Any flow which is sent to or from TCP port 22 is labeled an SSH flow, srvip is the address corresponding to that port

[2] CISCO Systems, "CISCO IOS NetFlow Datasheet", `http://www.cisco.com/en/US/products/ps6601/products_data_sheet0900aecd80173f71.html`, last fetched October 8th, 2007.

and clntip the other address[3]. stime, the start time, is derived directly from the corresponding value in the flow record, and is the time at which the recording router observed the flow's earliest packet.

The succ element is a binary-valued descriptor of whether the recorded flow describes a legitimate TCP session. succ is 0 when the flow describes a TCP communication that was not an actual session (*e.g.*, the target communicated with a nonexistent host), 1 when the flow describes a real exchange between a client and a server.

succ is an inferred property in the sense that it can be truly determined only by the receiving host — a sufficiently perverse attacker could generate one side of a session without the others' involvement. *In situ*, we can approximate succ using other flow properties, such as the number of packets in the flow or TCP flag combinations. In our work on IDS training [7], we approximate succ by setting it to 1 when a flow has 4 or more packets, on the grounds that a TCP session has at least 3 packets of overhead. Other methods for calculating succ include looking for indicators such as total payload, the presence of ACK flags, or aggregate measures such as Binkley and Singh's TCP work weight [3].

In our simulations we generate the succ values as part of the process of generating attack flows. During the simulations, attackers choose their targets from a hit list generated from the training data; the attack's success rate determines how many addresses come from this hit list, and how many addresses are chosen from a pool of dark addresses. For flows communicating with the hit list, succ $= 1$, and for flows communicating with the pool of dark addresses, succ $= 0$.

IDS properties are generated using 30 second (s) samples of traffic data. We refer to a distinct sample as a *log file*, Λ, consisting of all the flows $\lambda_1 \ldots \lambda_l$ whose stime values occur in the same 30s period. The use of 30s periods comes from our previous work on protocol graphs [6].

3.2 IDS State Variables

In the context of this paper, an IDS is a threshold-based alarm that triggers if a value derived from a log file Λ exceeds a threshold derived from a set of training data. Each IDS in this paper is based around a single *state variable* which, when evaluated against a log file produces a scalar *state value.* For this paper, we evaluate the state of a log file using five distinct state variables: g, c, h, d and r. Each state variable is used by one IDS; we will refer to each IDS by its state variable (*e.g.*, "g is an IDS").

$g(\Lambda)$ and $c(\Lambda)$ are, respectively, the total graph size and the largest component size of a *protocol graph* constructed from Λ. A protocol graph, described in our previous work on hit-list detection, is an undirected graph constructed from a log of traffic for a single protocol over a limited observation period [6]. In a protocol graph, the nodes represent hosts communicating using that protocol, and the links represent that a communication between these two hosts happened during

[3] We constrain the flows used in this paper to flows which used an ephemeral port between 1024 and 5000.

that time. In a protocol graph, the graph size is equivalent to the total number of hosts communicating using a particular protocol. The largest component size is the size of the largest connected component of the graph.

$h(\Lambda)$ is the entropy of server addresses in Λ. This metric is derived from work by Lakhina *et al.* [15] on mining traffic features. The entropy is defined as:

$$h(\Lambda) = - \sum_{i \in \mathsf{srvs}(\Lambda)} \left(\frac{|\{\lambda \in \Lambda | \lambda.\mathsf{srvip} = i\}|}{|\Lambda|} \right) \log_2 \left(\frac{|\{\lambda \in \Lambda | \lambda.\mathsf{srvip} = i\}|}{|\Lambda|} \right) \quad (1)$$

where $\mathsf{srvs}(\Lambda) = \bigcup_{\lambda \in \Lambda} \lambda.\mathsf{srvip}$ is the set of all server addresses observed in the log file. During a harvesting attack, an attacker will increase $|\mathsf{srvs}(\Lambda)|$, which reduces the probability of any one server being the target of a communication and therefore increases the entropy.

$d(\Lambda)$, the maximum degree of Λ, is the number of servers with which the busiest client in Λ communicated. $d(\Lambda)$ is arguably the simplest form of scan detection and consequently has been used by a variety of anomaly detection systems, notably GrIDS [23] and Bro [20].

$r(\Lambda)$ is the maximum *failed connection run* observed in Λ. A failed connection run is a sequence of flow records $\lambda_1 \ldots \lambda_n$ where each λ in the run has the same client address and $\lambda_i.\mathsf{succ} = 0$. This method is used by TRW scan detection [11] to determine if an address is actively scanning. We use the maximum failed connection run measure to indicate whether TRW would have detected at least one attack during the sample period.

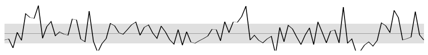
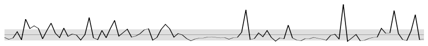
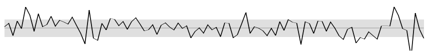
3.3 IDS Thresholds

In order to calculate detection thresholds for four of the IDS we consider (g, c, h and d), we first must train the IDS using log files of benign traffic from the monitored network. However, SSH traffic is prone to constant scanning [1] which, unless aggressively filtered, will result in artificially high thresholds.

To address the problem of constant clumsy scanning, we use a two-stage filtering method developed in previous work [7]. This approach is based on our previous observations that certain graph attributes of major protocols (graph size and largest component size) can be modeled using a Gaussian distribution when the traffic logs describing those attributes do not contain attacks [6]. Using these results, we use a stateless filter that eliminates records where $\mathsf{succ} = 0$. The resulting log files are then tested using the Shapiro-Wilk normality test [22] to identify those log files where the observed graph and largest component size are outside the expected range for a Gaussian distribution.

The initial training data consisted of 7,200 log files for the five business days between February 11–15, 2008. Source data was chosen exclusively from 1200GMT to 2359GMT for each day, a period corresponding to peak activity for the network. After filtering, the resulting set consisted of 5,619 log files from a source set of 7,200.

Table 1. Summary of Gaussian state variables in SSH training set

State variable x	Range	$\mu_X \pm \sigma_X$
g		299.27±42.49
c		35.13±21.32
h		6.88±0.35

Applying this filtering technique in order to isolate benign traffic yields a vector $\Lambda_1 \ldots \Lambda_m$ of log files, each representing benign traffic in a 30s interval. State values are calculated for each log file in this vector; we refer to the resulting vector of state values using the same subscript notation, *e.g.*, $r(\Lambda_i) = r_i$. We refer to the complete vector of values for a vector of log files by the corresponding capital letter (*e.g.*, $G = \{g(\Lambda_1) \ldots g(\Lambda_m)\}$).

We examined the H and D distributions in the filtered data to see if they could be modeled via a Gaussian distribution. (Our previous work already established that G and C are Gaussian for the monitored network [6].) Using the Shapiro-Wilk statistic (W) [22], we found that H had $W = 0.97$ and negligible p-value, and so we treated entropy as Gaussian. D had a Shapiro-Wilk statistic of $W = 0.77$ with negligible p-value, and consequently was not considered Gaussian.

Table 1 summarizes the Gaussian state variables, *i.e.*, g, c, and h. This table shows the summary data (left hand column), the mean and standard deviation (right side) and a sparkline for each data set. The sparkline is a time series plot of the activity over the training period. We plot the mean and shade an area one standard deviation from the mean in each sparkline.

For these three state variables, we can use (2) to calculate the detection threshold. For a given false positive rate, FPR, the corresponding threshold for a Gaussian IDS x is given by:

$$\theta_x = \mu_X + \sqrt{2}\text{erf}^{-1}(\mathsf{FPR})\sigma_X \tag{2}$$

where erf is the error function [14], μ_X is the arithmetic mean of the vector of observations X, and σ_X is the standard deviation of the same vector.

The detection threshold for d is computed differently since, as shown above, d is not normally distributed over the sample space. We use d's maximum observed value over the benign log files as the detection threshold:

$$\theta_d = \max(D) \tag{3}$$

The detection threshold for r is prescribed by Jung *et al.* to be

$$\theta_r = \frac{\beta \ln \frac{\beta}{\alpha} + (1-\beta)\ln\frac{1-\beta}{1-\alpha}}{t_1 \ln \frac{t_1}{t_0} + (1-t_1)\ln\frac{1-t_1}{1-t_0}} \tag{4}$$

Here, α and β are user-configured thresholds for the maximum false positive rate (α) and the minimum true positive rate (β). For this work, we set $\beta = 1 - \alpha$, and set α to our acceptable FPR (see below). t_0 and t_1 are, respectively, the probabilities that a normal user will successfully communicate with a target, and the probability that a randomly scanning attacker will successfully communicate with a target. Per these definitions, t_0 and t_1 depend on a variety of factors including the density of targets in the network, the type of protocol involved, and dynamic addressing, some of which are difficult to accurately evaluate for the monitored network due to our limited view of it. However, Jung's simulation analysis of TRW [10] suggest that choices of t_0 and t_1 have relatively little impact on performance. As such, we generally adopt Jung's original values of $t_0 = 0.8$ and $t_1 = 0.2$ and will examine the impact of alternative θ_r values in §5.2.

Recall that based on our previous work on graph-based anomaly detection [6], we monitor traffic over 30s periods. This 30s period governs the effective response time of the entire intrusion detection and response mechanism — an IDS sends out at most one alert in a period, and defenders respond to changes at that time. If we constrain the *aggregate* false positives for *all* of the detectors to one false alarm per eight hours (*i.e.*, the duration of a typical network analyst's shift), this yields a combined rate of 0.1% for the five IDS together. We solve for individual false positive rates FPR using

$$0.001 = 1 - (1 - \mathsf{FPR})^5 \tag{5}$$

Plugging this value of FPR into (2) yields detection thresholds $\theta_g = 447$, $\theta_c = 110$, and $\theta_h = 8.105$, and setting $\alpha = \mathsf{FPR}$ in (4) yields $\theta_r = 6$. We also use the value $\theta_d = 150$, computed directly from (3). These are the thresholds we use in our evaluations in subsequent sections. Equation 5 treats each IDS as a statistically independent. While not true, this simplifies our exploratory analysis.

4 Observable Attack Spaces and Detection Probability

In §3, we developed and configured a combined IDS based around five different state variables: graph size g, largest component size c, server address entropy h, maximum client degree d and maximum failed connection run r. In doing so, we specifically configured these systems to yield a low false positive rate, resulting in one false positive per eight-hours as predicted by our training data. Now that we have developed this hybrid IDS, we can evaluate its efficacy for deterring attackers.

In order to do this, we develop a method for describing attacker utility which we call the *observable attack space* (OAS). An observable attack space describes the range of attacks that an attacker can conduct *as observed by a particular logging mechanism*. In this section, we develop an observable attack space common to our logging system (NetFlow) and our five candidate IDS. Using this approach, we model the aggregate *detection surface* of the OAS and use this to evaluate both our combined IDS and the constituent IDS individually.

This section is structured as follows. §4.1 describes OAS, IDS and the estimation of detection surfaces. §4.2 then compares the effectiveness of our five detection methods both in aggregate and as individual detection schemes.

4.1 OAS and Detection Surface

The type of log data that an IDS uses strongly impacts the types of attacks that an IDS can detect. An example of this is the impact of choosing NetFlow. NetFlow is a compact representation of traffic that is viable to collect on large networks, but since it lacks payload, signature-matching techniques are not possible with this log format. An observable attack space is therefore a parameterized representation of all possible forms of a particular attack, as observable using a particular form of log data. For this work, the observable attack space has two attributes: *aggressiveness* (a) and *success* (s). The aggressiveness is a natural number describing the number of distinct addresses with which the attacker communicates in the observation period. The success of an attack is the fraction of these communications that were successful, and is within the range $[0, 1]$.

When conducting simulations, we limit a to the range of $(0, \theta_d)$ because we treat the d IDS as deterministic — it will trigger if *and only if* $a \geq \theta_d$. In doing so, we ignore the possibility that during an attack, a benign host contacts more than θ_d addresses, thus "accidentally" causing a true detection even though $a < \theta_d$. This treatment also presumes that the attack is launched from a bot that is not also contributing benign traffic at the same time, *i.e.*, $a < \theta_d$ implies that the bot host does, in fact, contact fewer than θ_d addresses in a 30s interval. The other IDS' chances of detecting attacks are not so directly dependent on an attack's characteristics within the OAS.

Consider a particular IDS $x \in \{g, c, h, r\}$. Given an arbitrary log file of control data Λ^{ctl} that we are confident does not contain an attack, $\mathcal{P}^{x}_{\mathsf{det}}(a, s)$ is the probability that the IDS x raises an alarm for the log file resulting from Λ^{ctl} merged with an attack Λ^{atk} with aggressiveness a and success s. That is,

$$\mathcal{P}^{x}_{\mathsf{det}}(a, s) = \mathbb{P}\left[x(\Lambda^{\mathsf{atk}} \cup \Lambda^{\mathsf{ctl}}) \geq \theta_x\right] \tag{6}$$

where the probability is taken with respect to the selection of Λ^{ctl} and the generation of Λ^{atk} with aggressiveness a and success rate s. For a particular IDS x, the *detection surface of* x is the surface of values $\mathcal{P}^{x}_{\mathsf{det}}(a, s)$ for $a \in (0, \theta_d)$ and $s \in [0, 1]$.

More specifically, to estimate the probability of detection and the corresponding detection surface, we evaluate the distribution of state variables for normal behavior merged with randomly generated attacks meeting the aggressiveness and success requirements specified by a and s. For this paper, we limit our simulations to $a \in \{10, 20, 30, 40, \ldots, 140\}$ (recall $\theta_d = 150$) and $s \in \{0.1, 0.2, 0.3, \ldots, 1.0\}$. At each point, we conduct 100 simulations, each using one of fifty randomly selected 30s periods from the week of February 18–22 (the week following that used for training) for Λ^{ctl}. Λ^{atk} is randomly generated for

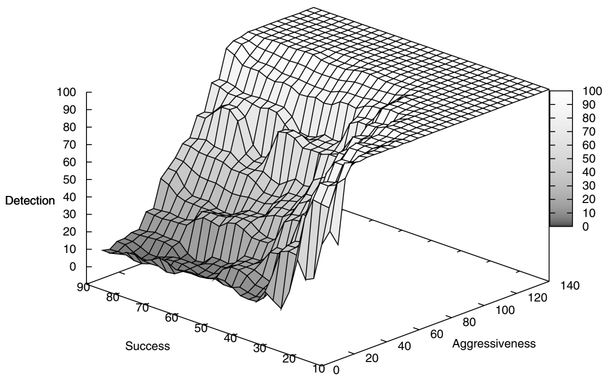


Fig. 1. Detection surface ($\mathcal{P}^{\mathsf{all}}_{\mathsf{det}}(a, s)$, as a percentage) for combined IDS

each simulation. Λ^{atk} contains a unique records, where each record has the same client address, and a different server address. The composition of the server addresses is a function of s: $a \cdot s$ addresses are chosen from a hit list of internal SSH servers identified in the training data[4] in order to approximate hit-list attacks; the remainder are sent to addresses with no listening server. We then merge Λ^{atk} with a randomly selected control log Λ^{ctl} and then calculate the state variables.

Four of the IDS examined by this paper (g, c, h, and d) are unaffected by the order of log records within the monitored 30s period. The fifth, r, is order-sensitive, however, in that TRW triggers an alert if any host is observed making more than θ_r failed connections *in a row*. This order sensitivity is a weakness, since an attacker can interleave scanning with connections to known hosts in order to avoid a failed connection run greater than θ_r [12]. To address this particular exploit, we randomly permute the records originating in each 30s interval. After this permutation, r is calculated for each host in the network.

Figure 1 plots the detection surface for all the IDS combined. As this figure shows, the combined detection mechanism generally increases proportionally to the aggressiveness of the attack and inversely relative to the success of the attack. Furthermore, the detection mechanisms tend to vary more as a function of the aggressiveness than due to the success rate.

The effectiveness of the aggregate IDS may be considered low, in the sense that an attacker with a relatively small hit list ($a = 40, s = 0.5$) can communicate with the network with little fear of detection. However, we should note that the attacks represented by this OAS are the most subtle space of attacks available. Our own experience indicates that the majority of attacks are orders of magnitude more aggressive than these hypothetical SSH scans, at which point *any* IDS will identify them. This latter point is particularly critical. As Figure 1 shows, once $a \geq 100$, the combined IDS will raise an alarm.

[4] This hit list is composed of all internal addresses in the training data which had one flow originating from them on port 22 and with a payload of greater than 1kB.

4.2 Detection Surface Comparison

In addition to the detection surface for the aggregate IDS, we have also calculated the detection surfaces for each component IDS in this system. We can use these results to evaluate the comparative effectiveness of each IDS.

Figure 2 plots detection surfaces for each IDS $x \in \{g, c, h, r\}$ as *contour plots.* A contour plot maps a 3-dimensional surface into a 2-dimensional representation using *contour lines.* Each contour line represents the axis coordinates where the surface takes on its labeled value.

These plots show that the most successful individual IDS are c and r : these IDS are the only ones to have significant ($\geq$ 10%) detection rates over the majority of the OAS. In contrast, the h IDS has a very *low* detection rate, less than 6% over the entire OAS. Of particular interest with c and r is their relative disconnectedness to each other: r's detection rate is dependent on s and less dependent on a. Conversely, c is largely independent of s, while a plays larger role in detection.

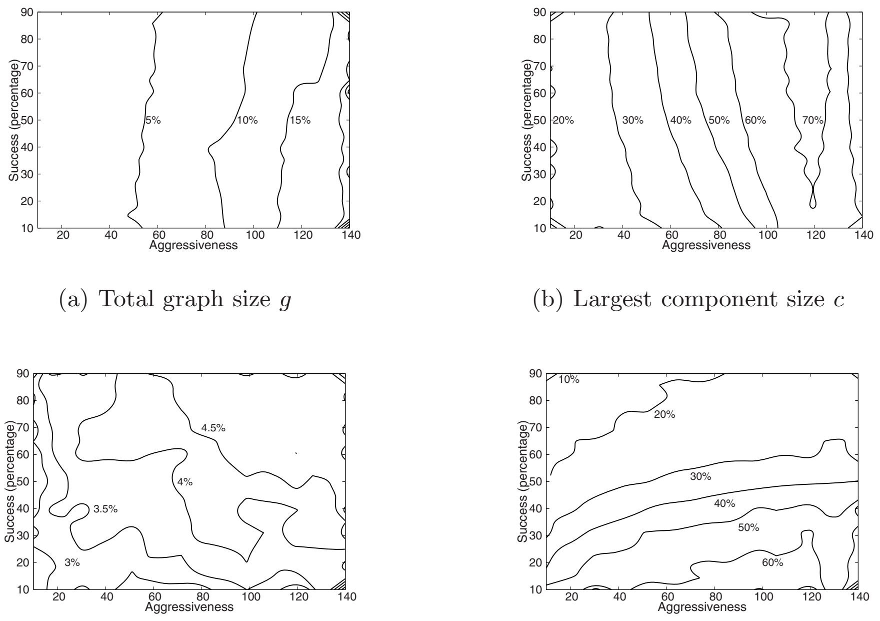
These IDS are calibrated to have an effective false positive rate of zero. As a result, they are largely insensitive to anomalies and have a relatively low detection rate. In addition, as noted above, the attacks represented here are extremely subtle. More aggressive attackers would be identified and eliminated *regardless* of the detection strategy used — by the time an attacker communicates with $\theta_d = 150$ addresses, the d IDS will raise an alarm, making other approaches effectively moot.

This phenomenon is partly observable in our models in Table 1. Recall that, for example, the model of graph size g, was 299 ± 42.47 hosts. If $g(\Lambda) = 299$ for some log file Λ, then an attacker will not trigger an anomaly until he has communicated with at least 149 hosts, at which point he is close to triggering d as well as g.

5 Modeling Acquisition

In §4.2 we examined the efficacy of the detection mechanisms purely as detectors: for a fixed false positive rate, we calculated the effective true positive rate. In this section, we use the detection surface in Figure 1 to examine the impact of IDS on *acquisition attacks.* We evaluate the efficacy of the detection surface by building a mathematical model for attacker payoff during an acquisition attack. Applying this model to the surface, we can determine how many hosts an attacker can expect to take over, and from these results determine how effective an IDS has to be in order to keep attackers from taking over hosts.

This section is divided as follows: §5.1 describes our model for acquisition attacks. §5.2 compares IDS efficiency using our payoff function. §5.3 considers the problem of IDS evaluation from a different perspective — instead of calculating efficiency in terms of true and false positives, we determine the minimum false positive rate required to counter an attacker.



(a) Total graph size g

(b) Largest component size c

(c) Server entropy h

(d) Maximum failed connection run r

Fig. 2. Detection surfaces ($\mathcal{P}^x_{\mathsf{det}}(a, s)$, as a percentage) for individual IDS

5.1 Acquisition Payoff Model

We define an acquisition attack as a game between two parties who are competing for ownership of a single network. The two parties in the game are the *attacker*, who attempts to take over hosts on the network, and the *defender*, who attempts to prevent takeover of hosts on the network. In this game, the attacker uses a single *bot* to perform a series of *attempts*, during each of which the bot communicates with multiple hosts within the network using a *hit list* acquired previous to the attack.

In each attempt, the attacker communicates with some number of addresses (specified by the attacker's a), each of which has s chance of succeeding. For the purposes of the simulations, a successful attack is one that communicates with a real host, and a failed attack is one that communicates with a nonexistent host. That is, we assume that if an attacker talks with a host, the attacker takes the host over. The *payoff* of an attempt, $\mathcal{H}_{\mathsf{acq}}$, is the expected number of hosts with which the attacker communicates during an attempt.

The goal of the defender is to minimize $\mathcal{H}_{\mathsf{acq}}$, and the goal of the attacker to maximize the same. To do so, the defender deploys an IDS x, and so the probability of detecting a particular attempt with aggressiveness a and success rate s is $\mathcal{P}^x_{\mathsf{det}}(a, s)$. We assume that once the defender successfully identifies an

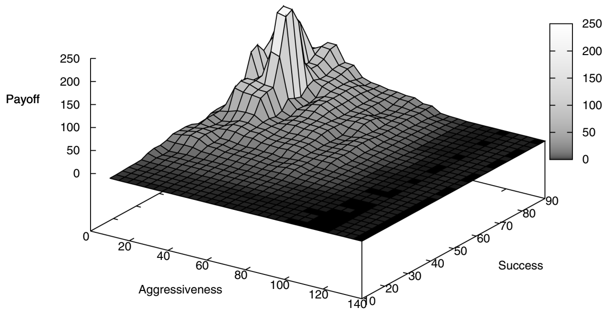


Fig. 3. Payoff $\mathcal{H}^{\mathsf{all}}_{\mathsf{acq}}(a, s, k^{\mathsf{all}}_{\max}(a, s))$ for acquisition attacks for combined IDS

attacker, it will block the attacker's bot, ending all further acquisition attempts by that bot. Furthermore, the defender will then recover all of the hosts that the bot communicated with during the game.

We note that this model assumes that the attacker and defender are perfect. That is, the probability that an attacker takes over a host that it contacts, and the probability that a defender correctly identifies an occupied host after being notified of an attack, are both one. The model can be modified by incorporating additional probabilities for measuring the attacker's takeover success per host contact and the defender's vigilance.

Let owned be a random variable indicating the number of hosts taken over, and let $\mathsf{alarmed}$ be the event that the bot is detected. Below, we assume that the probability of detection in each attempt is independent. If such is the case, then we can derive the payoff for an attack comprised of k attempts and for an IDS x as:

$$\begin{aligned}\mathcal{H}^{x}_{\mathsf{acq}}(a, s, k) &= \mathbb{E}\left[\mathsf{owned}\right]\\ &= \mathbb{P}\left[\mathsf{alarmed}\right]\mathbb{E}\left[\mathsf{owned} \mid \mathsf{alarmed}\right] + \mathbb{P}\left[\neg\mathsf{alarmed}\right]\mathbb{E}\left[\mathsf{owned} \mid \neg\mathsf{alarmed}\right]\\ &= (1 - \mathcal{P}^{x}_{\mathsf{det}}(a, s))^{k}(ask) \qquad (7)\end{aligned}$$

The last step follows by taking $\mathbb{E}\left[\mathsf{owned} \mid \mathsf{alarmed}\right] = 0$, since we presume that if the defender detects an attacker during an attempt, then the defender recovers *all* of the hosts the attacker has communicated with using that particular bot. Note that the attacker maximizes his payoff by maximizing k subject to $\mathcal{H}_{\mathsf{acq}}(a, s, k) - \mathcal{H}_{\mathsf{acq}}(a, s, k-1) > 0$ or, in other words,

$$k < \frac{1 - \mathcal{P}^{x}_{\mathsf{det}}(a, s)}{\mathcal{P}^{x}_{\mathsf{det}}(a, s)} \qquad (8)$$

We denote this value of k by $k^{x}_{\max}(a, s)$.

Figure 3 plots the payoff over the observed attack space using (7) with the maximum k satisfying (8). As this figure shows, aggressive attacks have a minimal payoff, a result that can be expected based on Figure 1. Above approximately $a \geq 80$, the attacker is consistently identified and stopped regardless of their success rate.

This behavior is the result of the interaction of two detectors: c and r. As s increases, the probability of the attacker combining previously separate components of the protocol graph increases, increasing the likelihood of detection by the c IDS. As the attacker's success rate decreases, he is more likely to generate a sufficiently long failed connection run to trigger the r detector. The other detectors will identify attackers, however their effectiveness is limited for attacks that are this subtle — an attacker who does disrupt g or h will typically already have disrupted d.

5.2 Calculating IDS Efficiency

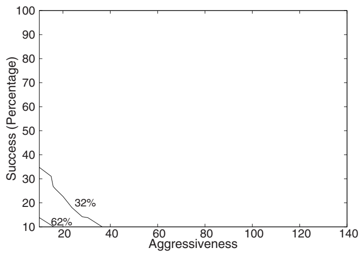
We can use (7) to also calculate a comparative efficiency metric for IDS. The volume under the surface specified by (7) is the ability of the attacker to take over hosts in the presence of a particular IDS. The *efficiency* of an IDS x is therefore the indicator of how much x reduces an attacker's payoff. We can express IDS efficiency as the ratio between the number of hosts an attacker expects to take over in the presence of an IDS x and the number of hosts the attacker can take over (in the same number of attempts) in that IDS' absence.

$$\mathcal{E}^x_{\mathsf{acq}} = 1 - \frac{\sum_{a \in (0,\theta_d)} \sum_{s \in (0,1]} \mathcal{H}^x_{\mathsf{acq}}(a, s, k^x_{\max}(a, s))}{\sum_{a \in (0,\theta_d)} \sum_{s \in (0,1]} ask^x_{\max}(a, s)} \tag{9}$$

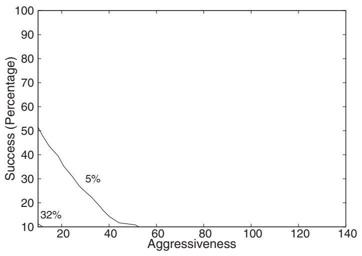
The subtraction in (9) is included simply to provide an intuitive progression for efficiency: if $\mathcal{E}$ is greater for IDS A than IDS B, then A is a better IDS than B. Based on (9), we can calculate an efficiency of 0.14 for g, 0.0099 for h, 0.73 for c and 0.22 for r. The effectiveness of the combined detector is 0.80.

Using Equation 9 we can examine the impact of alternative values for θ_r. Recall from §3.3 that θ_r is based on models of normal behavior and attacker behavior that can vary as a function of the protocol, the density of the observed network and other behaviors. Without revisiting the model, we can simply change the values of θ_r and examine how that changes the efficiency. In this case, we find that for $\theta_r = 3$, 4, and 5, $\mathcal{E}^x_{\mathsf{acq}} = 0.50$, 0.37, and 0.29, respectively. The most interesting result from these calculations is the relatively low efficiency of r as an IDS for acquisition attacks, despite its relatively good true positive rates (Figure 2). Because the detection mechanism relies on attacker failures, it is better at detecting attacks which have a relatively low s. IDS r is therefore very good at detecting attacks with low payoff.

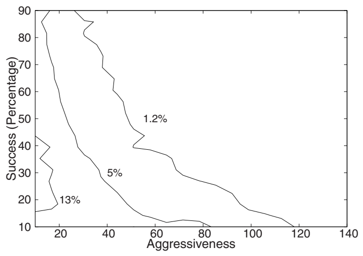
We expect that the comparative efficiency of these IDS will differ from one protocol to the next. g and h are affected by the aggregate traffic for one protocol, *e.g.*, the total number of hosts using a particular protocol. Conversely, r relies exclusively on per-host behavior. Consequently, using protocols with more clients or servers (such as HTTP) should result in less g and h efficiency, while r should have the same efficiency regardless of protocol.



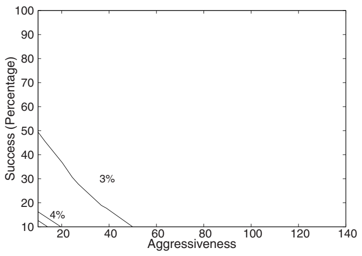
(a) Total graph size g



(b) Largest component size c



(c) Server entropy h



(d) Maximum failed connection run r

Fig. 4. False positive rates required to limit $\mathcal{H}^x_{\mathsf{acq}}(a, s, k) = 1$

5.3 Determining a Minimum False Positive Rate

As Figure 3 implies, even with all the detection mechanisms operating, attackers can still acquire a high rate of return with a sufficiently subtle hit-list attack. In this section, we will now address the question of detection from a different perspective: how high a false positive rate do we have to tolerate in order to prevent the attacker from seriously compromising the monitored network?

To do so, we invert (7) so that instead of calculating the attacker's payoff as a function of detectability, we calculate the probability of detection as a function of payoff. Solving for $\mathcal{P}^x_{\mathsf{det}}(a, s)$ in (7) yields

$$\mathcal{P}^x_{\mathsf{det}}(a, s) = 1 - \sqrt[k]{\frac{\mathcal{H}^x_{\mathsf{acq}}(a, s, k)}{ask}} \tag{10}$$

Suppose the defender wishes to minimize $\mathcal{P}^x_{\mathsf{det}}(a, s)$ (and hence also the false alarm rate) while restricting $\mathcal{H}^x_{\mathsf{acq}}(a, s, k) \leq 1$, and so the attacker wishes to maximize $\mathcal{P}^x_{\mathsf{det}}(a, s)$ in order to achieve $\mathcal{H}^x_{\mathsf{acq}}(a, s, k) = 1$. The attacker does so by choosing k so as to minimize $(ask)^{-1/k}$, for any a and s.

Using this strategy, we calculate the detection probability required to identify and stop attackers at points within the OAS. To calculate the resulting detection thresholds for each IDS, we use our simulated attacks with parameters a and s to calculate the threshold needed to filter off $\mathcal{P}^x_{\mathsf{det}}(a, s)$ of the attacks when overlaid on our training data.

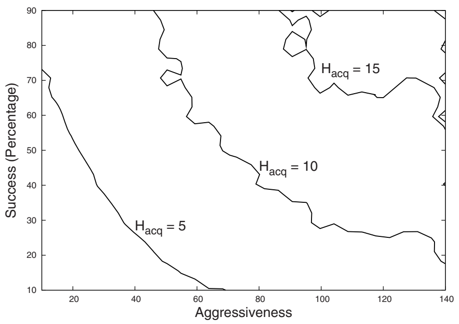


Fig. 5. Values of a and s for which $\mathcal{H}^c_{\mathsf{acq}}(a, s, k)$ can be limited to at most the specified value, using a threshold $\theta_c = \mu_C + 3.5\sigma_C$

The results of these runs are given in Figure 4. These figures are contour plots over the OAS as before. However, the contours for the figure are the false positive rates that would result from this analysis. For the g, c and h detectors, these values are calculated using (2). For r, this value is calculated by using (4).

As Figure 4 indicates, anomaly detection systems that are capable of defending against subtle attacks will require extremely high false positive rates. Recall that our measurement system conducts a test every 30s; for every 1% false positive rate we accept, we pay 10 alerts per eight-hour shift. As such, this figure indicates that the false positive rates for building systems that can limit the attacker to $\mathcal{H}^x_{\mathsf{acq}}(a, s, k) \leq 1$ are much higher than we can consider accepting.

One way to avoid such high false positive rates would be to not place such a stringent limit of $\mathcal{H}^x_{\mathsf{acq}}(a, s, k) \leq 1$. For example, if the defender insists on a near-zero false positive rate, we can determine if there is a higher threshold for the payoff that can accommodate this rate, such as $\mathcal{H}^x_{\mathsf{acq}}(a, s, k) \leq 5$. Figure 5 shows this for the c IDS, for $\mathcal{H}^c_{\mathsf{acq}}(a, s, k) \in \{5, 10, 15\}$. Specifically, each contour line shows the values of a and s for which $\mathcal{H}^c_{\mathsf{acq}}(a, s, k)$ can be limited to at most the specified value, using a threshold $\theta_c = \mu_C + 3.5\sigma_C$, which is large enough to ensure a false positive rate very close to zero. As this figure shows, the defender can effectively impose an upper limit on the attacker's payoff, but unfortunately this limit must be rather large ($\mathcal{H}^c_{\mathsf{acq}}(a, s, k) = 15$) in order to cover the majority of the attack space.

From Figures 4 and 5, we conclude that in order for an anomaly detection system to be a viable deterrent to host compromise, it must either use finer resolution data than NetFlow, develop mechanisms for coping with a high false positive rate, or permit higher attacker payoff than would be ideal.

6 Modeling Reconnaissance

In this section, we develop an alternative attack scenario, *reconnaissance*, where the attacker scouts out the network with his bots. In each attack, he communicates with addresses to simply determine the presence of hosts at certain addresses. The reconnaissance scenario differs from the acquisition scenario by the attacker's knowledge and goals. Specifically, the attacker's goal is to contact as many addresses as possible within a short period. To do so, the attacker uses a *chaff hit list* consisting of hosts that the attacker already knows about, and a target space of addresses to probe. The chaff hit list reduces the attacker's probability of detection by lowering his failure rate. However, it also reduces the attacker's payoff by requiring him to "sacrifice" a certain number of targets every round.

Let $\mathsf{alarmed} = i$ be the event that the bot is detected at the end of attempt i (and before attempt $i+1$); as before, an attempt is comprised of contacting a addresses with success rate of s (in this case, owing to the chaff hit list). Let $\mathsf{scanned}$ denote a random variable indicating the number of scans that one bot performs successfully (*i.e.*, determines whether the scanned address has a listening service or not), not counting the "chaff" that it introduces to avoid detection. Note that we suppose that the number of listening services the bot finds is sufficiently small that it does not relieve the bot from introducing a fraction s of chaff scans. We also presume that the probability the bot is detected in each attempt is independent.

$$\begin{aligned}
\mathcal{H}^x_{\mathsf{rec}}(a,s) &= \mathbb{E}\left[\mathsf{scanned}\right] \\
&= \sum_{i=1}^{\infty} \mathbb{P}\left[\mathsf{alarmed} = i\right] \mathbb{E}\left[\mathsf{scanned} \mid \mathsf{alarmed} = i\right] \\
&= \sum_{i=1}^{\infty} \left((1-\mathcal{P}^x_{\mathsf{det}}(a,s))^{i-1}\mathcal{P}^x_{\mathsf{det}}(a,s)\right)(ia(1-s)) \\
&= a(1-s)\frac{\mathcal{P}^x_{\mathsf{det}}(a,s)}{1-\mathcal{P}^x_{\mathsf{det}}(a,s)} \sum_{i=1}^{\infty} i(1-\mathcal{P}^x_{\mathsf{det}}(a,s))^i \\
&= a(1-s)\frac{\mathcal{P}^x_{\mathsf{det}}(a,s)}{1-\mathcal{P}^x_{\mathsf{det}}(a,s)} \frac{1-\mathcal{P}^x_{\mathsf{det}}(a,s)}{\mathcal{P}^x_{\mathsf{det}}(a,s)^2} \\
&= a(1-s)\frac{1}{\mathcal{P}^x_{\mathsf{det}}(a,s)} \qquad (11)
\end{aligned}$$

Applying (11) to the detection matrix over our OAS results in the payoff plot shown in Figure 6. This figure plots the aggregate payoff over the OAS for reconnaissance. Of particular note with this result is that it demonstrates that a sufficiently motivated and subtle attacker can scan a network by subtly exploiting attacks with high s rates. In this case, the attacker can slowly scan the network for an extended period — the observed peak at the $a = 20$ segment of the graph implies that the attacker scans for 25 minutes before being detected.

However, the attacker can achieve just as effective results by aggressively scanning the network. Recall that the effective aggressiveness of the attacker is bound by θ_d to less than 150 nodes. In the reconnaissance scenario, the attacker faces

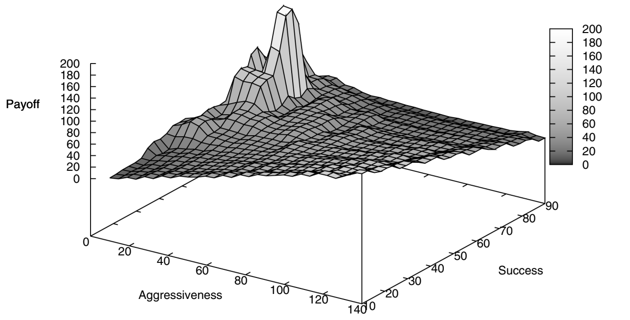


Fig. 6. Payoff $\mathcal{H}^{\mathsf{all}}_{\mathsf{rec}}(a, s)$ for reconnaissance attacks for combined IDS

no penalty for scanning at a higher aggressiveness rate, since the defender can only block an address. Consequently, this plot can continue out to whatever the practical upper limit for a is, a result which would correspond to the aggressive scanning we observe right now.

7 Conclusion

In this paper we have developed a new method for evaluating the performance of IDS based on an observable attack space, specifically the view of a harvesting or scanning attack that is available in flow logs that lack payload data. Our approach complements ROC-based analysis by enabling the creation of detection surfaces — models of an IDS' ability to detect different attacks. Moreover, we augment this analysis with a payoff-based metric. By incorporating payoffs, we are better able to characterize the deterrence offered by an IDS. In particular, instead of describing the detection of a system in terms of pure false positive and false negative rates, we are able to use payoff functions to calculate the gain that an attacker can expect from a certain type of attack. This also enables us to determine how high a false positive rate we must endure in order to limit the attacker's payoff to a target value.

Future work will focus on expanding the OAS approach to address different scenarios and parameters. First, our previous work on graph-based intrusion detection [6] considered the possibility of multiple bots being active simultaneously, and extending our OAS to account for this is a natural direction of future work. Second, generalizing from 30-second traffic samples to an approach considering multiple sample durations may provide additional detection capability [21]. Third, this work outlines two initial attack scenarios: harvesting and reconnaissance. However, a variety of other attacks may be considered and evaluated. In particular, different scanning strategies (such as topological scanning), bot command-and-control, and DDoS attacks all merit further investigation and similar payoff-based evaluation.

Several useful and, in some cases, discouraging results fall out of our analysis techniques as applied to SSH traffic observed on a very large network. For example, in §4.2 our analysis elucidated the complementary capabilities of detection using the size c of the largest component of a protocol graph [6] and the TRW scan detector r [11]. Consequently, there is good reason to use both simultaneously. Moreover, we showed that these detectors significantly outperform the server address entropy detector h, the graph-size detector g, and the degree-based detector d, for the stealthy attacks that form our observable attack space. That said, using our payoff analysis for acquisition attacks, we showed in §5.2 that r detection is primarily effective at detecting acquisition attacks with low payoff for the attacker, and so its utility for acquisition attacks is less compelling. In addition, we showed in §5.3 that to severely limit the attacker's acquisitions, the false positive rates that would need to be endured by any of the detectors we considered would be significant and, for a network of the size we studied, impractical. We showed how to derive more relaxed payoff limits that would enable near-zero false positive rates for an IDS.

References

1. Alata, E., Nicomette, V., Kaaniche, M., Dacier, M., Herrb, M.: Lessons learned from the deployment of a high-interaction honeypot. In: Proceedings of the 2006 European Dependable Computing Conference (2006)
2. Axelsson, S.: The base rate fallacy and the difficulty of intrusion detection. ACM Transactions on Information and System Security 3(3), 186–205 (2000)
3. Binkley, J.: An algorithm for anomaly-based botnet detection. In: Proceedings of the 2006 USENIX Workshop on Steps for Reducing Unwanted Traffic on the Internet (SRUTI) (2006)
4. Cárdenas, A., Baras, J., Seamon, K.: A framework for evaluation of intrusion detection systems. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy (2006)
5. Claffy, K., Braun, H., Polyzos, G.: A parameterizable methodology for internet traffic flow profiling. IEEE Journal on Selected Areas in Communications 13(8), 1481–1494 (1995)
6. Collins, M.P., Reiter, M.: Hit-list worm detection and bot identification in large networks using protocol graphs. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 276–295. Springer, Heidelberg (2007)
7. Collins, M.P., Reiter, M.K.: Anomaly detection amidst constant anomalies: Training IDS on constantly attacked data. Technical Report CMU-CYLAB-08-006, Carnegie Mellon University, CyLab (2008)
8. Gaffney, J., Ulvila, J.: Evaluation of intrusion detectors: A decision theory approach. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy (2001)
9. Gates, C., Taylor, C.: Challenging the anomaly detection paradigm, a provocative discussion. In: Proceedings of the 2006 New Security Paradigms Workshop, pp. 22–29 (2006)
10. Jung, J.: Real-Time Detection of Malicious Network Activity Using Stochastic Models. PhD thesis, Massachuesetts Institute of Technology (2006)

11. Jung, J., Paxson, V., Berger, A.W., Balakrishnan, H.: Fast portscan detection using sequential hypothesis testing. In: Proceedings of the 2004 IEEE Symposium on Security and Privacy (2004)
12. Kang, M., Caballero, J., Song, D.: Distributed evasive scan techniques and countermeasures. In: Hämmerli, B.M., Sommer, R. (eds.) DIMVA 2007. LNCS, vol. 4579, pp. 157–174. Springer, Heidelberg (2007)
13. Killourhy, K., Maxion, R., Tan, K.: A defense-centric taxonomy based on attack manifestations. In: Proceedings of the 2004 Conference on Dependable Systems and Networks (DSN) (2004)
14. Kreyszig, E.: Advanced Engineering Mathematics, 9th edn. J. Wiley and Sons, Chichester (2005)
15. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. In: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pp. 217–228 (2005)
16. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., Zissman, M.: Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In: Proceedings of the DARPA Information Survivability Conference and Exposition (2000)
17. Maxion, R., Tan, K.: Benchmarking anomaly-based detection systems. In: Proceedings of the 2000 Conference on Dependable Systems and Networks (DSN) (2000)
18. McHugh, J.: Testing intrusion detection systems: A critique of the 1998 and 1998 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Transactions on Information and Systems Security 3(4), 262–294 (2000)
19. Northcutt, S.: Network Intrusion Detection: An Analyst's Handbook. New Riders (1999)
20. Paxson, V.: Bro: A system for detection network intruders in real time. In: Proceedings of the 2008 Usenix Security Symposium (1998)
21. Sekar, V., Xie, Y., Reiter, M.K., Zhang, H.: A multi-resolution approach for worm detection and containment. In: Proceedings of the 36th International Conference on Dependable Systems and Networks, June 2006, pp. 189–198 (2006)
22. Shapiro, S., Wilk, M.: An analysis of variance test for normality (complete samples). Biometrika 52(3–4), 591–611 (1965)
23. Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Wee, C., Yip, R., Zerkle, D.: GrIDS – A graph-based intrusion detection system for large networks. In: Proceedings of the 19th National Information Systems Security Conference, pp. 361–370 (1996)
24. Stolfo, S., Fan, W., Lee, W., Prodromidis, A., Chan, P.: Cost-based modeling for fraud and intrusion detection: Results from the JAM project. In: Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (2000)
25. Tan, K., Maxion, R.: The effects of algorithmic diversity on anomaly detector performance. In: Proceedings of the 2005 Conference on Dependable Systems and Networks (DSN) (2005)