Coordinated Scan Detection

Carrie Gates
CA Labs, Islandia, NY
carrie.gates@ca.com

Abstract

Coordinated attacks, where the tasks involved in an attack are distributed amongst multiple sources, can be used by an adversary to obfuscate his incursion. In this paper we present an approach to detecting coordinated attacks that is based on adversary modeling of the desired information gain. A detection algorithm is developed that is based on solutions to the set covering problem, where we aim to recognize coordinated activity by combining events such that a large portion of the information space is covered with minimal overlap. We demonstrate this approach by developing a coordinated scan detector, where the targets of a port scan are distributed amongst multiple coordinating sources. In this case, the adversary wishes to gain information about the active hosts and ports on a particular network. We provide an algorithm that is capable of detecting horizontal and strobe scans against contiguous address spaces. We present experimental results from testing this algorithm in a controlled environment, demonstrating that it has an acceptably low false positive rate, discussing the conditions required to maximize the detection rate and the limitations of the approach.

1 Introduction

Adversaries have developed numerous techniques to avoid the detection of their activities by defenders. One such technique is the use of a coordinated attack. Braynov and Jadliwala [3] defined two types of co-operation that can be used in a coordinated attack: action correlation and task correlation. Action correlation refers to how the actions of one user can interact with the actions of another user. For example, a coordinated attack may require that one user perform a particular action so that another user can perform the actual attack. Task correlation, however, refers to the division of a task amongst multiple users, where a parallel (or coordinated) port scan is provided as an example. Braynov and Jadliwala focused their model on action correlation, whereas we address task correlation.

A security administrator might be interested in knowing about coordinated attacks against his system for two reasons: (1) he would want to know about this attack in much the same manner as he would want to know about any attack of this type, and (2) he might be even more interested in the attack given the lengths that the adversary has taken in order to remain undetected. Performing a coordinated attack requires greater resources and potentially greater technical skill, which might provide some indication to a defender as to the skill and motivation of the adversary.

A common example of a coordinated attack where the tasks have been distributed amongst multiple sources is the coordinated port scan. A port scan is a reconnaissance method used by an adversary to gather information about the responding computers and open ports on a target network. During a coordinated port scan, the adversary employs multiple sources, each scanning a portion of the target network. To the target, it appears as if multiple sources are each interested in some small portion of the network. While intrusion detection systems can be configured to recognize and report single-source port scanning activity, current systems do not recognize if any of these sources are collaborating.

This paper focuses on coordinated scan detection because it represents the form of coordinated attack that has been discussed most extensively. Section 2 describes the general problem of coordinated scan detection and provides some definitions. Our approach to solving the problem is presented in Section 3. Unlike previous detection approaches, which largely try to cluster packets according to similar features, we define various scan footprint patterns in the context of an adversary and the information he wishes to gain. We then develop an algorithm that takes advantage of the footprints that particular goals will generate. We explore the effects of parameters on our algorithm in Section 4, reporting on the overall results and discussing the limitations of our approach. We further discuss how the *effects* of time should be measured — as opposed to time itself — as a subsection on future work. A comparison to related work on detecting coordinated scans is provided in Section 5. Concluding comments are provided in Section 6.

2 Problem Description

The purpose of a port scan is to determine if a particular service(s) is available on a particular host(s). In order to avoid detection, stealth port scans were used, with early versions focusing on abusing the TCP protocol in order to avoid having the activity logged [6]. Later methods for avoiding detection included randomizing the order in which IPs are scanned, inserting time delays between scans and/or randomizing the length of the time delay between scans, and randomizing TCP packet field values (*e.g.*, sequence numbers, source ports) [12]. Scanners will also avoid detection by distributing their scan amongst multiple sources, each scanning a portion of the target network [18]. In addition to stealth, coordinated port scans can also gather information more quickly than single-source port scans by employing a parallel approach [12].

The definitions we use in this paper are:

Definition 2.1 *A* **target** *is a single port at a single IP address.*

Definition 2.2 *A* **scan** *is a set of connection attempts from a single source to a set of targets during some time interval.*

Definition 2.3 *A* **source** *is a computer system from which a scan originates.*

Definition 2.4 *A* **coordinated scan** *is a collection of scans from multiple sources where there is a single instigator behind the set of sources.*

Definition 2.3 specifies the physical origin of a scan, and not the IP number. Therefore, if the source of a scan changes its IP address in the middle of a scan, it is still a single source. However, as IP addresses are commonly used to represent a source, and as it is often difficult to determine if two different IP addresses represent the same source, in this paper we will use IP addresses to indicate the source, recognizing that we may do so erroneously in some cases.

The definition of coordinated scan presented here is consistent with those presented in both the hacker and academic literature. The first definition was presented by Staniford-Chen *et al.* [24], who defined *coordinated attacks* as "multi-step exploitations using parallel sessions where the distribution of steps between sessions is designed to obscure the unified nature of the attack or to allow the attack to proceed more quickly (*e.g.* several simultaneous sweep attacks from multiple sources)." Green *et al.* [10] later defined the term *coordinated attack* as "multiple IP addresses working together toward a common goal", and used this definition to describe behaviour they were observing on their network. Both of these definitions, as well as the definitions provided in other research (*e.g.*, [20], [29]), imply some level of coordination between the individual sources used in the scan.

Given that each source within the coordinated scan is itself performing a scan, coordinated scans essentially consist of multiple single-source scans. This is consistent with those definitions used by previous researchers, as described above. By distributing a scan amongst multiple sources, coordinated scans provide the advantage of speed, as various portions of the target network can be scanned in parallel, and discreteness, as the scan will appear to the target as multiple small scans, rather than one large scan [10][24].

We note that there is one publicly-available distributed scanner — Unicornscan [16] — that does not meet this definition of a coordinated scan. In this case the scanner distributes the various pieces of the TCP handshake across multiple agents; however, each agent needs to spoof the same source IP address and so, from the target's perspective, the scan appears to be a single source scan. We therefore do not consider this scanner to meet our definition.

The detection of single-source scans has been addressed by a number of different approaches in the literature (*e.g.*, [13], [15]). However, the recognition of a coordinated port scan (as a single entity, rather than recognizing solely the constituent parts) is a more difficult problem due to the lack of information on which to cluster. For example, in the case of detecting single-source scans, it is known that all packets will originate from the same source. Thus there is an *a priori* method for clustering packets into a set from which a decision can be made. However, in the case of coordinated port scans, no such obvious grouping based on one or more packet variables exists. In general, as stated by Braynov and Jadliwala [3], "cooperation through task correlation is difficult to discover. It is often the case that sensor data is insufficient to find correlation between agents' tasks. User intentions are, in general, not directly observable and a system trace could be intentionally ambiguous. The problem is further complicated by the presence of a strategic adversary who is aware that he has been monitored."

3 Detection Approach

Previous approaches to detecting coordinated port scans have focused on one of three methods (described in more detail in Section 5):

1. defining a coordinated port scan as having very specific characteristics so that scans can be easily clustered (*e.g.*, [29] [20] [28]),
2. clustering packets or alerts based on feature similarity using a machine learning approach (*e.g.*, simulated annealing [22], neural networks [25]), and
3. a manual analysis of network traffic, often aided by visualization approaches, to detect patterns in the traffic that are representative of coordinated scanning activity (*e.g.*, [4] [19]).

In the first instance, it becomes easy for an adversary to "game" the system if he knows how the defender has defined a coordinated scan. The second case is based on the assumption that coordinated scans will cluster together based on similarities between scans or packets; this assumption has never been verified. In fact, in [21], Staniford identified that the approach used in [22] clustered *events*, including denial-of-service attacks and misconfigurations as well as coordinated scans. The third case is reliant on having a person perform a manual analysis of the network traffic. It, too, can be gamed by generating enough events that the attack (coordinated scan) is obscured [5].

Rather than taking any of these three approaches, we instead base our approach on adversary modeling, building on the comments made by Braynov and Jadliwala [3] who state that "in order to detect such cooperation, one needs a clear understanding of agents' incentives, benefits, and criteria of efficiency." In the case of performing a port scan, an adversary is interested in gaining information about the existence of a particular service(s) on a particular host(s).

3.1 Adversary Representation

We observe that the primary goal of a scan is to gather information about the target network. One of the observables this generates for the defender is a footprint pattern. A footprint was defined by Staniford *et al.* [22] as "the set of port/IP combinations which the attacker is interested in characterizing." From the defender's perspective, the footprint is the set of IP/port pairs that he observes a particular source connecting to. We can characterize the footprint pattern generated by a scan C based on the following four characteristics: number of ports targeted ($|P|$), number of IP addresses targeted ($|A|$), the selection algorithm for how the IP addresses were selected (ς), and what camouflage (if any) is used to obfuscate the true target (κ).

The number of ports can be represented as either one or multiple, while the number of IP addresses can be similarly represented as one, multiple or all. The adversary can choose his targets based on one of three general approaches:

- randomly selected IP addresses, Ψ,
- based on some pattern in the IP addresses (such as only hosts known to exist, or only IP addresses that have a zero for the last octet), Φ, or
- some contiguous space, Σ.

Thus the selection algorithm can be chosen from the set $\{\Psi, \Phi, \Sigma\}$. The adversary can also choose to camouflage his true intent by scanning additional target IP addresses. These additional targets can be chosen using one of the following approaches:

- randomly scanning additional IP addresses, Ψ,
- scanning extra IP addresses that meet some property (*e.g.*, those IP addresses that are known to contain web servers), Φ, or
- scanning all IP addresses in some contiguous space or subnet, Σ.

Thus camouflage can be chosen from the set $\{\emptyset, \Psi, \Phi, \Sigma\}$, where $\emptyset$ represents that no camouflage was used.

Given these four characteristics, we can define a set of adversaries based on common values for each of the characteristics. These are presented in Table 1 (we will discuss the "Footprint Pattern" column shortly). Given that we defined 2 values for the number of ports, 3 for the number of IP addresses, 3 for the selection algorithm, and 4 for camouflage, this leaves us with 72 potential adversaries. However, not all combinations are sensible. For example, if someone is probing a single port on a single IP address, then there is no choice for selection algorithm, nor for camouflage. (While it is possible for an adversary to use camouflage here, it is highly unlikely that they would do so given that a probe is far less likely to be noticed than the increased activity from using camouflage.) Removing all of the combinations that are non-sensible results in only 21 adversaries. (More details on why particular adversaries were removed is available in [7].)

In addition to the four characteristics that define the scan footprint, and information goal, of the adversary, there are two additional characteristics of the footprint that can be derived and used from the defender's perspective: coverage and hit rate. That is, while the adversary can determine the selection algorithm and camouflage being used, the defender can only observe the IP/port pairs that are targeted by the scan without knowing which might be the target and which might be camouflage. Based on this overall scanning activity, the defender can observe the percent of his network that is targeted by the adversary (coverage), and the hit rate that the scan has within that targeted range. The coverage is calculated as

$$\zeta(C)\% = \frac{a_n - a_1 + 1}{|A|}$$

where $|A|$ is the number of IP addresses in the defender network, a_1 is the first IP address scanned (in IP space, not time) and a_n is the last IP address scanned. Given that the scan covered this portion of the network, the hit rate is then defined as the percentage of IP addresses within the scanned space that were actually targeted, and is defined as

$$\mathcal{H}(C) = \frac{|A_C|}{a_n - a_1 + 1}$$

where $|A_C|$ is the number of IP addresses on the monitored network that were targeted by scan C.

Adversary	Ports	Addresses	Selection	Camouflage	Footprint Pattern
1	One	One	—	None	$< 1, 1, 1/\|A\|, 1.0, \Phi, \emptyset >$
2	One	Some	Subnet	None	$< x, 1, x/\|A\|, *, \Sigma, \emptyset >$
3				Subnet	$< x, 1, x/\|A\|, *, \Sigma, \Sigma >$
4			Some	None	$< x, 1, x/\|A\|, *, \Phi, \emptyset >$
5				Some	$< x, 1, x/\|A\|, *, \Phi, \Phi >$
6				Subnet	$< x, 1, x/\|A\|, *, \Phi, \Sigma >$
7				Random	$< x, 1, x/\|A\|, *, \Phi, \Psi >$
8			Random	None	$< x, 1, x/\|A\|, *, \Psi, \emptyset >$
9	One	All	—	None	$< \|A\|, 1, 1.0, 1.0, \Sigma, \emptyset >$
10	Multiple	One	—	None	$< 1, y, 1/\|A\|, 1.0, \Phi, \emptyset >$
11			—	Some	$< 1, y, 1/\|A\|, 1.0, \Phi, \Phi >$
12			—	Subnet	$< 1, y, 1/\|A\|, 1.0, \Phi, \Sigma >$
13			—	Random	$< 1, y, 1/\|A\|, 1.0, \Phi, \Psi >$
14	Multiple	Some	Subnet	None	$< x, y, x/\|A\|, *, \Sigma, \emptyset >$
15				Subnet	$< x, y, x/\|A\|, *, \Sigma, \Sigma >$
16			Some	None	$< x, y, x/\|A\|, *, \Phi, \emptyset >$
17				Some	$< x, y, x/\|A\|, *, \Phi, \Phi >$
18				Subnet	$< x, y, x/\|A\|, *, \Phi, \Sigma >$
19				Random	$< x, y, x/\|A\|, *, \Phi, \Psi >$
20			Random	None	$< x, y, x/\|A\|, *, \Psi, \emptyset >$
21	Multiple	All	—	None	$< \|A\|, y, 1.0, 1.0, \Sigma, \emptyset >$

Table 1. The footprint patterns for the 21 types of adversaries, where x represents some number of IP addresses, y represents some number of ports, $|A|$ represents the number of IP addresses in the monitored network, and $*$ represents an unknown value.

We therefore define a characterization of the scan footprint using these six characteristics so that both the adversary's and defender's perspectives are included. We represent the footprint using the following tuple:

$$\mathcal{F} =< |P|, |A|, \zeta(C), \mathcal{H}(C), \varsigma, \kappa >$$

where $|P|$ is the number of ports, $|A|$ is the number of IP addresses, $\zeta(C)$ is the coverage for scan C, $\mathcal{H}(C)$ is the hit rate for scan C, ς is the selection algorithm (chosen from the set $\{\Psi, \Phi, \Sigma\}$) and κ is the camouflage approach (chosen from the set $\{\emptyset, \Psi, \Phi, \Sigma\}$). This footprint characterization can represent horizontal, vertical, strobe and block scans, as defined by Staniford *et al.* [22], as well as probes. A generic form of a scan footprint is $< x, y, \frac{x}{|A|}, *, *, \emptyset >$ where $*$ represents an unknown value, $x \geq 1$ and $y \geq 1$. For a more specific example, a horizontal scan targets a single port across multiple IP addresses and is represented by the tuple $< x, 1, \frac{x}{|A|}, *, \varsigma, \kappa >$ where $x \geq m$ and m is a user-defined minimum number of IP addresses required for a scan to be considered as a horizontal scan. (For example, this tuple can represent the definition of a horizontal scan used by Yegneswaran *et al.* [29] by setting $m = 5$.)

The footprint characterization presented represents the footprint information for all scans, including coordinated scans. In the case of coordinated scans, the overall footprint results from combining each of the individual scans. However, there are three additional characteristics for coordinated scans that are not in single-source scans: the number of sources, the amount of overlap and the algorithm used to distribute the targets amongst the sources. Thus a coordinated scan can be represented by the tuple $< \mathcal{F}, |S|, \theta, \mathcal{A} >$ where $\mathcal{F}$ is the overall footprint characterization, $|S|$ is the number of scan sources, θ is the overlap and $\mathcal{A}$ is the scanning algorithm. Overlap is defined as the number of targets in common between two sources:

$$\theta(C_0, C_1) = \frac{|A_0 \cap A_1|}{|A_0 \cup A_1|}$$

where A_i is the set of target IP addresses for scan C_i. The scanning algorithm, $\mathcal{A}$, is the algorithm used to determine how the targets are distributed amongst the scanning sources, and includes an interleaved series of length n, randomly distributed, and sequential blocks. The overlap, scanning algorithm and number of sources are included in the representation of a coordinated scan because they affect the footprints observed (of both the single-source scans and the overall coordinated scan) by the defender.

3.2 Set Covering Approach

We take advantage of the footprint information to generate an approach to detecting coordinated scans. As a coordinated scan consists of multiple single-source scans, we first use some approach to detecting single-source scans. The set of all scans detected during some time period provides us with our input set.

Given a set of scans and the adversary models given above, the problem can be reduced to finding a subset of scans such that one of the footprint patterns is detected. We focus here on detecting nine of the 21 adversary types. Specifically, we restrict our problem space to detecting those adversaries who perform either a horizontal or strobe scan of a large enough contiguous space. Thus we can detect adversaries 2, 3, 6, 9, 12, 14, 15, 18 and 21.

The result of restricting the problem space in this manner is that we can also reduce the footprint information required to consist of only the ports and IP addresses targeted by the scan, rather than needing to specifically define, for example, the selection algorithm or camouflage approaches used. This is an important aggregation because, from the network border, a defender can only determine what target IP/port pairs a source attempted to access, and reduces a scan footprint to be the same as that defined by Staniford *et al.* [22]. The end result is that the input required for each scan is just the set of IP/port pairs targeted by that source, and the input set consists of the set of all scans during some time period.

We assume that the scans are detected using traffic collected from the border of the monitored network. This assumption provides us with the view required to determine the scan footprint geometry required as input to the coordinated scan detection algorithm.

The problem of detecting a coordinated port scan based on footprint geometry can be compared to the set covering problem. The set covering problem is defined as: given all the sets that are available in some space, choose the minimum number of sets such that the entire space is covered.

Grossman and Wool [11] surveyed eight approaches to solving the set covering problem, implementing and testing them on a series of set covering problems. The algorithm that generally performed the best was "Altgreedy", which is a variation on the greedy algorithm. Altgreedy works by first choosing the variable with the largest number of unsatisfied inequalities, Δ_j, and adding it to the solution set. (That is, Altgreedy chooses the set that covers the largest number of elements that are still not covered by some other set in the solution). Ties in variables are broken by choosing the variable with the smallest index. At this point, Altgreedy diverges from the traditional greedy algorithm to reject variables from the solution set. It chooses the variable that is the "most redundant" (covers the most elements that are also covered by other sets) in the solution set and removes it. This process repeats until the number of unsatisfied inequalities in the solution set is either equal to $\Delta_j - 1$, or is as close to this value as possible while not exceeding it.

It should be noted that the problem of detecting coordinated port scans deviates from the set covering problem in the following ways:

1. We are interested in minimizing the overlap between each of the sets (scans). The distributed scanning tools that we analyzed — DScan [2] and NSAT [17] — did not exhibit any overlap in the target footprint between scanning sources.

2. We do not require that the entire space be covered, but rather focus on some minimum amount of contiguous coverage. This is because a coordinated scan might focus on some subset of the monitored network. We note that the tools that we analyzed scanned contiguous network space.

3. We do not require that *every* IP within the (contiguous) space be covered. This is because our algorithm takes into consideration the possibility of missing data or dropped packets.

Thus we are not interested in detecting the smallest number of sets (scans) that cover the largest amount of space space. Rather, we are interested in detecting that multiple sets (scans) fit together in such a manner that some large portion of the entire space is covered *while minimizing the amount of overlap between each of the sets*. We therefore modify and extend the algorithms for solving the set covering problem, in particular the Altgreedy approach, to consider these additional restrictions.

To support our usage of the set covering paradigm, we demonstrate the detection capabilities of our algorithm in Section 4, which evaluates the algorithm. We note that we used two distributed scanning tools — available "in the wild" and not developed by the authors — to test our use of the set covering approach to coordinated scan detection.

3.3 Algorithm

Like the Altgreedy approach, our approach consists of slowly building a subset through the addition of new scans and the removal of scans that are the most redundant. However, while Altgreedy chooses the set that covers the most space, we instead choose the scan that covers the least number of destination IP addresses. Thus, rather than building our set by continually adding in the largest sets, we instead add in the smallest sets first. We make this change because we find it more likely that small scans will be part of a coordinated scan. This is because a large scan is likely to already cover all or most of the IP space, as it is likely that in any

given set of scans there was at least one horizontal scan that covered most of the space. In contrast, since we are looking for coordinated scans, the adversary will have divided the IP space amongst multiple sources, and so it is likely to consist of several smaller scans. We therefore start by adding in the smaller scans first.

In fact, we add a condition on the scan that is added to the set. We choose the next scan based on two contrasting goals: to maximize coverage while minimizing overlap. Thus we actually choose to add the smallest scan that also has the smallest overlap with any other scans in the set. We choose to minimize the overlap based on the assumption that an adversary will not want to scan the same target multiple times as it provides no additional information. However, they might add some overlap to their scans in order to decrease the likelihood that their activity will be detected.

After adding a new scan to the set containing a possible coordinated scan, we check to see if it "covers" some other scan in the set. If there is another scan that substantially overlaps with the scan that we are about to add (where substantial is arbitrarily defined as 95% of the IP addresses also being present in the newly added scan — future work should investigate the effect of varying this value), then we remove the covered scan from the set. We check the entire set to determine if the amount of overlap has exceeded some threshold. If the overlap is too large, then we remove the scan with the largest overlap with other scans in the set. The overall Altgreedy-inspired portion of the algorithm is presented in Figure 1.

When we remove a scan from the set, we do not immediately remove it from further consideration. Instead, we "remember" that it has been rejected once already, and if it is added to the set again later and then rejected again, we reject it permanently from further consideration. This portion of the algorithm is presented in Figure 2.

We continue to loop, adding and removing scans, until all of the scans have been examined and either added to the set or permanently rejected. The resulting set is then examined for the presence of a coordinated port scan, which means that the set meets the following four conditions:

1. there is more than one scan (i) in the set, $|S| > 1$,
2. the contiguous coverage of the network is greater than the minimum acceptable network coverage, $\zeta(C) > X\%$,
3. the overlap in the set is less than the maximum acceptable overlap, $\theta < Y\%$,
4. the hit rate within the covered area is at least some user defined minimum, $\mathcal{H}(C) \geq Z\%$,

where X, Y and Z are defined by the user. (Section 4 discusses the effect of different values for these three variables given different environments and scan characteristics.)

Input: A set of scans A
Input: Maximum percent of overlap MAXOVERLAP
Input: Minimum percent of network covered MINCOVERAGE
Output: Set S of scans forming a coordinated scan
  $S \leftarrow$ smallestScan(A)
  $rejected \leftarrow \emptyset$
  $rejectedOnce \leftarrow \emptyset$
  repeat
    $i \leftarrow$ smallestOverlap($A - rejected$, S) {*get scan with smallest amount of overlap*}
    if newlyCoveredIPs(S,i) > 0 **then**
      $S \leftarrow S \cup \{i\}$ {*add scan to solution set if it covers at least 1 new IP*}
    else
      Run rejection algorithm($rejected$, $rejectedOnce$, i) {*possibly reject scan* }
    end if
    if overlap(S) $>$ MAXOVERLAP **then**
      $i \leftarrow$ greatestOverlap(S) {*get scan with most overlap with other scans* }
      $S \leftarrow S - \{i\}$ {*reject scan from solution set*}
      Run rejection algorithm($rejected$, $rejectedOnce$, i) {*possibly reject scan* }
    end if
  until $S \cup rejected == A$

Figure 1. Altgreedy [11] portion of algorithm.

If the set does not form a coordinated scan, we check to see if it might be due to the influence of noise (single-source scans that are not actually part of the coordinated scan) on a coordinated scan that does not span the entire monitored network. In order to address this, the largest gap in the monitored space (largest contiguous space of IP addresses not covered by a scan in the set) is found, and the set split into two halves. All those scans in the smallest half are removed from the set, and the remainder is checked for the presence of a coordinated port scan. This process continues until either a coordinated scan is found or there are not enough IP addresses covered to form a coordinated scan. This algorithm is provided in Figure 3.

We extended the basic algorithm in Figure 1 to detect the presence of strobe scans (Adversaries 14, 15, 18 and 21). We do this by first checking for the presence of any coordinated scans on each individual port, and then combining the ports. The scans for two different ports are combined into a single coordinated scan if they meet one of two conditions:

1. the two sets of scans cover nearly the same set of target IP addresses (with an agreement of 95% or better, allowing for the possibility of missing data), or
2. the two sets contain nearly the same scan sources (with an agreement of 95% or better).

Input: A set of scans $rejected$
Input: A set of scans $rejectedOnce$
Input: A scan i
Output: Set of scans $rejected$ (possibly modified)
Output: Set of scans $rejectedOnce$ (possibly modified)
  if $i \in rejectedOnce$ **then**
    rejected $\leftarrow$ rejected $\cup$ $\{i\}$ {*reject scan so it can no longer be used* }
  else
    rejectedOnce $\leftarrow$ rejectedOnce $\cup$ $\{i\}$ {*mark scan as rejected, but allow further use* }
  end if

Figure 2. Rejection algorithm.

This process is repeated for each set of ports (so a strobe scan can be recognized even when it consists of more than two ports). The resulting aggregated sets are then examined for the presence of a coordinated scan using the same four criteria that were defined earlier.

If a coordinated scan is found, we provide the source IP addresses of the scan to the user and then remove those scans from the set of all scans. We repeat the algorithm to determine if there is a second coordinated scan in the set. The algorithm exits when no more coordinated scans are found. This algorithm is provided in Figure 4.

Input: A set of scans A
Input: Maximum percent of overlap MAXOVERLAP
Input: Minimum percent of network covered MINCOVERAGE
Output: Set $results$ containing a coordinated scan
  $results \leftarrow \emptyset$
  $S \leftarrow$ Altgreedy Portion of Algorithm(A, MAXOVERLAP, MINCOVERAGE)
  while overlap(S) $>$ MAXOVERLAP **do**
    $i \leftarrow$ greatestOverlap(S) {*remove scans with too much overlap* }
    $S \leftarrow S - i$
  end while
  { *while we have not found a coordinated scan, and the coverage is still large enough that there might be one, keep looking* }
  while (! isDPS(S)) && (coverage(S) $>$ MINCOVERAGE) **do**
    gap $\leftarrow$ largest set of contiguous IP addresses not covered in S
    $S \leftarrow$ scans in largest subset of S when split into two sets separated by gap
  end while
  if isDPS(S) **then**
    $results \leftarrow S$
  end if

Figure 3. Detection Algorithm

4 Experimental Results

4.1 Metrics

The usual metrics for measuring the performance of an intrusion detection system consist of true and false positive rates, indicating the probability that an intrusion would be detected (or that a non-intrusion would cause an alert). The detection rate is the conditional probability that an alert A was generated given that there was intrusive behaviour I, $P(A|I)$. The false positive rate is the conditional probability of an alert A given that intrusive behaviour is *not* present $\neg I$, $P(A|\neg I)$. However, coordinated scans consist of multiple events that need to be detected, and so this metric needs to be modified. We choose as our unit of analysis the scan, and define true positives as those scans that were correctly flagged as being part of a coordinated scan. Similarly, we define a false positive as a scan that was not part of a coordinated scan, but was flagged as being so.

We also define an "effective" false positive rate, which is the average number of coordinated scans that are reported per data set that are not "correct". We define correct here to mean that we included at least 50% of the correct sources in the reported coordinated scan. As the true impact of the false positive rate is on the amount of wasted effort by the system administrator, we use these values to indicate how many incidents a defender can be expected to investigate.

4.2 Experimental Design

Our experiments consist of performing coordinated scans in an isolated environment and capturing the network traffic for later analysis [8]. This provides us with ground truth. We then inject the results from these scans into a set of noise data, described in more detail below. The isolated test environment we use is the DETER network [26], which uses the Emulab software [27]. We were able to use up to 100 agents in this environment, along with using two nodes to act as a /16 subnet. A monitoring node running `tcpdump` was inserted between the agents and the /16 subnet in order to capture all scan traffic entering the monitored subnet. The scan information was extracted using the algorithm provided by Gates *et al.* [9]. This scan information was then injected into a set of "noise" scans gathered from a live network.

We use two scanning algorithms — DScan [2] and NSAT [17] — that are available "in the wild" in order to avoid injecting our personal biases into how a coordinated scan algorithm should perform. DScan distributes target IP/port pairs randomly among the different sources, while

Input: A set of scans A
Input: Maximum percent of overlap MAXOVERLAP
Input: Minimum percent of network covered MINCOVERAGE
Output: A set S of coordinated scans
$S \leftarrow \emptyset$
$P \leftarrow$ all unique ports in A
repeat
for every $p \in P$ **do**
results$_p$ $\leftarrow$ Detection Algorithm(A, MAXOVERLAP, MINCOVERAGE)
end for
{*Check for strobe scan — 2 ports mostly cover same IPs or have same sources*}
for every $\{i, j\} \in P, i \neq j$ **do**
if (coverage(i, results$_i$) $\approx$ coverage(j, results$_j$)) || (results$_i \approx$ results$_j$) **then**
results$_i \leftarrow$ results$_i$ + results$_j$
results$_j \leftarrow \emptyset$
end if
end for
for every $p \in P$ **do**
$S \leftarrow S \cup \{$ results$_p$ $\}$
$A \leftarrow A$ - results$_p$
end for
until results$_p = \emptyset \; \forall \; p \in P$

Figure 4. Strobe Algorithm

NSAT uses an interleaving pattern of length n where there are n sources. Neither algorithm had the ability to specify any overlap, and so we added this feature to DScan in order to test the ability of our detection algorithm to perform given this camouflage behaviour.

The noise set was gathered from four sparsely populated /16 subnets on the same /8 network in the .gov domain. Flow level network traffic was gathered at the border of the network for the month of March, 2005, and analysed for the presence of single-source scans using the algorithm by Gates *et al.* [9]. The number of scans on each network ranged from 1069 to 1253, the majority of which were horizontal (74%) or strobe (25%). The result was four different data sets, each consisting of scans that were detected using the same algorithm as on the isolated network and that contained the same information needed for input to the detection algorithm (that is, a source IP address and its footprint geometry).

Noise sets of the desired size were extracted by first randomly choosing which of the four data sets to use, and then randomly choosing a start point within that data set, taking the next x scans in order to generate a noise set of size x. This procedure was performed rather than a random sampling in order to preserve any temporal relationships that might exist within the data.

In these experiments we control six different variables (the size of the subnet being scanned — a /16 subnet in these experiments — is actually a seventh variable): the coverage and overlap of the scans, the scanning algorithm, the number of scanning sources, the number of ports scanned, and the number of noise scans. The range of values for coverage is 10% to 100%, overlap is 0% to 20% (for DScan only), number of sources is 2 to 100, number of ports is 1 to 5 and the number of noise scans is 100 to 1000 (in sets of 100).

We note that time is not a variable that is considered here, as it is not part of the detection algorithm. The impact of time, in particular as it relates to single-source scan detection and real-time or near real-time coordinated scan detection, is discussed further in Section 4.6.

We performed 48 experiments using the extreme values. That is, for NSAT, which has no overlap, we performed experiments where the coverage was 10% and 100%, the number of scanning sources was 2 and 100, the number of scanned ports was 1 and 5, and the number of noise scans was 100 and 1000. This resulted in 16 experiments. Another 16 were performed for DScan with no overlap, and a further 16 for DScan with 20% overlap. We performed an additional 39 experiments where the values were chosen randomly from the ranges provided.

4.3 Results

We ran our detection algorithm on the 87 data sets that were generated for experimental validation. We defined a correct detection as at least 50% of the single-source scans that were identified as part of the coordinated scan being true positives. The detector recognized 61 of the 87 coordinated scans (70%). In 47 of the 61 cases, every scan that was part of the coordinated activity was detected, with only 14 "partial" detections (with the percentage of the sources that were recognized ranging from 86.4% to 99.0%). For the remaining 26 cases, none of the sources were recognized as being part of a coordinated scan. Thus this approach tends to perform either very well or not at all.

Examining the results in greater detail, we find that the detector performed poorly at detecting some of the extremes. In particular, of the 48 cases where the extreme values for the ranges were used, only 28 of the scans were detected (58%). However, for the 39 cases where the values for each variable were chosen randomly, 33 of the scans were detected (85%).

We can examine the performance of the detector by building a regression model. We use a logistic regression because the results are so strongly bimodal. The result from the model can be interpreted as the probability that the coordinated scan will be detected given the values of the variables. We then use the Akaike Information Criterion [1] in

order to remove variables that do not contribute to the detection rate. The resulting model is:

$$\hat{P}(\text{coordinated scan is detected}) = \frac{e^{\hat{y}}}{1+e^{\hat{y}}}$$

where

$$\hat{y} = -1.5920551 + 0.0312865x_1 - 0.0026240x_4 + 0.0205417x_5 + 0.5760024x_6$$

where x_1 is the coverage of the network, x_4 is the number of noise scans, x_5 is the number of scanning sources, and x_6 is the number of scanned ports. The two variables that were removed were the amount of overlap (x_2) and the scanning algorithm (x_3), indicating that they have little impact on the detection rate. Every variable in the equation is significant at $p < 0.05$. The positive co-efficients for x_1, x_5 and x_6 indicate that as these values increase, so does the detection rate. However, as the number of noise scans increases (x_4), the probability of detection decreases. An example of how the detection rate varies with changes in the network coverage (x_1) and number of noise scans (x_4) using this equation is provided in Figure 5. This graph shows that the scan needs to cover at least 68% of the network, and the input set contain at most 500 scans, in order to obtain a detection rate of at least 80%. For the false positive rate, there were 27 experiments that had no false positives and 76 experiments (87%) where the false positive rate was ≤ 0.01. The maximum value was 0.08, with the median = 0.0025 and the mean = 0.0068. We model this using a linear regression:

$$fp = -0.007494 + 0.00005559x_1 + 0.0004216x_2 + 0.00005877x_5 + 0.001903x_6$$

where x_1 is the percentage of the network covered, x_2 is the overlap, x_5 is the number of sources and x_6 is the number of ports. This indicates that an increase in any of these variables results in an increased false positive rate.

The false positive rate does not provide a true indication of administrator overhead. This is because the overhead from examining 5 coordinated scans with 3 sources each is greater than the overhead of examining a single scan with 15 sources, even though the false positive rate for both sets is the same. A false positive in this case is any coordinated scan where fewer than 50% of the sources are true positives. For our 87 examples, there were 39 cases (45%) where the effective false positive rate was zero. That is, any false positives that occurred were added to a correctly recognized coordinated scan. In 36 cases the effective false positive rate was 1, and the largest rate observed was 8, giving an average effective false positive rate of 0.95 and a median rate of 1. Thus on average an administrator will examine one scan that was not coordinated for each use of the detector. These values indicate that the administrative overhead from false positives is low.

4.4 Gaming the System

To avoid detection, the adversary needs to take one (or more) of the five steps described below. For each approach to avoiding detection (all of which consist of ensuring that the detection thresholds are not exceeded), we describe the loss of information suffered by the adversary in order to avoid detection.

scanning fewer ports: As the number of ports included in the scan increases, so does the detection rate. Thus to reduce the chances of detection, an adversary can only scan a single port, potentially reducing his information gain.

using fewer scanning sources: As the number of sources used in a scan increases, so does the probability that the scan will be detected. In order to avoid this, an adversary needs to use fewer sources. The tradeoff is that it will now take longer for the adversary to scan the target space, thus increasing his workload. In addition, this goes against the principal of using a coordinated scan if the original intention was stealth.

scanning less of the network: The larger the portion of the network that is scanned by the adversary, the greater the probability that he will be detected. This reduces the information gain by the adversary.

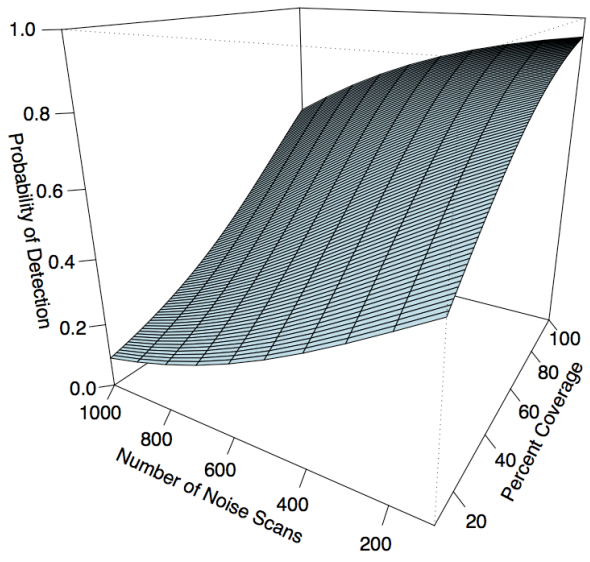

Figure 5. The effect of the network coverage (x-axis) and number of noise scans (z-axis) on the overall detection rate (y-axis), where the number of ports scanned = 2 and the number of scanning sources = 50.

scanning under the hit rate: The adversary can choose to randomly drop enough targets from the scan list that his scan will not be detected. However, this reduces the amount of information that the adversary then obtains. Additionally, if the targets are not dropped in a uniform random fashion, it is still possible that a subpart of his scan will be detected.

scanning more slowly: The adversary can have each source scan with enough time between the start of each scan that not all of the sources will fall in the same scan window being analysed. This increases the time required to perform the scan, and therefore adversary workload.

4.5 Limitations and Future Work

The input for this algorithm consists of single-source port scans. Thus, if an attacker can avoid detection by the single-source scan detector, then he also avoids detection by the coordinated scan detector. In fact, this idea underlies the paper by Kang *et al.* [14], where they use a coordinated scan, called z-scan, to gather information while avoiding detection by the Threshold Random Walk algorithm of Jung *et al.* [13] (which can detect a single-source scan in as few as five targets).

However, while the algorithm presented in this paper was tested using the input from a single-source scan detection approach based on grouping packets by the source IP address, this is not a required input algorithm. If a different approach to detecting scans were employed, such as that by Whyte *et al.* [28] who use exposure maps, where any hit of a non-existing IP/service is recorded, this would avoid the limitation identified. Similarly, the input could consist of all activity from any source that targeted even a single non-responsive IP/service pair, regardless of the number of valid connection attempts performed. However, testing the performance of this algorithm given a different underlying structure for determining the input remains as future work.

4.6 The Effect of Time and Future Work

The algorithm in this paper abstracts away the notion of time as the analysis was performed in an off-line manner using a month as the time scale of interest. In this subsection, we argue that time itself is not a valid variable to be considered; rather, the other characteristics that are generally thought to be time-related are the correct parameters to use. For example, our training and testing data sets covered one month of data; however, this time frame was chosen because it provided over 1000 scans for use in testing. (In fact, the number of scans ranged from 1069 in one data set to 1253 in another.) On other networks, it might be the case that 1000 scans occur over either much shorter or much greater time periods. It is for this reason that our testing methodology did not include time as a variable, but rather the number of scans being tested. In fact, in order to make valid comparisons between networks, and to be able to make *any* comment on how well the algorithm will perform, we *must* abstract away the notion of time, focusing instead on those aspects of the environment that affect the detection rate regardless of the time frame used.

This observation refocuses the issue of time. At its simplest, the question being asked is: how much of the coordinated scan is required before it can be detected? This question has been partially addressed in this paper — we have examined how much of the network needs to be covered in order to detect the coordinated scan (using the values of 10% of contiguous network space up to 100% coverage of the network). Additionally, detection is affected by the amount of data that can be "missing". We identify here three aspects to missing data:

1. Assuming that each source scans at approximately the same time (that is, with few or no noise scans occurring in between the individual scans of the co-ordinated scan), how much of each scan is required before they can be recognized as forming a coordinated scan? This relates to the variable that was intended to allow for missing data, and that was arbitrarily set at 95% for these experiments. The effect of different values for this variable on the detection and false positive rates needs to be evaluated.

2. Assuming that each scan occurs sequentially, and assuming again that there are few or no noise scans occurring at the same time, how many scans are required before the co-ordinated scan can be detected? This is related to the scanning algorithm. (For example, are targets distributed randomly among the scanning sources? Or, does each source scan a contiguous block of network space?)

3. What is the impact of removing the assumption above that there are few noise scans occurring at the same time as the coordinated scan? What is the impact of the number of noise scans occurring at the same time, and the number of detected targets for each of those scans?

Ideally, we would like to evaluate is how well the algorithm performs "in real time" in an operational setting. Given an operational setting, the algorithm needs an additional layer that represents a sliding window over the detected scans. This adds the following complexity:

- Once a (single source) scan has been detected, we need to continue to record the targets of the scan for the du-

ration of the scan. This continuous updating potentially impacts when and how often the detection algorithm should be used. For example, the algorithm could be run every time a currently recorded scan is updated with some number of new targets, or it might only be used when a new scan is detected.

- The impact on the detection rate of adding scan information as it occurs needs to be examined. More specifically, how does the completeness of the noise scans impact the detection rate? For example, if we have a completed coordinated scan in our sliding window, is it more easily detected if all of the other scans have also completed (representing fast scans), or if we have only a few targets for each of the noise scans (representing low and slow scans)? [1] What is the interaction between the completeness of the noise scans and the completeness of the coordinated scan?

- We need to determine how the scans in the sliding window should be ordered, as this impacts when a scan is removed from the window and replaced with a "newer" scan. For example, the ordering could be by the time of detection for the scan; however, this does not address slower versus faster scans. Thus it might be the case that the sliding window should be ordered by the most recent scan update (that is, by the time that each source scanned its most recent target).

- Related to ordering the scans in the sliding window is the impact on the detection rate of the aging out of older scans. This might, for example, age out portions of the coordinated scan. That is, if the sources scan sequentially, then it is possible that some of the individual scans will be removed from the sliding window and so not be used as input to the detection algorithm. This relates back to the issue identified above of how to detect a coordinated scan in the presence of missing data.

The algorithm presented in this paper requires that some contiguous portion of the network space be covered. Deploying this in an operational setting *implies* a notion of time; however, we abstract this notion into the core question of: How many targets of the coordinated scan must be observed in order to make a positive identification of the coordinated scan? We have identified some of the complexities behind providing a comprehensive answer to this question, and note that addressing this issue remains as future work.

[1] In our noise data sets, the shortest scan occurred over only a few seconds, while the longest scan was over seven days in duration.

5 Comparison to Related Work

Robertson *et al.* [20] contend that their method detects coordinated port scans. However, this detection is based on the assumption that port scans that originate from source IP addresses that are located close together (*e.g.*, as part of the same class C subnet) are related, rather than assuming that the scanner may have compromised a number of machines on various subnets.

Yegneswaran *et al.* [29] also claim to detect coordinated port scans. They define a coordinated (or distributed) port scan as "scans from multiple sources (five or more) aimed at a particular port of destinations in the same /24 subnet within a one hour window". This definition misses several possible coordinated scans, such as scans from fewer than five sources, or scans where each source scans in a separate hour. In addition, completely unrelated sources might scan the same port on the same /24 subnet within the same hour, simply by chance. The probability of this happening increases with the number of scans observed on a subnet and the release of new exploits.

We do not provide a comparison of our approach to the above two approaches because the definition of coordinated scans differs significantly. In each of the previous cases the coordinated scans as defined are a subset of our definition of coordinated scans. The result is that any comparison using our data set would be unfair due to these different definitions.

Conti and Abdullah [4] also claim to have a method for detecting coordinated scanning activity. They have developed an approach to visualizing network traffic, which they have tested on human subjects to determine the features that are recognizable in the traffic. In addition to identifying scans, they claim that coordinated scans are easily recognized by the pattern they create. However, they did not provide any experimental evidence for this, nor any example visualizations. In addition, they have not published any investigation of how well such scans can be recognized given a background of other scanning activity as well as legitimate network activity. While this work may possibly detect the same types of coordinated scans as our approach, it is not an automated technique but rather requires human analysis.

Staniford *et al.* [22] present a method for the detection of stealth port scans, where stealthy port scans are defined as port scans with properties that would elude traditional intrusion detection systems, such as long delays between scans and using numerous source IPs in a coordinated port scan. Their approach can be divided into two distinct sections: the network anomaly detector (called Spade) and the correlation engine (called Spice). The network anomaly detector determines how anomalous a packet is (using an entropy-based measure), passing it on to the correlation engine, Spice, if it is sufficiently anomalous. Spice inserts

the packet into a graph, where the nodes represent packets and the connections between nodes contain weights indicating the strength of the relationship between the two nodes (packets). The weights are based on a combination of four feature characteristics (equality, proximity, separation, and covariance). A simulated annealing procedure is used on each of four connections to generate clusters. In the final graph, all edges with weights less than a certain threshold are dropped, and the remaining subgraphs represent interesting network events.

Spice was not designed specifically to detect coordinated port scans. Rather, it clusters together network packets that have similar properties. Each cluster can represent a variety of events, including coordinated port scans, port scans with long delays between each scan, distributed denial of service attacks, and network misconfigurations [21]. The true and false negative and positive rates for this approach have not been reported in the literature.

Staniford *et al.* state in [23] that "Such distributed scanning has already been seen in the wild—Lawrence Berkeley National Laboratory received 10 during the past year." The scans in this paper were recognized by security analysts due to the sudden increase in the number of sources scanning, and the small amount of overlap between the scans [19].

While the approach by Staniford *et al.* [22] is likely to detect coordinated scans following the same definition as provided in this paper (as opposed to only a subset of possible coordinated scans), they have not provided sufficient detail in their paper to replicate their results.

6 Conclusions

In this paper we presented an approach to detecting coordinated port scans. Unlike previous approaches, which have been based on clustering and manual analysis, our approach uses adversary modeling. We define a set of adversaries based on the information that they are attempting to acquire, and then use a set covering approach to recognize when multiple sources, when pooled together, will have obtained that information.

We developed an algorithm capable of detecting coordinated scans that have a horizontal or strobe footprint across a contiguous network address space. We tested this algorithm through 87 different experiments where coordinated scans were performed in an isolated environment, combining the network traffic traces with those collected from live networks. We controlled for six different variables in these tests, providing regression equations that model our results. The detection rate varies considerably with four of the six variables, while the false positive rate remains generally low (less than 1%). We defined a new measure, the effective false positive rate, which measures the number of false positive coordinated scans detected that an administrator would need to investigate, and found that our approach has an average of 1 false coordinated scan per data set. We identified the limitations of this approach and discussed how an adversary can remain undetected and the cost (in terms of the lack of information gain) for doing so.

Finally, we discussed the effect of time on our detection rate, arguing that time should not be considered as a variable; rather, the other characteristics that are generally considered to be related to time should be identified and tested further. Such an abstraction will allow the results to be generalizable to other environments (such as networks that see many more or many fewer scans during some time period, or networks that see primarily fast scans versus primarily low-and-slow scans).

References

[1] H. Akaike. Information theory as an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaksi, editors, *Proceedings of the 2nd International Symposium on Information Theory*, pages 267 – 281, Budapest, Hungary, 1973.

[2] D. R. (Anthraxx) and B. P. (Kolrabi). DScan software. `http://www.u-n-f.com/dscan.html`, 2002. Last visited: 2 July 2002.

[3] S. Braynov and M. Jadliwala. Detecting malicious groups of agents. In *Proceedings of the First IEEE Symposium on Multi-Agent Security and Survivability*, Drexel University, Philadelphia, PA, USA, August 2004.

[4] G. Conti and K. Abdullah. Passive visual fingerprinting of network attack tools. In *Proceedings of 2004 CCS Workshop on Visualization and Data Mining for Computer Security*, pages 45 – 54, Washington, DC, USA, October 2004.

[5] G. Conti, M. Ahamad, and J. Stasko. Attacking information visualization system usability overloading and deceiving the human. In *Symposium on Usable Privacy and Security*, Pittsburgh, PA, USA, July 2005.

[6] Fyodor. The art of port scanning. *Phrack Magazine*, 7(51), 1997.

[7] C. Gates. *Co-ordinated Port Scans: A Model, A Detector and an Evaluation Methodology*. PhD thesis, Dalhousie University, Feb 2006.

[8] C. Gates. A case study in testing a network security algorithm. In *4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, Innsbruck, Austria, March 2008.

[9] C. Gates, J. J. McNutt, J. B. Kadane, and M. Kellner. Scan detection on very large networks using logistic regression modeling. In *Proceedings of the IEEE Symposium on Computers and Communications*, Pula-Cagliari, Sardinia, Italy, June 2006.

[10] J. Green, D. Marchette, S. Northcutt, and B. Ralph. Analysis techniques for detecting coordinated attacks and probes. In *Proceedings of the Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, California, USA, April 1999. Usenix Association.

[11] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, 101(1):81 – 92, 1997.

[12] hybrid. Distributed information gathering. *Phrack Magazine*, 9(55), 1999.

[13] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 211 – 225, Oakland, California, USA, May 2004. IEEE Computer Society.

[14] M. G. Kang, J. Caballero, and D. Song. Distributed evasive scan techniques and countermeasures. In *Proceedings of Fourth GI International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment*, Lucerne, Switzerland, July 2007.

[15] C. Leckie and R. Kotagiri. A probabilistic approach to detecting network scans. In *Proceedings of the 2002 IEEE Network Operations and Management Symposium*, pages 359 – 372, Florence, Italy, April 2002.

[16] R. E. Lee. Unicornscan. `http://www.unicornscan.org/`, 2007. Last visited: 12 December 2008.

[17] Mixter. Network security analysis tools. `http://nsat.sourceforge.net/`, 2003. Last Visited: 9 November 2004.

[18] C. X. Naval Surface Warfare Center, Dahlgren Division. Shadow indications technical analysis: Coordinated attacks and probes. `http://www.nswc.navy.mil/ISSEC/CID/co-ordinated_analysis.txt`, 1998. Last visited: 23 March 2002.

[19] V. Paxson. Private communication between Carrie Gates and Vern Paxson, February, 2006.

[20] S. Robertson, E. V. Siegel, M. Miller, and S. J. Stolfo. Surveillance detection in high bandwidth environments. In *Proceedings of the 2003 DARPA DISCEX III Conference*, pages 130 – 139, Washington, DC, April 2003.

[21] S. Staniford. Intrusion correlation engine. `http://www.silicondefense.com/research/ACM-tutorial-11-4-01.ppt`, 2001. Powerpoint presentation on The Intrusion Correlation Engine from an ACM conference. Last visited: 13 January 2003.

[22] S. Staniford, J. Hoagland, and J. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1):105 – 136, 2002.

[23] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149 – 167, San Francisco, California, USA, August 2002. USENIX Association.

[24] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – a graph based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, pages 361 – 370, Baltimore, MD, USA, October 1996.

[25] W. W. Streilein, R. K. Cunningham, and S. E. Webster. Improved detection of low-profile probe and denial-of-service attacks. In *Proceedings of the 2001 Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, June 2001.

[26] University of Southern California Information Sciences Institute. The DETER testbed: Overview. `http://www.isi.edu/deter/docs/testbed.overview.pdf`, 2004. Last Visited: 9 November 2004.

[27] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255 – 270, Boston, MA, USA, December 2002. USENIX Association.

[28] D. Whyte, P. van Oorschot, and E. Kramakis. Tracking darkports for network defense. In *Proceedings of the 23rd Annual Computer Security Applications Conference*, Miami Beach, Florida, December 2007.

[29] V. Yegneswaran, P. Barford, and J. Ullrich. Internet intrusions: Global characteristics and prevalence. In *Proceedings of the 2003 ACM Joint International Conference on Measurement and Modeling of Computer Systems*, pages 138 – 147, San Diego, California, USA, June 2003.