A Case Study in Testing a Network Security Algorithm

Dr. Carrie E. Gates
CA Labs, CA
Islandia, NY 11749
carrie.gates@ca.com

ABSTRACT

Several difficulties arise when testing network security algorithms. First, using network data captured at a router does not guarantee that any instances of the security event of interest will be captured. Similarly, if the event of interest is not detected, this does not guarantee that it does not exist in the captured data. Further, such network data is often not publicly available, making comparisons with other detectors difficult. On the other extreme, purely simulated data can be made publicly available and can provide guarantees that the event of interest exists in the data set. However, simulated data often has unintended artifacts and may also incorporate the biases of the particular simulator. In this paper I describe an emulation approach that takes advantage of captured data while using the DETER network to generate realistic traffic for the event of interest. The problem domain was described in terms of seven variables, where the DETER network provided a flexible medium for examining the complete problem domain. The results of a set of experiments using this approach are provided, along with regression equations that describe the expected true and false positive rates.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: General—*Security and protection*

Keywords

Network Security, Testing

1. INTRODUCTION

Testing software properly in order to demonstrate that it performs as expected is a very time-consuming and difficult process. However, in general the inputs, outputs, and expected behaviour are known before the software is deployed. In the case of software intended to perform security functions, this may not be the case. This is especially true of security algorithms that are expected to detect security events on a host or network, such as intrusion or anomaly detection systems, worm detection algorithms, and behavioural analysis systems.

Conference name: Tridentcom 2008, March 18 - 20, 2008,
Innsbruck, Austria.

Testing security detection algorithms, particularly ones that use network data as input, is complicated because the input can be extremely variable. Network traffic varies considerably by time of day, time of year, type of network (*e.g.*, university, government, corporation), size of network, and enforced securty policies. Further, network traffic is continually evolving as new applications are developed. Even assuming no malicious data is present in the network and testing the algorithm purely for false positives, it is not possible to test against every possible combination of factors given the variability in possible deployment environments and of network traffic itself.

This situation is further complicated once one includes the security events of interest, as the security events themselves can also be extremely variable. This variability is expressed in the event (for example, is it a slow-scanning or fast-propagating worm?) and in the resulting network traffic. For some detectors (such as intrusion or anomaly detection systems), the number of different types of events alone that the system should be tested against in order to determine its detection capabilities is prohibitive.

The two approaches that are most commonly used for testing network security algorithms are simulation (particularly the use of the Lincoln Labs [8] dataset) and network traces [2]. Simulation has the advantage that, if the variables required can be identified, the variability can be included in the dataset and the algorithm tested against it. However, a simulation may not necessarily be representative of actual data and may contain unintended biases. While network traces do not have this issue, it cannot be guaranteed that they contain the events of interest, nor is the location of such events known (thus making false negatives difficult or impossible to determine). While this can potentially be resolved through the use of a red team to perform security experiments against the monitored network, this is generally not a viable option due to both resource requirements and legal limitations.

In this paper I present a case study in the use of an emulation approach to solving the problem. Background data is obtained using network traces. The security events of interest are performed on the DETER network, and the network traffic captured. These two datasets are then combined to create a single set that contains known (labeled) security events. I first describe the usual approaches to testing network security algorithms in Section 2, along with the limitations of both appraoches. I then descibe the emulation approach that I use for testing network security event. I present a case study using this approach on the detection of co-ordinated scans, which I describe in detail in Subsection 4.1. A brief description of the detection algorithm tested is presented in Subsection

4.2. Details on how the testing was performed and the DETER network are provided in Subsection 4.3, demonstrating the emulation approach. The testing results are outlined in Subsection 4.4, indicating that the emulation appraoch is valid. Some concluding remarks are provided in Section 5.

2. EVALUATION METHODOLOGIES

Traditional intrusion detection research has not focused on developing appropriate evaluation methodologies. The result is that the capability of detection algorithms has tended to be presented in terms of either the Lincoln Labs dataset [8] or the use of proprietary network traces accessible only to the authors of a particular system. As noted by Athanasiades *et al.* [2], the result is that the "ad-hoc methodology that is prevalent in today's testing and evaluation of network intrusion detection algorithms and systems makes it difficult to compare different algorithms and approaches." The two appraoches can be summed up as using either real data or simulation, and both of these approaches are discussed in more detail below, along with a third option — emulation — that is less widely used.

2.1 Real Data

The use of real data consists of the collection of network traces from some network. Due to privacy and anonymization issues, these traces are often not made available to the general public, but rather remain proprietary to those authors who have negotiated trust relationships with the custodians of the data. Many researchers are not able to even develop these trust relationships due to legal issues surrounding data access.

The benefits of using such a data set is that there is no bias in the data, which can occur in other environments, as discussed below. Thus one can have some confidence in the predictive capabilities of their algorithm in a deployment setting.

However, there are several limitations to using real data. First, such data is often limited due to privacy concerns, and consists at most of network flows or header information, and rarely of full packets. Additionally, such data sets are not labeled. That is, one does not know what attacks are in the data, nor where those attacks occur. There is no control over what attacks are present, and so some portions of a detector may remain untested. As the presence of attacks are not known *a priori*, the false negative rate cannot be established as not detecting a particular attack might occur because the attack was not present, or it might occur because the detector failed to find the attack(s). Additional problems caused by the lack of labeling can be found when one is dealing with anomaly detectors, which often require either labeled or clean training data in order to detect later anomalies or attacks.

Another difficulty from using actual network traces is that the results obtained are not representative of other environments. This was observed by Maxion and Tan [10] in the context of anomaly detection. They noted that, even when a detector has been retrained for a new environment, the same detector will not achieve the same performance given this different environment. They go on to state that "This is in absolute contrast to current practice" which assumes that results obtained on one network are transferable to other networks. This indicates that the results obtained for two different detectors can not be compared unless both detectors were tested using the same network traces. However, when real network traces are used, this is often not possible as the traces are largely proprietary in nature, as discussed above.

2.2 Simulation

A second option when testing security algorithms is simulation. In this case, network traffic, including the malicious behaviour to be detected, is simulated and then the algorithm is tested against this traffic.

There are three advantages to simulating network traffic. The first is that any simulated data can be released for public use. Thus any two systems can be tested using the same environment and then their performance can be compared objectively. Secondly, simulating data provides a large amount of control over the environment. Thus guarantees can be made, such as that the traffic contains particular attacks. Further, as all of the attacks are known, they can be labeled, and thus accurate true and false positive and negative rates can be calculated. The third advantage is that there are no privacy issues or limitations on the data, and thus payload can also be simulated and used for those algorithms that require it.

However, simulated network traffic has distinct disadvantages. The first is that the simulation will not necessarily be representative of actual network traffic patterns. As noted by Paxson and Floyd [13], network traffic is difficult to simulate due to its self-similar nature. Simulating payload may be even more difficult. Further, biases can unintentionally be added to the simulation. For example, if one is simulating a particular attack and has assumptions on how that attack will appear, then these assumptions, which may not be correct, will be incorporated into the simulation.

One (largely) simulated data set was developed by MIT's Lincoln Labs in 1998 [8], and is still widely used. Ths data set was created with the express purpose of comparing different anomaly detection approaches. It consisted of simulated legitimate traffic that was modeled from real network traces captured at an Air Force base. Attacks were then performed in an isolated environment and the traffic from these were injected into the simulated traces. This data set was later critiqued by McHugh [11] who found that, in general, the traffic was too well behaved. For example, there was no Internet background radiation [12] present. Further, the data rates used were not indicative of a network of the size being used in the simulation, but there were much fewer traffic flows than would be expected. Additionally, the security events injected into the simulated traffic did not represent the actual frequency of attacks. For example, there were only a few scans present in the data, yet scanning is a fairly common activity. The Lincoln Labs data set was later analyzed further by Mahoney and Chan [9] who compared this data set with a real network trace. They found that the Lincoln Labs data exhibited regularities that were not present in real data. For example, TCP SYN packets in the simulated data always had exactly four option bytes, whereas in the network traces this value ranged from zero to 28. They concluded that such regularities could affect the training or testing of an anomaly-based security detector.

3. EMULATION APPROACH

Simulation has a definite advantage in that it provides labeled data and that the characteristics of the attack or network can be controlled. I aim to maintain these advantages while using real network traces. In order to achieve this I perform the attacks on a network testbed, capturing the resulting traffic for analysis. By doing this I can control exactly the characteristics of the attack. This step is the same as that performed by Lincoln Labs in generating their attack data [8], however they generated their attacks considering only a single network. Given the flexibility of network testbeds, multiple network configurations can be examined (based on the availability

of network traces, which is discussed further below). Further, any characteristics that can be controlled in an attack (such as the number of attacking hosts or the attacking algorithm) can be varied so that a detector can be tested in a systematic manner.

To maintain the "realness" of actual network traces, actual network traces are used for background data. Ideally, traces are captured from multiple networks that are controlled for the characteristics of interest. For example, traffic might be collected from both /24 and /16 networks. The attack data that was captured on the testbed is then injected into the network traces, resulting in a labeled data set containing actual network traffic and actual attack data, yet providing attacks that meet the desired characteristics to allow for a systematic testing of a security detector. It is important that traffic at least be collected over a long period of time, if multiple networks are not available, so that the network traffic used for background noise can be varied across tests. Otherwise a detector has only been tested in a single environment, and no statements can be made as to its generalizability to other networks that potentially have different noise characteristics that might affect its detection accuracy.

There are three limitations to this emulation approach. The first is that it is very time-consuming, requiring attention to detail and the running of multiple experiments. However, this approach is still not as time consuming as generating a reasonable simulation, nor as locating and labeling attacks in network traces. The second limitation is that an assumption is made that the network traces used for background noise do not contain any instances of the security event of interest. If such events are present, then either the detection rate will be incorrect (if the event was not detected when present) or the false positive rate will be incorrect (if the event was detected but considered as not being a true event). Finally, this approach may not work with all detectors. For example, intrusion detection systems aim to detect multiple types of events, which may result in too many different security events to fully examine its detectio capability. Rather, this approach is geared to testing detectors that focus on a single type of event.

A potential fourth limitation si also present and is related to the network traces used for background traffic. An assumption is made that the available network traces contain traffic at the granularity required for the security detector. In practice, this may not be true. For example, it is very difficult to acquire full-packet network traces due to privacy, and potentially legal, concerns. However some detectors, such as signature-based intrusion detection systems Snort and Bro, operate at this level. Thus it might not be possible to use this testing approach if the appropriate network traces are not available.

In the next section I provide a case study demonstrating the usefulness of an emulation approach to testing a network security detector. In this case, I was able to systematically examine a detector's performance by controlling the relevant attack and network variables, generating a model of the detector's performance that should indicate its accuracy given a new attack or network.

4. CASE STUDY

A case study is presented in this section that indicates how the emulation approach can be used to test a security detector. The problem domain for which the detector was developed — that of detecting co-ordinated scans — is presented first. This is followed by a brief description of the detection algorithm, presented in order to demonstrate the variables that might impact on its detection capability. The emulation data generated, based on a combination of attacks generated on a testbed and actual network traces, is described with a focus on how the variables that might affect the detector were determined and varied. The last subsection provides a brief summary of the results obtained, demonstrating that the capabilities of a security detector can be modeled when examined in a systematic fashion, and that this model implies how the detector will perform in other circumstances.

4.1 Problem Domain

The hypothesis that was tested was that it was possible to detect the presence of co-ordinated scanning activity.

Scanning consists of a series of probes against a target system or network, where a probe is a reconnaissance activity aimed at a single target. A target here could be a single host, or a particular service on a particular host. Reconnaissance activity consists of the attempt to determine if the target exists. For example, someone could send an ICMP ping to a particular host. In this case, the ping is the reconnaissance activity and the host is the target. Alternatively, someone could send a single SYN packet to port 80 on a particular host. Here the reconnaissance activity is the attempt to determine if a web server is present on that host. When an adversary probes multiple targets, this is considered to be a scan.

Staniford *et al.* [15, 14] defined four types of scans: vertical, horizontal, strobe and block. A vertical scan consists of scanning multiple ports on a single host, where the aim is to find a weakness in that particular host. A horizontal scan was a scan against a single service across multiple hosts, such as scanning a network for the presence of web servers. Scanning multiple services across multiple hosts, such as scanning for hosts running both http and ssl services, was called a strobe scan, while a block scan consisted of scanning all ports on all hosts.

Staniford *et al.* [15, 14] went on to define *stealthy scans* as scans that were designed to evade detection. One approach to doing this was to scan slowly enough that the detection system did not correlate the activity. A second approach, distributed scanning, consisted of using multiple hosts to each scan a portion of the target space.

In this paper I use the term *co-ordinated scanning* to represent the co-ordinated use of multiple sources to scan a target, where each source scans a portion of the target space. A *distributed scan*, by comparison, consists of multiple hosts scanning a target space, but where there is no co-ordination between the hosts. An example of a distributed scan is the case of Stumbler, described by Intrusec as a passive peer-to-peer distibured port scanner [7].

4.2 Detection Algorithm

The algorithm developed to detect the presence of co-ordinated scans focused on scans performed against a network — either horizontal or strobe scans. It made the assumption that a scan detection system was already in place to monitor the target network, and that the scans performed as part of the co-ordinated activity are detected by that system. Thus the input to the algorithm is the set of detected scans over some period of time. In particular, the algorithm requires for each scan the source of the scan and each scan target (IP/port pair).

Given this input, the algorithm was inspired by the set covering problem, where the goal is to determine if there is some subset of

the scans that, when combined, cover the same port or ports across a contiguous portion of the network. The set covering problem consists of determining the minimum number of sets required to cover an entire space. In contrast, this algorithm focused finding a set of sets (scans) such that the coverage of the space (network) was maximized while the overlap between sets (scans) was minimized. This algorithm is described in detail by Gates [4].

There are three variables that are controlled by the administrator who is using this algorithm: scan window, coverage and overlap. The scan window refers to the number of scans that are input for analysis to determine if any co-ordinated scans are present. Coverage refers to the percentage of the target network that the adversary has scanned. The administrator can specify if he only wants to detect those adversaries who have scanned the entire network, or some minimum portion of it. Overlap refers to the amount of overlap between the scans. This variable was added to ensure that the adversary could not avoid detection by having sources that scan overlapping targets. The administrator can set the maximum amount of overlap he expects an adversary to use.

4.3 Generating Test Data Sets

The DETER network[1], which is based on Emulab software[2] [16], is a testbed that is focused on supporting network security research. Figure 1 demonstrates a typical network setup for testing the security algorithm described in Subsection 4.2. In this case there are five agents (scanning clients), one handler (who controls the five agents), two hosts emulating a /16 network, and a monitoring host positioned between the scanning hosts and the scanned network. The monitoring host was running tcpdump to capture all of the network traffic generated by the scanning hosts.

The algorithm being tested could be reasonably varied in seven different dimensions. The first three are coverage, overlap and scan window, which are controllable by the security administrator and discussed in Subsection 4.2. The adversary can also choose particular scan characteristics that might affect the detection capability of the algorithm, such as the number of sources involved in the co-ordinated scan, the number of ports being scanned, and the scanning algorithm being used. The scanning algorithm refers to the algorithm used to distribute the scan destinations amongst the scanning hosts (two algorithms were available for these tests). The last variable is the size of the network being scanned, as the detection capability might be different between, for example, a /24 network and a /16 network. In this case, the network size was treated as a discrete value, with only /24 and /16 networks being considered. This limitation was placed on th data due to the characteristics of the available network traces.

The extreme values for each of the seven variables were determined (*e.g.*, for the number of sources the values ranged from two to 100). Using just the two extreme values for each variable resulted in 128 possible combinations. However, one of the scanning algorithms did not have the capability to generate overlap between the scanning sources, and so the actual final value was 96 combinations. Thus experiments were performed using these 96 combinations. In addition, 20 combinations were randomly chosen from within these ranges where possible. (Note that the choice of scanning algorithm was discrete, with only two choices, as was the choice of network size.) Network traffic was collected from running these combinations. The final result was 116 different data sets consisting of scan traffic (collected using tcpdump), where the scans met the specified conditions. Thus the scans used for testing the detector were generated in a systematic manner and generally represent the more extreme cases, thus testing the limitations of the detector.

[1] `http://www.isi.edu/deter/`
[2] `http://www.emulab.net/`

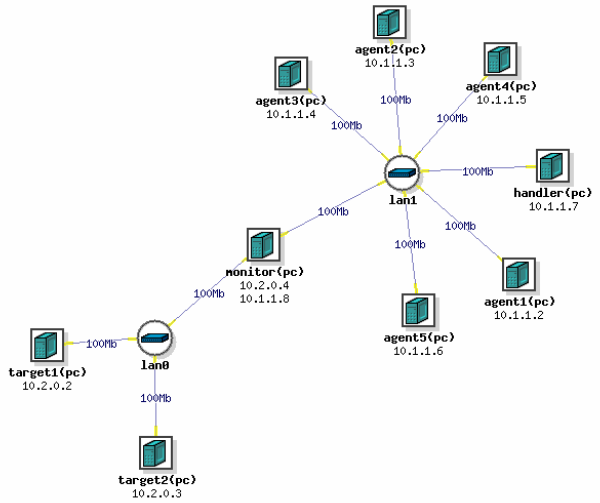

Figure 1: Co-ordinated port scan DETER set up with 5 agents, 1 handler and a /16 subnet.

Network traffic flows were collected from several live networks using the SiLK [5] suite of software. More specifically, traffic was collected from four /16 networks during a time period of approximately two weeks, and traffic was collected from another four /24 networks during a one month time period. Summary scan information was extracted for each network using the scan detection approach outlined by Gates *et al.* [6]. An assumption was made that no co-ordinated scans were present in this background traffic. The resulting traffic capture was used as background traffic for the 116 scan combinations identified earlier. Care was taken to ensure that the background traffic selected for each of the scans was different for each data set. This was done by first randomly choosing the traffic collected from one of four different networks (of the appropriate size), and then by randomly choosing a starting location within the background traffic. The number of noise scans required (varied from 100 to 1000 for this particular set of tests) were then collected from that point forward into the scan data.

4.4 Testing Results

I analysed the results from our experiments using detection rate and false positive rate as described by Axelsson [3]. Here the detection rate is the probability of generating an alert, A, given that there was an event, I, $P(A|I)$. The false positive rate is the probability that there was an alert given that there was no event, $P(A|\neg I)$.

The detection algorithm was run on each of the 116 data sets described in Section 4.3 with the detection rates and false positive

rates calculated for each data set. The detection rate was defined as the number of sources correctly identified as being part of a co-ordinated scan, while the false positive rate was defined as the number of sources that were incorrectly identified as being part of a co-ordinated scan. The number of co-ordinated scans detected were not taken into account. Thus the detection rate might be 100% with a 0% false positive rate for a particular data set, yet the single co-ordinated scan was actually detected as multiple co-ordinated scans. Cases such as this are not analyzed separately in this paper.

Both the detection rate and the false positive rate were modeled using regression equations. In the case of the detection rate, the data demonstrated a nearly bimodal distribution, with 96 of the 116 co-ordinated scans either being detected perfectly or not at all. The remaining 20 were determined to be detected if at least 50% of the scanning sources were correctly detected. Due to the bimodal distribution, a logistic regression was used, resulting in a model of the detection rate that returned a probability that scans with a particular characteristic will be detected:

$$\hat{P}(\text{co-ordinated scan is detected}) = \frac{e^{\hat{y}}}{1 + e^{\hat{y}}} \quad (1)$$

where $\hat{y}$ is a weighted summation of the seven variables identified in Subsection 4.3. The scanning algorithms were mapped to either zero or one, while the network size was represented by the number of hosts in the subnet (256 or 65536). The number of variables was reduced using the Akaike Information Criterion [1], indicating those variables that contributed most to the detection rate of the algorithm. The sign on the weight for each variable indicates if an increase in the value of the variable causes an increase or decrease in the detection rate.

The false positive rate was similarly modeled, however it used a linear regression rather than a logistic regression (as the false positives did not demonstrate a bimodal distribution). The actual results and modeling approach are described in more detail by Gates [4].

This subsection demonstrates that testing an algorithm in a systematic manner can result in a model of how that detector performs. Gates [4] demonstrates in detail how well the model performs at predicting the detector's performance in previously unseen circumstances through further testing, and further demonstrates how this approach can be used to compare detectors. A complete description of these results is outside the scope of this paper, but is briefly provided here to demonstrate the power of using an emulation approach to testing network security detectors.

5. CONCLUSIONS

In this paper I described the two appraoches currently used in testing network security detectors: simulation and real network traces. Simulation provides the most control over the test environment, however it is difficult to simulate network traffic [13] and previous attempts [8] have had several flaws [11, 9]. Network traces have the advantage of containing actual data (and so eliminate the problems associated with simulation), however the data is not labeled, and so it is not possible to determine the detection rate as the number of events in the data are not known. Additionally, the events of interest may not even be present in the captured traffic. Further, testing in a single environment, be it simulated or real, does not indicate how well an algorithm will perform given a different environment.

I presented an emulation approach to testing network security detectors that is based on a combination of actual network traces and attacks captured from a network testbed. This combined appraoch provides both realistic background traffic and realistic attacks. It addresses the disadvantage of simulations not having realistic data, as well as the disadvantage of network traces not containing labeled attacks. However, this approach is particularly suited to testing detectors designed for a single event, rather than, for example, generic intrusion detectors.

This approach was demonstrated by testing a co-ordinated scan detector using actual network traces and co-ordinated scans captured from the DETER network security testbed. The results were analysed, demonstrating that they could be modeled using regression equations. The use of the emulation appraoch allows the researcher to test their algorithm in multiple environments in a controlled and systematic fashion. Further, the use of regression equations provides the potential to extrapolate from the emulation to previously unexplored environments, indicating how the detector might perform given a new network or attack characteristic.

6. 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] N. Athanasiades, R. Abler, J. Levine, H. Own, and G. Riley. Intrusion detection testing and benchmarking methodologies. In *Proceedings of First IEEE International Workshop on Information Assurance*, pages 63 – 72, Darmstadt, Germany, March 2003.

[3] S. Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security*, 3(3):186 – 205, 2000.

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

[5] C. Gates, M. Collins, M. Duggan, A. Kompanek, and M. Thomas. More NetFlow tools: For performance and security. In *Proceedings of the 18th Large Installation Systems Administration Conference (LISA 2004)*, pages 121–132, Atlanta, Georgia, USA, November 2004.

[6] 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*, pages 402 – 407, Pula-Cagliari, Sardinia, Italy, June 2006.

[7] Intrusec. Intrusec alert: 55808 trojan analysis. `http://www.intrusec.com/55808.html`, 2003. Last visited: 1 July 2003.

[8] R. P. Lippmann et al. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition*, volume 2, pages 12 – 26, 2000.

[9] M. V. Mahoney and P. K. Chan. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*, pages 220 – 237, Pittsburgh, PA, USA, September 2003.

[10] R. A. Maxion and K. M. Tan. Benchmarking anomaly-based detection systems. In *Proceedings of 2000 International Conference on Dependable Systems and Networks*, pages 623 – 630, June 2000.

[11] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, 3(4):262 – 294, 2000.

[12] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of internet background radiation. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 27 – 40, Taormina, Sicily, Italy, October 2004.

[13] V. Paxson and S. Floyd. Why we don't know how to simulate the internet. In *Proceedings of the 29th conference on Winter simulation*, pages 1037–1044, Wisconsin, 1997. ACM Press.

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

[15] S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical automated detection of stealthy portscans. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, November 2000.

[16] 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.