

Detecting Low-Profile Probes and Novel Denial-of-Service Attacks

Raj Basu, Robert K. Cunningham, *Senior Member, IEEE*, Seth E. Webster, Richard P. Lippmann, *Senior Member, IEEE*

Abstract—Attackers use probing attacks to discover host addresses and services available on each host. Once this information is known, an attacker can then issue a denial-of-service attack against the network, a host, or a service provided by a host. These attacks prevent access to the attacked part of the network. Until recently, only simple, easily defeated mechanisms were used for detecting probe attacks. Attackers defeat these mechanisms by creating stealthy low-profile attacks that include only a few, carefully crafted packets sent over an extended period of time. Furthermore, most mechanisms do not allow intrusion analysts to trade off detection rates for false alarm rates. We present an approach to detect stealthy attacks, an architecture for achieving real-time detections with a confidence measure, and the results of evaluating the system. Since the system outputs confidence values, an analyst can trade false alarm rate against detection rate.

Index Terms—Intrusion Detection, Probes, Scans, Information Assurance, Security

I. INTRODUCTION

Heavy reliance on the Internet and worldwide connectivity has greatly increased the potential damage that can be inflicted by remote attacks launched over the Internet. Intrusion detection systems have been developed to provide alerts for attacks that occur despite preventative measures. A review of the many alternative approaches to intrusion detection is available elsewhere [1], and a summary of approaches to port-scan detection is also available [2]. The majority of commercial and research intrusion detection systems deployed today search for attack signatures in network traffic. Signatures are often keywords in the data part of network packets or characteristics of packet headers. Port-scanning signatures are only slightly more complex. For example, a port-scan detector might look for packets to more than N different port/IP combinations within M seconds from a single source [2]. This approach is popular because one system can monitor traffic to many workstations and the computation required to reconstruct network sessions or analyze packet headers to search for signatures is not excessive. This approach to intrusion detection, however, has many weaknesses. Stealthy probe and denial-of-service

(DoS) attacks may be missed and novel new attacks that have not been seen before are not detected. This paper describes an intrusion detection system designed to remedy these weaknesses. It is designed specifically to detect novel attacks and to detect slow, stealthy probe and DoS attacks. The remainder of this paper is organized as follows: first we describe the attack database used to evaluate the new system, then we describe an off-line version of the system and the detection and false alarm rates for the off-line system. We next describe a more-recent real-time version of the system. The paper finishes by describing current research directions.

II. EVALUATION CORPUS

This research was made possible by the availability of a large-scale realistic intrusion detection corpus created under DARPA funding in 1998 and 1999 [3] [4]. In 1999, three weeks of network traffic were generated that was similar to the type of traffic observed flowing between U.S. government sites and the Internet. Traffic and attacks were generated on a network, which simulated 1000's of Unix hosts and 100's of users. Attacks were launched from outside the simulated site against Windows NT, Linux, SunOS, and Solaris UNIX victim hosts on the inside. Sniffing data containing all bytes transmitted to and from this simulated site were used for system development and testing along with attack labels and timing information. This database contains two weeks of training data with background traffic and attacks and one week of test data. Only training data were used for system development. Formal evaluation guidelines [3,4] were followed for testing, and only one pass through the test data was performed for the final evaluation.

Test data in this database contains 114 old attacks previously used and described in 1998 executed in the clear, 62 novel new attacks developed for 1999 and 35 stealthy versions of attacks used in 1998. Details on these attacks are available in [4,5,6]. This study used the probe, DoS and novel new attacks. Probe attacks automatically scan a network of computers or a DNS server to find valid IP addresses (*ipsweep*, *lsdomain*, *mscan*), active ports (*portsweep*, *mscan*), host operating system types (*queso*, *mscan*), and known vulnerabilities (*satan*). All of these probes except one (*satan*) are either new in 1999 (e.g. *ntinfoSCAN*, *QueSO*, *illegalsniffer*) or are stealthy versions of 1999 probes (e.g. *portsweep*, *ipsweep*). Probes are

This work was sponsored by the Department of Defense under Air Force contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Air Force.

Robert K. Cunningham, Seth E. Webster and Richard P. Lippmann are with MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02420.

considered stealthy if they issue ten or fewer connections or packets or if they wait longer than 59 seconds between successive network transmissions. Stealthy probes are similar to clear probes because they gather similar information concerning IP addresses, vulnerable ports, and operating system types. They differ because this information is gathered at a slower rate and because less, but more focused, information is gathered from each attack instance. For example, stealthy port sweeps are slow and focus only on ports with known vulnerabilities. Denial-of-service (DoS) attacks are designed to disrupt a host or network service. New 1999 DoS attacks crash the Solaris operating system (selfping), actively terminate all TCP connections to a specific host (tcpreset), corrupt the ARP cache on a victim host (arppoisson), crash the Microsoft Windows/NT web server(crashiis), and crash Windows/NT (dosnuke).

III. SYSTEM DESIGN AND DEVELOPMENT

Evidence for probes and network DoS attacks can be found in network data packets. Sometimes a single packet can indicate an attack because of its header or destination. For example, QueSO [8] sends packets constructed with invalid combinations of header bits to identify the operating system. Although the tool must send several packets to each host operating system that is being identified, some header bit combinations should never occur in normal traffic. Presence of these packets indicates that a probe has occurred. Other probing tools, in an effort to determine which services are available, send connect requests to non-existent machines or closed ports. If one knows how a network and its hosts are being used then these probes can be detected.

Sometimes multiple packets must be examined to detect one of these attacks. For example, most services have an expected volume of data that is passed between the client and the server. If too much or too little data is passed, then that may indicate that an attacker is misusing a service.

To record these and other features, we used our network data parser tool [9]. This tool uses the libpcap libraries to acquire network data and can produce either session transcripts or a summary of the packets in a session. The summary includes the duration of the connection, the MAC and IP addresses of the source and destination, the port addresses, the number of fragmented packets in each direction, the number of overlapping fragments, the number of enclosed fragments, the total number of bytes and the number of missing bytes, the number of TCP header flags seen in each direction, and any anomalies found in the packets (such as strange combinations of TCP flags or fragments reassembling into too large a packet).

For TCP connections, a session includes all packets from the initiation of the connection to the final packet. Although neither UDP nor ICMP are session-based protocols, we have found that it is useful to construct a session for these. We define UDP and ICMP sessions to be all transmissions of the same type between the same

host pair within a set time window. (For the system described here, a 10 second window was used for both UDP and ICMP.) This grouping reduced the amount of data written when the traffic rate was high, such as when a flooding attack is present. Similar summary statistics are created for these sessions.

Development proceeded in two stages. In the first stage, we considered how to select features and construct a system that could detect attacks using previously recorded network data. In the second stage, we considered how to build a system that could detect attacks as they occur.

A. Off-Line System Development and Evaluation

Our off-line system included four major components: a network data parser, a pair of feature extractors, a set of classifiers, and a post processor.

During system training, the feature extractor reformatted the data parser output to support detecting or identifying an attack amongst other network packets. Each of the feature lists were then fed into a suite of classifiers. The classifier used was a multi-layer perceptron (MLP) with one hidden layer and four hidden nodes, trained with stochastic weight updating [10]. The MLP output is the posterior probability that an attack took place in each connection. False alarms were reduced by an expert system post-processor that also reformatted the data for human interpretation.

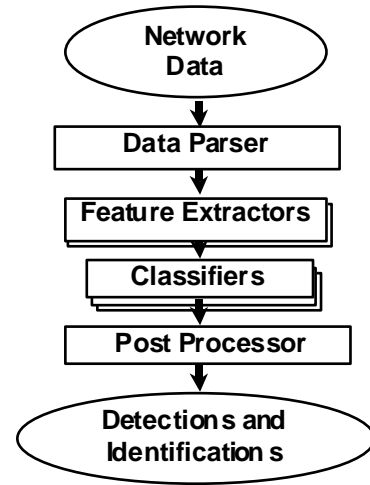


Fig. 1 Offline System Architecture.

The focus of the off-line research was selecting a good set of features to detect and separately identify attacks. The detection feature set consisted of packet header features, traffic level features, windowed-statistics features, and general attack features. Packet header features included unusual or unspecified TCP flag combinations. For example, the combination of SYN and FIN flags was often indicative of a probe. Traffic level features described the quantity and type of data being sent during a certain connection. These included the number of data bytes sent and the number of enclosed or overlapping fragments in a specific service's traffic. Large changes in these statistics could be indicative of flooding attacks. Similarly, uncharacteristically high levels of fragmentation

were caused by certain denial-of-service attacks. We use two different types of windowed-statistic features to detect denial-of-service attacks that exhausted network and host resources. The first is a time-based window that counted the number of connections to each machine in a fixed (usually short) period. This feature set was intended to detect denial-of-service attacks and fast port scans. Both of these attacks make a large number of connections to ports of a single IP address in a short period. This is not present in normal traffic and is easy to detect. The second set of windowed-statistic features are windowed based on a fixed number of previous connections. These tracked statistics across the last one hundred connections originating from or destined for a specific IP address. Features gathered within connection-based windows included the different number of services and IP addresses connected to during the course of the window. During long, slow probes there are an unusual number of connections to various ports on a single IP address or an unusual number of connections to various IP addresses from a single IP address.

Finally, general attack signatures included flags that indicate abnormalities characteristic of known attacks, such as a malformed "GET" request in the HTTP protocol.

The second feature set is used to *identify* detected attacks. Since identification requires labeling attacks previously encountered, this feature set was tailored to attacks present in the training data. These features included flags that were set when a specific attack pattern was recognized.

Once feature extraction was complete, classifiers were trained on labeled data provided by Lincoln Laboratory for both the 1998 and the 1999 evaluation. The first two classifiers each detected an entire attack class. One detected probes and one detected denial-of-service attacks. The next group of classifiers was trained so that each was designed to detect only one attack type. A winner-take-all competition between classifiers was used to associate the highest identification score to a session. If the classifier generating the highest score was an identification classifier, then the corresponding identity was associated with the feature set. If a detection classifier produced the highest score, then the attack was reported as detected (and of a particular class) but the specific attack was not identified.

The offline system was evaluated on the 1999 DARPA Intrusion Detection testing data, following the rules of the evaluation. Unfortunately, these data were produced by members of our group, so this is not a true blind evaluation as it was for DARPA researchers.

Fig. 2 depicts the Detection False Alarm (DFA) plot for our system [6]. For every possible operating point of this system it shows both the detection rate and the false alarm rate. Where near-vertical runs of operating points appear, a user would select the operating point with the highest detection rate at the given false alarm rate. For example, at a false alarm rate of 0.1 per day, the detection rate for

probes can be set to roughly 35%, and this detection rate increases to roughly 72% at a false alarm rate at and above 0.3 per day. Over the range of 0.1-1.4 false alarms per day (for these data), the overall system detection rate can be made to vary from 20 to 65%. This represents reasonably good control of the tradeoff, and compares favorably to other research systems. For false alarm rates above one per day, detection performance is roughly 70% for both probe and DoS attacks. The low false alarm rate is due in part to the expert system post processor. Each connection that produced an identification alert was compared against a set of rules for the labeled attack type. If the connection did not satisfy the rules provided, the score was decreased by a constant factor. By simply decreasing the score instead of setting the score to zero, similar but not identical attacks could still be detected. This helped some with detecting new DoS attacks, although not as well as was hoped. At about 1 false alarm per day, the system only detected 13% of new attacks. This was good, but not as good as the best of the evaluated research systems, which detected more than 20% of the new attacks [7].

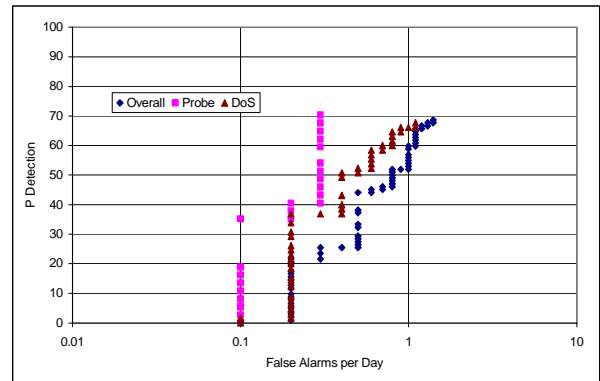


Fig. 2. System performance: overall, on just probes, and on just DoS attacks.

The probe detection system was particularly good at detecting stealthy attacks—the technique of modeling past system use on per-connection basis allowed the system to detect attacks scanning just a few hosts and ports over many hours almost as well as attacks that addressed many ports and hosts in a short period of time. This is demonstrated in Fig. 3. This figure shows that the system performs almost as well detecting the “stealthy” attacks as it does detecting the old, clear attacks. At a false alarm rate of roughly 0.3 per day, the detection accuracy was roughly 80% for old-clear attacks and 72% for stealthy attacks. The system also does a good job of detecting new attacks, as anomalous connections or partial connections to a port and host, regardless of the type of connection, still initiate a response. At a false alarm rate of roughly 0.3 per day, the detection accuracy for new attacks was roughly 55%. This detection performance is better than other evaluated systems [7].

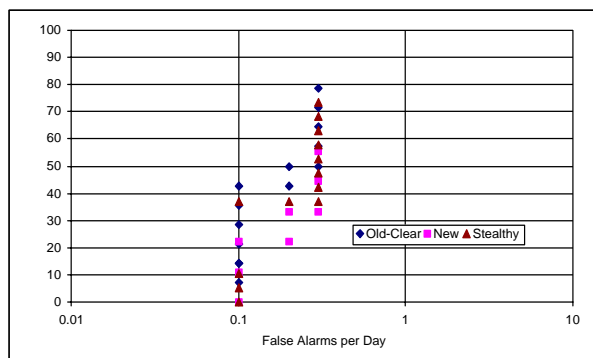


Fig. 3. Probe detection sub-system on various classes of attacks.

One area where the system didn't perform as desired was in the detection of attacks that were not present in training data and varied greatly from any attack present in the training data (novel attacks). Deficiencies in the feature set and the expert system led to this weakness. The second stage of development focused on improving this aspect of the system and building a real-time system.

B. Real-Time System Development and Evaluation

Whereas the off-line system could collect all of the data it needed on a session, a real-time system is most useful when it can issue an alert for an on-going attack before the attack is complete. To build such a system, TCP connections were evaluated twice--upon the completion of the three-way handshake and upon termination of the session. In addition, to support detecting attacks against different network configurations, a separate network profile file was introduced so that the same code base could be used for different networks. The real-time system architecture is depicted in Fig. 4.

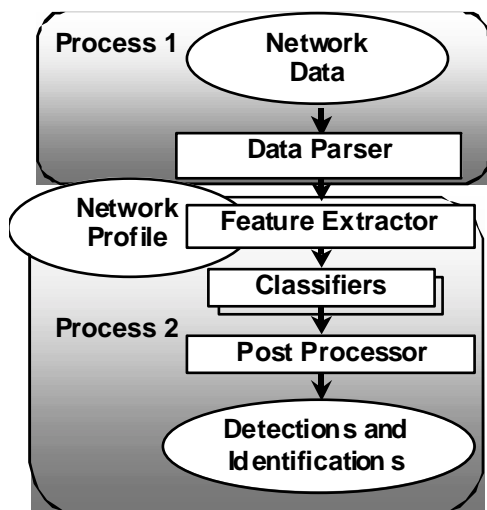


Fig. 4 Real-time System Architecture.

Other than the network profile, the real-time system architecture is quite similar to the off-line system. The real-time system consists of several components: a network data parser, a feature extractor, a classifier, and an output processor. To ensure timely servicing of incoming packets, the lightweight network data parser is run as a separate process from the more computationally expensive

detection process.

To exchange information between the two processes, the network data parser used in the offline system placed tokens into a shared memory buffer. Data from the shared memory buffer are then reformatted into computationally efficient structures used by the feature extractor. To simplify computing probing features, data are stored in two hashes: the first hashes the source IP address, while the second hashes the destination IP address. Each include buckets that list all ports addressed within a given window type. This supports easy detection of all incoming host and service probes from a single machine to multiple addresses. It also supports detection of probes that start at multiple hosts and attempt to scan for services on a single host. The data structures proved efficient but suffer from two weaknesses: first, if the details of the hashing function are known to an attacker, the system can be forced into worst-case performance. Second, the data structure obscures IP scanning that comes from multiple IP addresses. We are working on fixes to these two problems.

The feature extractor reformats features for input to a classifier, again a single hidden layer perceptron. The classifier provides the posterior probability that the given feature vector relates to an attack, given the training data. Detection probabilities and identifications are saved for human interpretation.

Since we already had an effective feature set for the off-line system, we merely needed to optimize it for the real-time system. To reduce processing, we performed all analysis using a single feature set for both detection and identification. The number of features in the single feature set was also reduced to those that were most useful in classifying the DARPA data, as indicated by classifier weights.

In addition, a few new network profile-based features were examined. For the profile, valid IP addresses were recorded along with common services that were active on those IP addresses. Connection rates to the various services were also recorded. Once this profile was constructed, feature extraction determined the variation in network traffic between what was present and what was expected. The feature extractor noted connections to IP addresses that didn't exist in the network profile along with connections to common services on machines that didn't have those services active. These two features contributed towards novel probe detection and helped detect novel denial-of-service attacks.

Several other optimizations needed to be made to keep up with high data rates. Instead of computing a result for all classifiers and voting, processing was done in stages, and an alert produced at one stage would prevent further processing. There were several binary flags in the new feature set. These flags and the network profile deviations were the first things checked. If a flag was set, then the detection score produced was a pre-determined value based on the type of flag.

Once the binary flags in the feature set were evaluated, the system examined the network profile. If the current connection was destined for an IP/Port combination that was not present in the network profile, then the detection score was set to:

$$\frac{\text{malicious 1998}}{\text{total 1998}}, \text{ if novel IP/port.} \quad (1)$$

This computes the fraction of malicious sessions to new IP/port combinations present in the 1998 data. Classification was not performed in this case.

If the current connection was established for an IP/port combination in the network profile, then a feature was computed that compared the deviation between the observed rate of connection to the current IP/port combination and what was expected. The feature was calculated using the following formula:

$$1 - e^{-\frac{-(\text{observed} - \text{expected})}{\text{expected}}}, \text{ if IP/port profile.} \quad (2)$$

In this equation, observed refers to the number of connections to the current IP/port combination within the time window being used, and expected refers to the number recorded in the network profile based on normal traffic.

The identification classifiers were also selectively triggered. Once a detection score was produced, it was compared to a hard-coded threshold. The identification classifiers were used only in the cases when the detection score exceeded the threshold. Since under normal operating conditions the majority of the traffic encountered by the system is not attack-related, processing normally ceases once the detection classifiers have been triggered.

The real-time system was evaluated on the same 1999 DARPA data as the offline system, with the results presented in Fig. 5. These results cannot be compared with other evaluated systems, because we used the test results from the offline system to select new features and build an improved system. They are most useful for verifying that the system performance did not diminish when we changed our processing to create a real-time system. In fact, the best detection for probes with false alarm rates below one per day increased slightly from 72% to 84% and for DoS attacks it decreased slightly from 68% to 64%. Because the DARPA data included more examples of probing attacks than DoS attacks, overall system detection rate improved slightly, although the false alarm rate also increased slightly.

The roughly 10 percent improvement to probing detection rate can be attributed to the new network profile features. These features allowed the classifiers to detect novel attacks that were missed by the offline system.

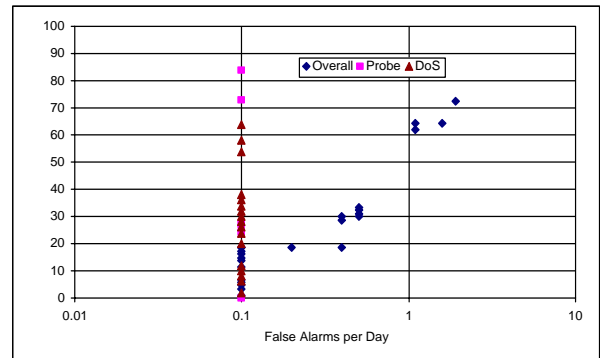


Fig 4. Real-time System performance.

Once a performance benchmark had been established, the system was deployed at MIT Lincoln Laboratory's connection to the Internet. First, data was gathered to create an accurate network profile. Second, network characteristics were measured to determine if the network being monitored was similar to the network on which the classifiers had been trained. Since the data rates, network topologies, services present, and user behavior were similar, it was determined that the classifiers did not need to be retrained on the network being monitored. The system was deployed for a period of 5 days. The average data rate during that time was 4.8 Mb/sec. The system generated an average of 5 alerts a day. It alerted on several TCP connections that had their reserved bits set. This is characteristic of probes that try to determine what type of operating system a host is running. There were also several TCP connections to services on IP addresses that did not support those services according to the network profile. After examining the data, it was unclear whether alerts associated with the TCP reserved bits were real probes or false alarms. The network profile was updated after it was determined that the connections it had alerted on were connecting to valid services. An improvement to this manual process would be an automatic update of the network profile that waits for a response from the destination machine before issuing an alert. If the response indicates that the service is active then the network profile can be appropriately updated and the alert suppressed. However, if the response isn't characteristic of an active service, e.g. an ICMP Destination Unreachable Packet is observed, then the session should be flagged with an alert. This automation would slow alert production and could affect detections of stealthy attacks.

IV. DISCUSSION

Others have proposed using network profiles to detect probing attacks [2], although this is the first use of connection-based windows to detect low-profile attacks of which we are aware. The technique is robust and works well for detecting new and stealthy attacks.

The variable output is quite useful for allowing a user to adjust the detection-false alarm trade-off, and it is useful to know the posterior probability that the system has seen an attack. This information has proven useful for input to

more complex fusion systems [11].

Further development of this tool will involve longer test periods on a live network to determine system weaknesses and resource consumption. The system needs to be tested to determine the highest data rate that it can handle. A mechanism needs to be added so that the network profile can be automatically updated as the network configuration changes. In addition, analysis of other protocols such as HTTP, ARP, NetBios, and DHCP needs to be added.

V. SUMMARY

We have built and evaluated a system that detects and identifies probes and denial of service attacks. Statistics are recorded using time windows, which others have shown are useful for detecting DoS attacks. It also includes connection windows, which we have shown are useful for detecting low profile probing attacks. It is especially accurate when detecting stealthy and new probes, but is also good at detecting denial-of-service attacks. Its accuracy can be attributed in part to an internal model of the local network traffic.

VI. REFERENCES

- [1] M. Bishop, S. Cheung, C. Wee, J. Frank, J. Hoagland, and S. Samorodin, "The Threat from the Net," *IEEE Spectrum*, vol. 34, pp. 56-63, 1997.
- [2] S. Staniford, J. Hoagland, and J. McAlerney, "Practical Automated Detection of Stealthy Portscans," presented at ACM CCS IDS Workshop, Athens, Greece, 2000.
- [3] Lincoln Laboratory, MIT (1999) "DARPA Intrusion Detection Evaluation", <http://www.ll.mit.edu/IST/ideval/index.html>.
- [4] J.W. Haines, R. P. Lippmann, D. J. Fried, M. A. Zissman, E. Tran, and S. B. Boswell, "1999 DARPA Intrusion Detection Evaluation: Design and Procedures," MIT Lincoln Laboratory Technical Report 1062, 26 February, 2001.
- [5] R. P. Lippmann, R. K. Cunningham, D. J. Fried, S. L. Garfinkel, A. S. Gorton, I. Graf, K. R. Kendall, D. J. McClung, D. J. Weber, S. E. Webster, D. Wyschogrod, and M. A. Zissman, "The 1998 DARPA/AFRL Off-Line Intrusion Detection Evaluation," presented at First International Workshop on Recent Advances in Intrusion Detection, Louvain-la-Neuve, Belgium, 1998.
- [6] R. P. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Computer Networks*, vol. 34, pp. 579-595, 2000.
- [7] R. P. Lippmann and J. Haines, "Analysis and Results of the 1999 DARPA Off-line Intrusion Detection Evaluation," presented at Recent Advances in Intrusion Detection, Toulouse, France, 2000.
- [8] QueSO Scanner, <http://www.apostols.org/projectz/queso/>.
- [9] R. K. Cunningham, R. P. Lippmann, and S. E. Webster, "Detecting and Displaying Novel Computer Attacks with Macroscopic," *IEEE Systems, Man and Cybernetics Part C*, in press.
- [10] R. P. Lippmann, L. C. Kukulich, and E. Singer, "LNKnet: Neural Network, Machine Learning, and Statistical Software for Pattern Classification," *Lincoln Laboratory Journal*, vol. 6, pp. 249-268, 1993.
- [11] A. Valdes and K. Skinner, "Adaptive, Model-Based Monitoring for Cyber Attack Detection," presented at Recent Advances in Intrusion Detection, Toulouse, France, 2000.