Improved Detection of Low-Profile Probe and Denial-of-Service Attacks[1]

William W. Streilein, Robert K. Cunningham, Seth E. Webster

Abstract- **We present enhancements to our network-based intrusion detection system, which makes use of multiple neural network classifiers to accurately detect several classes of attacks including stealthy probes and novel denial-of-service attacks. An intrinsic representation of the local network and detection features derived from network traffic enable the system to detect entire attack classes. Improvements to our system include enhanced robust TCP session reconstruction, handling simplex and duplex traffic modes, an expanded feature vector that includes measures of inter-packet delays and counts of anomalous TCP sessions, and binary tree-based internal data structures which are faster and less vulnerable to attack. Our system achieves a detection rate of 100% with a false alarm rate of .1% when tested against stealthy attacks in the DARPA 1999 IDS Evaluation. It also performs well on a moderately loaded research network.**

.

***Index terms*—intrusion detection, security, probe; denial-of-service, neural networks**

I. Introduction

As more people make use of the Internet, their computers and the valuable data they contain become exposed to attackers lying in wait in cyberspace. Attackers are constantly scanning the Internet for victim machines that can be broken into and commandeered in order to suit their malicious purposes, such as, the enlistment of new zombies for distributed denial-of-service attacks, the unauthorized use of network storage resources or the defacement of corporate or government web-pages. In order to protect computer systems, network-based intrusion detection systems (IDSs) have been developed to analyze Internet traffic and recognize when attackers are at work probing a network or attacking a machine. State-of-the-art network-based intrusion detection systems detect attackers by comparing network traffic with signatures of known attacks. Knowledgeable attackers can alter the details of many attacks to avoid using the short signatures detected by these systems. System Design

In this paper, we present enhancement to our network-based intrusion detection system, which makes use of multiple neural network classifiers to accurately detect several classes of attacks including stealthy probes and novel denial-of-service attacks. Improvements to our sisteym include enhanced robust TCP session reconstruction, handling of simplex and duplex traffic modes, an expanded feature vector that includes measures of inter-packet delays and counts of anomalous TCP sessions, and binary tree-based internal data structures which are faster and less vulnerable to attack.

This paper is organized as follows. We begin with a description of our system architecture, going into some detail about our internal data structures. Section III discusses the enhanced feature extraction methods employed in our system. In section IV, we present the results from training and testing our system on traffic data from the DARPA 1998 and 1999 evaluations. Following a brief discussion of our results of feature selection we close with a conclusion and general discussion about network-based intrusion detection and some suggestions for follow-on work.

II. System Description

As depicted in Figure 1, our system contains several modules that permit the analysis of network data for the purposes of detection network intrusions.

[1] This work was sponsored by the Department of the Air Force 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.

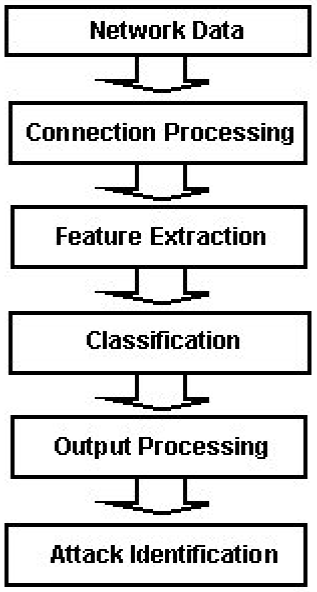

Fig. 1. Diagram of processing modules in our network-based intrusion detetion system.

After reception of the network data, the data analysed and compiled into connections by the *Psplice* connection-parsing library described in the next section. This library supports the intrusion detection application, by taking care of organizing traffic on the network into coherent connection streams. The application receives indications from the *Psplice* library when important 'events' occur on the connection such as: OPEN – when the 3-way handshake is received; DATA – when data is received on the connection, and CLOSE – when the connection is closed for any reason, such as a reset condition or normal 3-way goodbye.

A. Psplice Connection-Parsing Library

Psplice is a library, written in C++, for reliably reassembling the data transferred in a TCP connection from a packet stream despite attempts by a malicious host to confuse it. Specifically, it is resilient in the face of many of the attacks discussed in [reference to Ptachek and Neuman paper]. The Psplice library allows applications to treat TCP stream reassembly as a "black box" in which packets from multiple connections are input and connection information, along with the data transferred in those connections, is output. Furthermore, applications which need more precise control over exactly how data is reassembled or special cases are handled can extend or replace certain key classes without having to modify the library's framework.

Psplice maintains accurate connection state by delaying the decision on whether to accept or drop a packet until the recipient's response to that packet can be observed. Instead of attempting to predict if a packet will reach a host and how it will be handled, Psplice simply observes the hosts response to see if the connection state on that host has changed, indicating the packet was accepted and acted upon. Using the recipient's response to directly observe a packet's fate prevents an attacker from taking advantage of network topology or operating system specific behavior to craft a packet that would be either dropped by the recipient but accepted by Psplice or vice versa. While combinations of attacks are still able to confuse it, Psplice's packet verification techniques make inserting or spurious packets or causing packets to be erroneously dropped much more difficult.

B. Internal Data Structures

Changes were made to the internal data structures of our system to support enhanced real-time detection of stealthy and distributed attacks and to reduce the effects of an attack against the system itself. In general, all internal data tables are binary-tree based data structures, which are less vulnerable to attack than linked-list data structures and promise predictable insert and search time requirements.

Several data structures are used to characterize the local network on which the system is deployed to provide the system with an idea of normal traffic and host configurations. The first of these data structures is the network profile table which tracks connections to local hosts by frequency. The ARP and DHCP data structures track IP/Mac Address mappings and general network configuration information for the purposes of recognizing denial-of-service attacks which attempt to 'poison' a host's configuration through spoofing.

The connection anomaly table tracks the anomalous-ness of each new connection seen on the network by first checking the network profile table for it's likelihood. A score derived from the likelihood of this connection occurring, is used as part of the feature vector delivered to the classification engines and helps to discover rare connection events that might indicate the presence of a stealthy probe.

A central table of denial-of-service alerts is used as a clearinghouse for all DoS alerts issued by the system. The purpose of this table is to support the the detection of distributed DoS attacks. A separate algorithm considers DoS type (determined during classifcation), source IP network and time and duration of attack to group separate DoS alerts together and attempt to recognize a distributed attack.

C. *Enhanced Processing of Events*

In addition to the in-band event-driven processing that acts upon individual connection events delivered from the *psplice* library, our system utilizes periodic timer event mechanism to permit out-of-band processing of alerts. Such out-of-band processing includes the timing-out of alerts that are being considered in the Alert table and to consider individual DoS alerts as part of a distributed alert scheme. As depicted in Figure 2

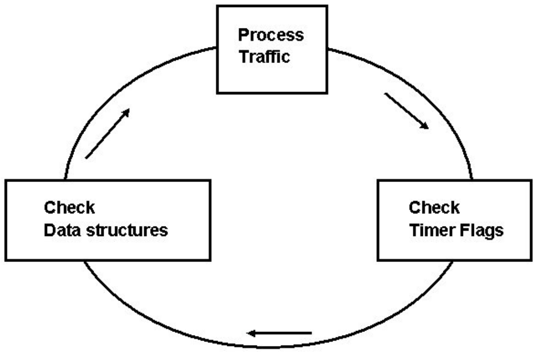

Fig. 2. Processing loop for real-time connection events and out-of-band maintenance and alert aggregation. In-band connection event processing is complemented by out-of-band alert time-out and aggregation.

D. *Output Control/Throttling*

A final new data structure supports the output of the intrusion detection system by controlling the flow of out going alerts issued. All alerts are labeled with their specific characteristics including: src/dst IP, src/dst Port, type of alert, time of occurrence. This enables alerts of the same intrusion event to be grouped together and allows only the first alert in a series of alerts needs to be issued. A major complaint of many intrusion detection systems is that they tend to produce a flood of alerts during an attack often overwhelming the analyst. This change attempts to remedy the problem by stemming the flow of alerts. Several configuration parameters, such as alert-timeout period and the frequency and count of alerts during an on-going attack, enable the control of this feature to match deployment requirements.[need reference];

A second way in which the system is improved is through the use of throttling on the alerts produced. Often an intrusion detection system can produce many alerts during an attack, one for each connection or network event that appears to be part of an attack. In our system we make use of some simple heuristics to throttle the flow of alerts being issued to the system administrator. The first of these heuristics involves a time window surrounding the production an alert. This time window suppresses all identical alerts produced by the system after the first one for a period extending 20 seconds beyond the receipt of the last alert produced for this alert.

Alert throttling

Heuristics provide guide for alert throttling

Require more than 1 alert for DoS attack
Alert only once every 20 seconds
Stop inundation with alerts
Configurable to administrators likings.

.

E. *Feature Extraction and Classification*

Our system relies upon individual neural-network classifiers to detect the presence of probe and denial-of-service attacks in network data. In this section we describe the feature vector that is constructed from connection data and connection history and presented to the neural network classifier.

1) *Feature Vector*

Several improvements were made to our method of extracting features from connection data. These improvements in the feature vector take advantage of the closer connection to actual network traffic which is supported by the *psplice* library. For examples, *Psplice* provides an indication to the intrusion detection system when the first SYN packet is received on a new connection rather than waiting for the full 3-way handshake to complete. This early indication of the a connection attempt supports the intrusion detection applications tracking of uncompleted connections and enables it to detect attacks that utilize this stealth mode of connecting in order to probe a system or to hide from being reported in a system log.

The feature vector contained elements that monitor immediate packet information as well as aggregate information dependent upon the source or destination IP address and application port. The complete feature vector included: an indication of the connection state: opened or closed; the protocol: tcp, udp or icmp; an indication of strangeness in the connection as reported by *psplice*: no TCP FIN flag, TCP reserved bits set; unusual connection: strange IP/PORT combination, source IP from inside network (based upon network configuration); packet fragment information: # of overlapping fragments, # of enclosed fragments; features of connection close events: # to same host, # to same service, # abnormal connections; features of connection open events: # to same host, # to same service; ratio of open to close events: ratio to same host, ratio to same service; connection timing: same host open interval, same host close interval; and raw connection counts: # different services connected to, # connections to same service, # icmp echos from source.

F. *Neural Network Classifiers*

Our system employs individual neural classifiers for each of the attacks which it is trying to detect and classify. Probe classifiers include ipsweep, portsweep and satan attack, while denial-of-service classifiers include the neptune, smurf and teardrop attacks. Each of the classifiers is trained with feature vectors compiled from examples of attack and non-attack data. The set of attack examples includes features vectors compiled for only the attack relevant to the classifier. While the set of 'normal' examples includes feature vectors compiled from normal, non-attack traffic and examples from the other attacks. The addition of other-attack data to the 'normal' data set, represents a modification in the training regimen from our previous system and was an attempt to teach the neural classifiers to improve the specificity of the detections.

All classifiers were multi-layer perceptrons trained off-line using the LNKNet pattern recognition library from Lincoln [Reference needed]. The output from this LNKNet library is a set of trained detectors which are easily incorporated into the intrusion detector's application software.

III. Results

A. *Training and Testing Methodology*

Our system was trained and tested against data from the 1998 and 1999 DARPA Intrusion Detection System Evaluation carried out by MIT Lincoln Laboratory [6][20]. This dataset contains examples of many types of attacks included user-to-root and remote-to-local attacks as well as other OS-dependent attacks. For the purposes of our system, however, we focused on stealthy probe attacks as denial-of-service attacks. The format for the evaluation is of individual files for each day of the test stored in tcpdump format.

For training our system, we extracted individual examples of probe and denial-of-service attacks from the 1998 training and test data and the 1999 training data. For each attack file, feature vectors were compiled and then utilized to train the neural classifiers. The neural classifiers were trained using 10-fold cross-validation to discriminate between their specific attack class and the other attack classes and normal traffic.

When presented with original tcpdump files from the two weeks of testing, the system performed as shown in Figure XX. As shown in the figure, the system did quite well, achieving detection and classification of all of the stealth attacks with a very low false alarm rate of < 1% per day. This represents a significant improvement over the performance of the previous system.

Indicate previous performance results

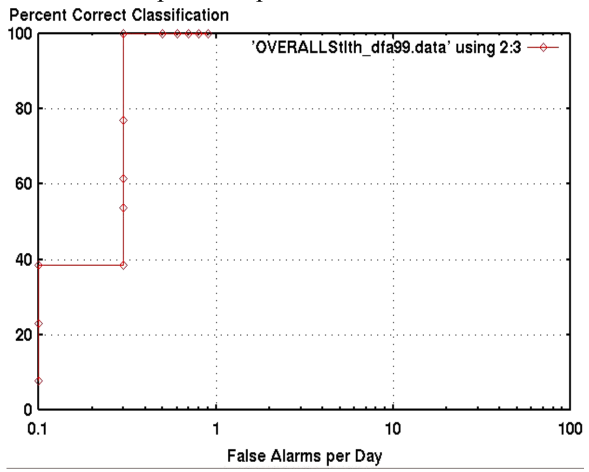

Figure XXX presents classification results for our system for probe attacks, stealthy and clear, old and new.

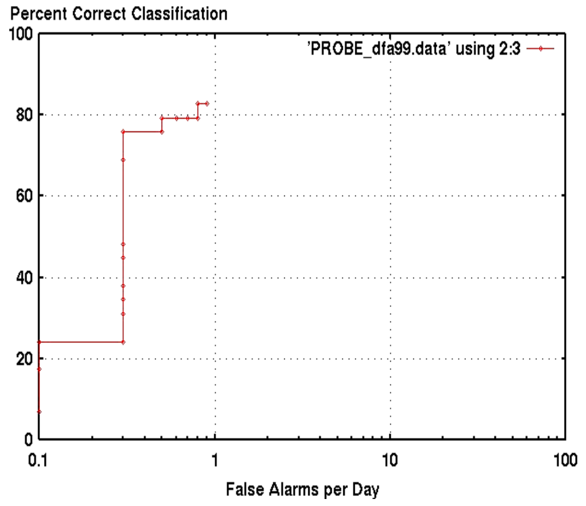

Fig. 3. System Accuracy on Probe attacks from the 1999 DARPA Evaluation

Figure IV presents the results classification for our system for detection and classification of denial-of-service attacks.

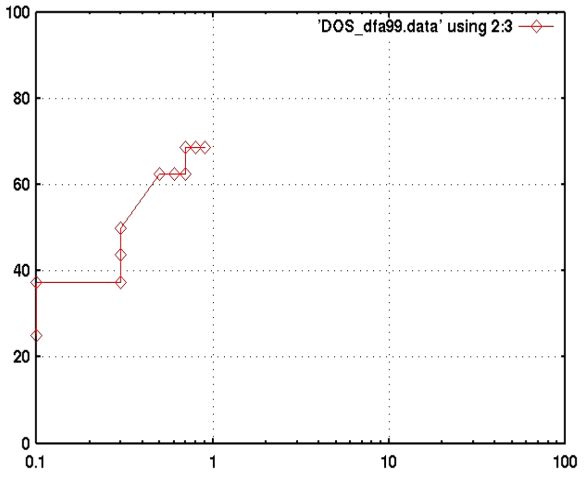

Fig. 4. System Accuracy on Denial-of-Service attacks from the 1999 DARPA Evaluation

IV. Feature Selection

Improved performance of our intrusion detection system is in part to many of the enhancements we have made, not the least of which includes the addition of new features elements. In an effort to understand the source the improved detection, we performed backward feature selection on the training data. In backward feature selection, the goal is to being with a complete feature vector and gradually remove the element that has the least effect on the performance of the classifier. In the end you are left with a minimal set of features that support the desired classification rate. Figure 5 presents the classification error as a function of feature number. From the figure it appears that most important features for the detection of probes and denial-of-service attacks are 2,6, 21, 8 and finally feature 17. These features correspond to

2- ICMP FLAG
6 STRANGE IP/PORT
21 SAME HOST OPEN TIME CNT
8 INSIDE IP
17 DIFFERENCT SERVICES connected to

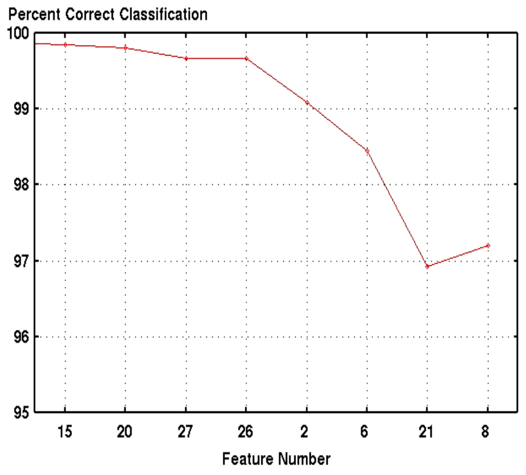

Fig. 5. Results of feature selection based upon misclassification of training set vectors

V. Future Work

Future enhancements to our system include dynamice network map building that take a cuu from system administrators as to the configuration of the network. This can assist the detection algorithm in ruling out potential false alarms by recognizing common behavior from known machines.

Other further work would include the

VI. Conclusions

Our enhancements to our network-based intrusion detection make a system with incredible potential.

VII. References

[1] “Axent Intruder Alert User Manual,” V. 3.0, Oct. 1998.

[2] D. Anderson, T. Lunt, H. Javitz, A. Tamaru, and A. Valdes. "Safeguard final report: detecting unusual program behavior using the NIDES statistical component," Computer Science Laboratory, SRI International, Menlo Park, CA, Technical Report, December 1993.

[3] T. Berners-Lee, R. Fielding, and H. Frystyk, “Hypertext Transfer Protocol -- HTTP/1.0,” RFC 1945, 1996.

[4] M. Bishop,, S. Cheung, C. Wee. "The Threat from the Net", IEEE Spectrum, 1997, 38(8).

[5] Cisco Systems, Inc. "NetRanger Intrusion Detection System Technical Overview, ”http://www.cisco.com/warp/public/778/security/netranger/ntran_tc.htm, 1998.

[6] R. Cunningham, R. Lippmann, D. Fried, S. Garfinkel, I. Graf, K. Kendall, S. Webster, D. Wyschogrod, and M. Zissman, "Evaluating intrusion detection systems without attacking

your friends: The 1998 DARPA intrusion detection evaluation," in Proceedings of ID'99, Third Conference and Workshop on Intrusion Detection and Response, San Diego, CA: SANS Institute, 1999.

[7] R. Cunningham, R. Lippmann, D. Kassay, S. Webster, and M. Zissman, Host-based Bottleneck Verification Efficiently Detects Novel Computer Attacks," MILCOM'99, November 1999.

[8] H. Debar, M. Becker, and D. Siboni, "A Neural Network Component for an Intrusion Detection System," in Proc. of IEEE Computer Society Symposium on Research in Security and Privacy, 1992.

[9] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, "A sense of self for Unix processes," in Proceedings of 1996 IEEE Symposium on Computer Security and Privacy, 1996.

[10] S. Forrest, S. Hofmeyr, and A. Somayaji, "Computer Immunology," Communications of the ACM, 40(10), 88-96, 1997.

[11] A. K. Ghosh, A. Schwartzbard and M. Shatz, "Learning Program Behavior Profiles for Intrusion Detection", in *Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, California, April 1999, http://www.rstcorp.com/~anup/.

[12] T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A Network Security Monitor", in IEEE Symposium on Research in Security and Privacy., 1990, pp. 296-304.

[13] H. Javitz, and A. Valdes, "The NIDES statistical component description and justification," Computer Science Laboratory, SRI International, Menlo Park, CA, Technical Report, March 1994.

[14] T. Heberlein, "Network Security Monitor (NSM) - Final Report", U.C. Davis: Feb. 1995, http://seclab.cs.ucdavis.edu/papers/NSM-final.pdf.

[15] R. Kemmerer. "NSTAT: A Model-based real-time network intrusion detection system," Computer Science Department, University of California, Santa Barbara, Report TRCS97-18, http://www.cs.ucsb.edu/TRs/TRCS97-18.html.

[16] C. Ko, M. Ruschitzka, and K. Levitt, "Execution Monitoring of Security-Critical Programs in a Distributed System: A Specification-Based Approach," in Proc. IEEE Symposium on Security and Privacy, 1997, pp. 134-144, Oakland, CA: IEEE Computer Society Press.

[17] Lawrence Livermore Nat'l Laboratory (1998). "Network Intrusion Detector (NID) Overview," Computer Security Technology Center, http://ciac.llnl.gov/cstc/nid/intro.html.

[18] Lippmann, R.P., et al. "Using Bottleneck Verification to Find Novel New Attacks with a Low False Alarm Rate," in Recent Advances in Intrusion Detection, 1998, Louvain-la-Neuve, Belgium.

[19] T. Lunt,, "Automated Audit Trail Analysis and Intrusion Detection: A Survey", in Proceedings 11th National Computer Security Conference., 1998, pp. 65-73.

[20] MIT Lincoln Laboratory, "Intrusion detection evaluations," http://www.ll.mit.edu/IST/ideval/index.html.

[21] V. Paxon, "Bro: A System for Detecting Network Intruders in Real-Time," in Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, January 1998, http://www.aciri.org/vern/papers.html.

[22] P. Porras, and P. Neumann, "EMERALD: Event Monitoring Enabling Response to Anomalous Live Disturbances," in Proceedings 20th National Information Systems Security Conference, Oct 7, 1997.

[23] J. Postel, "Transmission Control Protocol," RFC 793, USC/Information Sciences Institute September 1981.

[24] T. Ptacek, and T. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection," Secure Networks, Inc. January, 1998.

[25] M. Ranum, K. Landfield, M. Stolarchuk, M. Sienkiewicz, A. Lambeth, and E. Wall. "Implementing A Generalized Tool For Network Monitoring", Eleventh System Administrators Conference (LISA), 1997.

[26] J. Ryan, L. Meng-Jang, and R. Miikkulainen, "Intrusion Detection with Neural Nets," in Advances in Neural Information Processing Systems 10, Edited by M. Jordan, M. Kearns, and S. Solla, MIT Press: Cambridge, MA, 1998, pp. 943-94

[27] S. Webster, "The Development and Analysis of Intrusion Detection Algorithms," Masters Thesis in Computer Science, Massachusetts Institute of Technology: Cambridge, MA, June 1998.