Intelligent Agents for Intrusion Detection

Guy G. Helmer,[†] Johnny S. K. Wong, Vasant Honavar, and Les Miller
Iowa State University, Ames, Iowa 50011

Abstract

This paper focuses on intrusion detection and countermeasures with respect to widely-used operating systems and networks. The design and architecture of an intrusion detection system built from distributed agents is proposed to implement an intelligent system on which data mining can be performed to provide global, temporal views of an entire networked system.

A starting point for agent intelligence in our system is the research into the use of machine learning over system call traces from the privileged `sendmail` program on UNIX. We use a rule learning algorithm to classify the system call traces for intrusion detection purposes and show the results.

INTRODUCTION

Our experience in managing networks of computer systems with known vulnerabilities has resulted in the investigation of the problem of detecting misuse of computer systems [4]. Because of the effort required to monitor systems and networks manually, we could not easily detect either attempts at misuse or successful attacks without the help of intelligent intrusion detection systems.

We propose an artificially intelligent system for intrusion detection and countermeasures on computer systems in a network environment using data mining technology. The system would be built using distributed intelligent agents to apply a data mining approach to intrusion detection. Data gathering agents will render system logs and activity data into common formats while low-level agents classify recent activities and provide data and current classification states to each other and to a higher level of agents that implement data mining over the entire knowledge and data sources of the system.

RELATED WORK

Projects related to our intrusion detection project include DIDS, Computer Immunology, JAM, and EMERALD.

The Distributed Intrusion Detection System (DIDS) [8] uses a combination of host monitors and local area network monitors to monitor system & network activities with a centralized director aggregating information from the monitors to detect intrusions. It is similar to our agent system for intrusion detection and countermeasures in that it uses multiple monitors and artificial intelligence algorithms to determine the severity of events. DIDS differs from our system in that the intelligence is purely centralized, and it does not make use of any agent technology.

The Computer Immunology project [6] explored designs of intrusion detection systems that can effectively detect and defend intrusions in a networked computer system in a manner similar to the immune system in animals. One portion of the project [5] researched a method that could provide a component of an immune system for computers. They developed a sense of "self" for privileged programs by creating a database of normal and abnormal system call traces for instances of execution of the programs.

The Java Agents for Meta-Learning (JAM) Project [11] is the most similar to our proposed agent system for intrusion detection and countermeasures in that it used intelligent, distributed Java agents to learn models of fraud and intrusive behavior. The knowledge learned by the distributed Java agents can be exchanged and used to help train other agents that can identify attacks based on the combined knowledge. The JAM project built on the work done by Forrest et. al [5] in the area of identifying attacks against privileged programs. A portion of our work derives from this idea of detecting intrusions based on system call traces of privileged programs [7].

The SRI EMERALD project addresses the problems of network intrusions via TCP/IP data streams [10]. EMERALD's design is similar to DIDS's in that network surveillance monitors observe local area network traffic and submit analysis reports to an enterprise monitor, which correlates the reports. Like DIDS, EMERALD

[†] This work supported in part by the Applied Mathematical Sciences Program of the Ames Laboratory, U. S. Department of Energy under contract number W-7405-ENG-82.

appears to concentrate the intelligence in a central system and does not incorporate any agent technology.

DESIGN AND IMPLEMENTATION OF THE AGENT-BASED SYSTEM

A system of intelligent agents using collaborative information and mobile agent technologies [1] [9] is developed to implement a prototype intrusion detection system [3].

The goals of the system design are to:

- Learn to detect intrusions on hosts and via networks using individual agents targeted at particular subsystems;
- Use agent technologies to intelligently process audit data at the sources by using mobile agents;
- Have agents collaborate to share information on suspicious events and determine when to be more vigilant or more relaxed;
- Apply data mining techniques to the heterogeneous data and knowledge sources to identify and react to coordinated attacks on multiple subsystems.

A notable feature of the intrusion detection system based on data mining is the support it offers for gathering and operating on data and knowledge sources from the entire observed system. The system could identify sources of concerted or multistage attacks, initiate countermeasures in response to the attack, and provide supporting documentation for system administrators that would help in procedural or legal action taken against the attacker.

An example of an attack involving more than one subsystem would be a combined NFS and rlogin attack. In the first step, an attacker would determine an NFS filehandle for an `.rhosts` file or `/etc/hosts.equiv` (assuming the appropriate filesystems are exported by the UNIX system) [12]. Using the NFS filehandle, the attacker would re-write the file to give himself login privileges to the attacked host. Then, using rlogin from the formerly untrusted host, the attacker would be able to login to an account on the attacked host, since the attacked host now mistakenly trusts the attacker. At this point, the attacker may be able to further compromise the system. The intrusion detection system based on data mining would be able to correlate these attacks, help identify the origin of the attack, and support system management in responding to the attack.

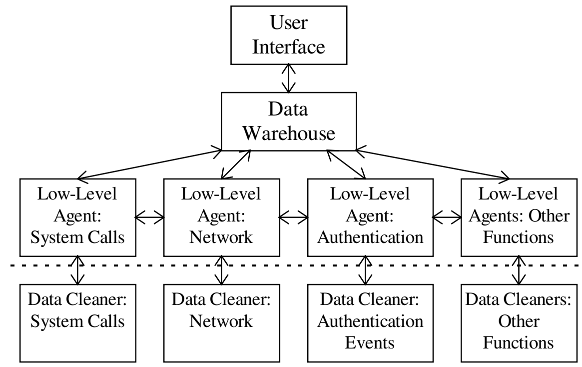

Figure 1: Architecture of the Intrusion Detection System

System Design

The components of the agent-based intrusion detection system are shown in Figure 1. Distributed data cleaning agents process data obtained from log files, network protocol monitors, and system activity monitors on heterogeneous systems. A lower-level layer of agents, just above the data cleaning agents in the system architecture, form the first level of intrusion detection. Using mobile agent technology, these agents travel to each of their associated data cleaning agents, gather recent information, and classify the data to determine whether suspicious activity is occurring. Like the JAM system [11], the agents will be able to use a variety of classification algorithms, the choice of which will depend on the data. Unlike the JAM system, though, the agents at this level will collaborate to cooperatively set their suspicion level so as to determine whether a suspicious action is more severe in the presence of other suspicious activity.

At the top level, intelligent agents maintain the data warehouse by combining knowledge and data from the lower layer of agents and applying data mining algorithms to discover associations, suspicious events that occur together with some frequency, and patterns. Because the data warehouse would provide a global, temporal view of the knowledge and activity of the monitored distributed system, we believe this system could help train system administrators to spot & defend attacks as well as assist system administrators in developing better protections and countermeasures for their systems and identifying new attacks.

The user interface to the agent-based intrusion detection system directs the operation of the agents in the system and shows the status reported by the low-level agents. When the data warehouse portion of the system is developed, the user interface will provide access to its features, including managing the knowledge in the warehouse and applying mining functions to discover associations and correlations from the stored knowledge.

SYSTEM CALL TRACES

To develop a prototype intelligent agent for our intrusion detection system, we used data made available by the University of New Mexico containing system call traces for normal and abusive use of the `sendmail` program as run on SunOS 4.

It has been shown that system call traces can be used to identify anomalous use of privileged programs and thus determine whether this class of attacks are being mounted against a system. Forrest et. al [5] developed databases of system calls from normal and anomalous uses of two privileged programs, `sendmail` and `lpr`. They showed that a database of known good sequences can be developed from a reasonably sized set of `sendmail` executions, and then they showed that intrusive behavior can be determined by comparing system call sequences against the database of known good sequences. [7] used the data from Forrest's project to show that a rule learning algorithm can learn to classify normal and abnormal system call sequences.

The University of New Mexico system call data is simply a set of files consisting of lines giving a process ID number (PID) and system call number. [7] shows the use of a window of length k+1 with a step size of 1 across the system call trace sequence, where k is varied in different experiments to determine the best window length. When used on training data, each window is classified "normal" if it matches a window obtained from proper operations of `sendmail`, else the window is classified "abnormal". An example of system call windows and training labels are shown below in Table 1.

System Call Sequences (k=6)	Label
4, 2, 66, 66, 4, 138, 66	Normal
2, 66, 66, 4, 138, 66, 5	Normal
66, 66, 4, 138, 66, 5, 5	Normal
66, 4, 138, 66, 5, 5, 4	Abnormal
4, 138, 66, 5, 5, 4, 39	Abnormal

Table 1: Sample System Call Windows With Training Labels

We duplicated the JAM approach [7] of grouping system calls into sequences and using the RIPPER [2] learning algorithm to classify system call windows.

In the JAM approach, the quality of the learned classification rules is determined by the gap between the normal `sendmail` testing traces and the abnormal trace with the lowest "% abnormal regions" score. In our first experiment, we computed a gap of 0.47%, which is disturbingly smaller than the 1.9% seen in [7] Experiment B. However, the more dangerous attacks, such as sscp, syslog-local, and syslog-remote, were clearly identified using this approach, as their percentage of abnormal regions are at least an order of magnitude larger than the percent of abnormal regions seen from the normal `sendmail`.

In our second experiment, where we used a different technique to assemble the training data, the computed gap was 0.71%, which again was smaller than the 1.9% seen in Lee's Experiment B. We attribute the discrepancies in our results as due to the differences in selection of the training data, which [7] does not explain in detail.

EXTENDING THE SYSTEM CALL EXPERIMENT

A feature vector technique was proposed for application to the system call traces, using a single bit to indicate whether a particular sequence appeared in a sequence of system calls. This feature vector can then be used with any number of learning algorithms, allowing us to easily compare the performance of other algorithms to RIPPER. With the sequence window set to 7, 1112 normal sequences and 704 abnormal sequences were seen in the `sendmail` system call data. Thus, a bit vector of length 1816 was used as the feature vector for the `sendmail` traces. Feature vectors were computed on a per-PID basis from the `sendmail` system call traces.

The same training data selection technique was used for the feature vector technique as was used for the previous two experiments. 80% of the PID files for normal traces and all of the PID files for the four selected anomalous traces were used as training data. Since the set of abnormal training data was quite small (15 records) in proportion to the set of normal training data (520 records), the set of abnormal training data was duplicated 36 times so that 540 abnormal records were present in the training data as opposed to the 520 normal records. RIPPER quickly learned a very simple rule set, consisting of 4 fairly simple rules, as opposed to the 208 rules RIPPER learned in one of the previous experiments.

The rule set learned by RIPPER from this data produced the results shown in Table 2, where the total number of feature vectors, number of vectors predicted abnormal by RIPPER, and percentage of predicted-abnormal feature vectors is shown. Since only one feature vector is computed for each process, each trace tends to have few feature vectors.

Trace Name	Total Feature Vectors	Feature Vectors Predicted Abnormal	% Feature Vectors Predicted Abnormal
chasin	6	2	33.33
decode1	6	1	16.67
decode2	6	1	16.67
fwd-loops-1	2	1	50.00
fwd-loops-2	1	0	0.00
fwd-loops-3	2	1	50.00
fwd-loops-4	2	1	50.00
fwd-loops-5	3	1	33.33
recursive	25	3	12.00
sm565a	3	1	33.33
sm5x	8	2	25.00
smdhole	3	2	66.67
sscp-1	1	1	100.00
sscp-2	1	1	100.00
sscp-3	1	1	100.00
syslog-local-1	6	6	100.00
syslog-local-2	6	5	83.33
syslog-remote-1	7	7	100.00
syslog-remote-2	4	4	100.00
Normal sendmail	120	1	0.83

Table 2: Results of Learning Rules for Feature Vectors

The idea behind the feature vector approach is to predict at least one of the processes involved in an intrusion as abnormal. From this experiment we see that this is not only possible but tends to have more clearly defined results than the previous experiments.

Overall, both types of learning experiments using RIPPER on the system call data from sendmail traces appeared to work well. Classifying processes by windows over predictions and the feature vector approach both seemed to provide the desired capability to determine whether an intrusive use of the `sendmail` program had taken place.

CONCLUSION AND FUTURE WORK

The distributed intelligent agent approach to intrusion detection and countermeasures has been described. The work on the `sendmail` system call traces showed the use of a machine learning approach to intrusion detection on a component of the distributed system. A portion of the JAM project's work in this area has been duplicated and a different data representation, the feature vector approach, seemed to work well.

Future work includes the study of using the feature subset selection approach [13], which shows the potential of improving classification by selecting the most influential features in a training set.

BIBLIOGRAPHY

[1] J. M. Bradshaw, "An Introduction to Software Agents", in *Software Agents*, Bradshaw, J.M. (ed.), Cambridge, MA: MIT Press, 1997.

[2] W. W. Cohen, "Fast Effective Rule Induction", in *Machine Learning: The 12th International Conference*, Lake Tahoe, CA, 1995.

[3] M. Crosbie and G. Spafford. "Defending a Computer System using Autonomous Agents", in *Proceedings of the 18th National Information Systems Security Conference*, October 1995.

[4] Dorothy Denning, "An Intrusion-Detection Model", *IEEE Transactions on Software Engineering*, no. 2, page 222, February 1987.

[5] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A Sense of Self for UNIX Processes", in *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, Los Alamitos, CA, 1996, pages 120-128.

[6] S. Forrest, S. Hofmeyr, and A. Somayaji, "Computer Immunology", *Communications of the ACM*, vol. 40 no. 10, pp. 88-96, November 1997.

[7] W. Lee and S. Stolfo. "Data Mining Approaches for Intrusion Detection", in *Proceedings 1998 7th USENIX Security Symposium*, January, 1998.

[8] Biswanath L. Mukherjee, Todd Heberlein, and Karl N. Levitt, "Network Intrusion Detection", *IEEE Network*, vol. 8 no. 3, pp. 26-41, May/June 1994.

[9] Hyacinth S. Nwana, "Software Agents: An Overview", *Knowledge Engineering Review*, vol. 11 no. 3, pp. 205-244, October/November 1996.

[10] Phillip A. Porras and Alfonso Valdes, "Live Traffic Analysis of TCP/IP Gateways," in *Networks and Distributed Systems Security Symposium*, March 1998.

[11] S. Stolfo, A. Prodromidis, S. Tselepis, W. Lee, D. Fan, and P. Chan, "JAM: Java Agents for Meta-learning over Distributed Databases", in *AAAI97 Workshop on AI Methods in Fraud and Risk Management*.

[12] Leendert van Doorn, `nfsbug.c`, available online at `http://www.asmodeus.com/archive/Xnix/nfsbug/nfsbug.c`, 1994.

[13] Jihoon Yang, and Vasant Honavar, "Feature Subset Selection Using a Genetic Algorithm", *IEEE Intelligent Systems Special Issue: Feature Transformation and Subset Selection*, 1998.