Automated Intrusion Detection Methods Using NFR

Wenke Lee Christopher T. Park Salvatore J. Stolfo
Computer Science Department, Columbia University
500 West 120th Street, New York, NY 10027
{wenke,cpark,sal}@cs.columbia.edu

Abstract

There is often the need to update an installed Intrusion Detection System (IDS) due to new attack methods or upgraded computing environments. Since many current IDSs are constructed by manual encoding of expert security knowledge, changes to IDSs are expensive and require many hours of programming and debugging. We describe a data mining framework for adaptively building Intrusion Detection (ID) models specifically for the use of in Network Flight Recorder (NFR) [10]. The central idea is to utilize auditing programs to extract an extensive set of features that describe each network connection or host session, and apply data mining programs to learn rules that accurately capture the behavior of intrusions and normal activities. These rules can then be used for misuse detection and anomaly detection. Detection models are then incorporated into NFR through a machine translator, which produces a working detection model in the form of N-Code, NFR's powerful filtering language.

1 Motivation

Intrusion detection is often used as another wall to protect computer systems, in addition to the standard methods of security measures such as user authentication (e.g. user passwords or biometrics), avoiding programming errors, and information protection (e.g., encryption). Intrusion detection techniques can be categorized into *anomaly detection* and *misuse detection*. Anomaly detection systems, for example, IDES [7], flag observed activities (i.e., possible intrusions) that deviate significantly from the established normal (statistical-based) usage profile as anomalies. Misuse detection systems, for example, IDIOT [5] and STAT [4], use patterns of known attacks or weak spots of a system to match and identify intrusions. While accuracy is the essential requirement, its extensibility and adaptability are also critical design criteria in today's network computing environment. There are multiple "penetration points" for intrusions to take place in a network system. For example, at the network level, carefully crafted "malicious" IP packets can crash a victim host; at the host level, vulnerabilities in system software can be exploited to yield an illegal root shell. Since activities at different penetration points are normally recorded in different audit data sources, an

IDS often needs to be extended to incorporate (additional) modules that specialize on certain components (e.g., hosts, subnets, etc.) of a network systems. The large traffic volume in security related mailing lists and Web sites suggest that new system security holes and intrusion methods are continously being discovered. Therefore, it is imperative that IDSs be updated frequently and rapidly.

Currently building an effective IDS is an enormous knowledge engineering task. System builders largely rely on their intuition and experience to select the statistical measures for anomaly detection [8]. Many IDSs only handle one particular audit data source, and updating these systems is expensive and slow . Some of the recent research and commercial IDSs have begun to provide built-in mechanisms for customization and extension. The Network Flight Recorder (NFR) is one such extensible system that combines data collection and analysis and storage within a single platform. More than likely, systems such as these would be located between a firewall and an Internet connection, an area aptly named the DMZ [**?**] Analysis in NFR is accomplished by scripts based on a language called N-code, NFR's flexible language for traffic analysis. Information is displayed in NFR to a web-based interface with Java support. One of NFR's key features is that it does not interfere with network activity, which is a necessary design criteria to obtain accurate data for analysis. NFR also has a real time alerting capability and a storage subsystem that allows data to be stored, rotated, and archived to other external devices [10]. However, this does not eliminate the need experts to first analyze and categorize attack scenarios and system vulnerabilities, and hand-code the corresponding rules and patterns in N-code for misuse detection. Because of the manual and ad hoc nature of the development process, current IDSs including NFR have limited extensibility and adaptability. Our goal is to substantially reduce this effort by automating:

1) the task of intrusion detection through data-mining.

2) generating the N-code for NFR to monitor such intrusions via machine translator.

While using such methods, system builders and administrators will still have to maintain and fine-tune the respective IDS. However a large amount of their work will be automated, thus effectively reducing time and man-power costs in fielding an effective IDS.

2 Data-Mining and Meta-Learning Overview

In this section we provide an overview of our data-mining algorithms. Data mining generally refers to the process of (automatically) extracting models from large stores of data [3]. The recent rapid development in data mining has made available a wide variety of algorithms, drawn from the fields of statistics, pattern recognition, machine learning, and databases. Several types of algorithms are particularly useful for mining audit data.

Classification : maps a data item into one of several pre-defined categories. These algorithms normally output "classifiers", for example, in the form of decision trees or rules. An ideal application in intrusion detection will be to gather sufficient "normal" and "abnormal" audit data for a user or program, then apply a classification algorithm to learn a classifier that can label or predict new unseen audit data as belonging to the normal class or the abnormal class.

Link analysis : determines relations between fields in database records. Correlations of system features in audit data can serve as the basis for constructing normal usage profiles. A programmer would have "emacs" highly associated with "C" files, for example. Observed deviations from these automatically learned associations may suggest an attack.

Sequence analysis : models sequential patterns. These algorithms can discover what (time-based) sequence of audit events frequently occur together. These frequent event patterns provide guidelines for incorporating temporal statistical measures into intrusion detection models.

A framework has been developed, first proposed in [6], of applying data mining techniques to build intrusion detection models. This framework consists of programs for learning classifiers (and meta-classifiers [2]), association rules [1] (for link analysis) and frequent episodes [9] (for sequence analysis), as well as a support environment that enables system builders to interactively and iteratively drive the process of constructing and evaluating improved detection models. The end product of this process is a set of concise and intuitive rules (that can be easily inspected and edited by security experts when needed) that can detect intrusions. The rules are then subsequently compiled over to N-code by a translator using various modules and sub-routines we have written.

In order for data-mining programs to compute effective intrusion detection models, we first process and summarize packet-level network traffic data into "connection" records. Each record has an extensive set of features. By examining connections in the past n seconds (e.g. 2 seconds), we include features describing connections that have the *same destination* host as the current connection (Table 1) and include features describing connections that use the *same service* as the current connection (Table 2).

For each connection, we also include the features described in Table 3.

An N-code filter has been written for each of these attributes. It is best to think of these terms as sub-routines or modules that the data-mining process can then call upon when it deems necessary for building a new model.

short_count	the count of such connections
short_rej_count	the count of connections that get the flag“REJ” (i.e. a packet that has a flag ”S” which is met by an ”R” packet from the receiving end)
short_s01_count	the count of connections that send a S packet but never get the ack packet (s0), or receive an ack on S that they never have sent (s1)
short_diff_services	the count of unique (different) services
short_diff_srv_rate	short_diff_services / short_count

Table 1: Same Destination

srv_short_count	the count of such connections
srv_short_diff_hosts	the count of unique (different) destination hosts
srv_short_diff_rate	srv_short_diff_hosts / srv_short_count

Table 2: Same Service

3 Implementation

Here we provide several example rules used to detect known attacks. In particular, we illustrate how to detect and recognize an attack categorized as “denial-of-service”, (e.g. teardrop, smurf, ping-of-death).

A “training” period is initially required for the algorithms to gather the necessary training data on the network to compute models. The purpose is two-fold:

1) establishing “normal” traffic patterns and variants that the network may encounter to establish anomaly detection,

2) introducing known intrusion methods and attack scripts into the network in order for the data-mining process to establish sufficient data for classification of intrusions

Once normal network traffic and attacks have generated sufficient audit data, scripts, the data-mining algorithms produce rules that are characteristic of each intrusion. This tool, saves the System Builder hours of labor intensive effort to find a pattern for each intrusion.

Let us suppose that a hacker launches “teardrop” from machine im.a.hacker.com is attempting to bring

duration	the length (in seconds) of the connection
protocol_type	type of protocol being used
protocol	if the protocol is privileged ($\geq$ 1024) or not
flag	normally SF (successfully connected and terminated according to the protocols, but can be REJ, S0, or S1
urgent	is the ”urgent” flag used in any of the data
wrong_size_rate	if wrong fragmentation, the rate $\geq$ 1 packet?

Table 3: Other Features

down the server your.machine.org. Teardrop transmits a number of overlapping fragmented UDP packets to a specified host. Since UDP is an unreliable protocol, no acknowledgment would be needed, therefore the tcpdump data file would record the following events:

```
...
...
12:22:18.336681 im.a.hacker.com.2019 > your.machine.org.talk: udp 28 (frag 242:36@0+)
12:22:18.336681 im.a.hacker.com > your.machine.org: (frag 242:4@24)
12:22:18.356681 im.a.hacker.com.2019 > your.machine.org.talk: udp 28 (frag 242:36@0+)
12:22:18.356681 im.a.hacker.com > your.machine.org: (frag 242:4@24)
12:22:18.376681 im.a.hacker.com.2019 > your.machine.org.talk: udp 28 (frag 242:36@0+)
12:22:18.376681 im.a.hacker.com > your.machine.org: (frag 242:4@24)
...
...
```

Here we know the intrusion and approximately its start time, source port and destination port, we use this knowledge to "label" the connection records and train our model to "recognize or classify" this type of intrusion. Thus, applying our data-mining algorithms, the IDS generates a rule referred to in Table 4 for monitoring the TEARDROP intrusion.

Rule	**Translation**
if (protocol = private and wrong_size_rate $\geq$ 1)	(the connection is on a privileged port $\leq$ 1024 and fragmentation is wrong)

Table 4: Tear Drop Rule

While this may seem intuitive to a System Builder, it is important to distinguish that the IDS has *automated* the process of generating such heuristics. Our system would then call upon the machine translator to compile the teardrop rule, and generate from the sub-routines defined earlier, the appropriate N-code that filters out all network traffic except for the information concerning this particular type of attack. Recording the data onto disk proves to be an easy task as NFR has provided a thorough and complete resource package that is capable of accessing the data in terms of a sequential database [10]. The types of intrusions that we are primarily concerned with fall into 4 main categories:

1) denial-of-service (e.g., ping-of-death, SYN flood, smurf, teardrop, etc)

2) unauthorized access from a remote machine (e.g./ guessing password)

3) unauthorized access to local superuser privileges by a local unprivileged user (e.g., various buffer overflow attacks)

4) surveillance and probing (e.g., port-scan, IPscan,etc)

It makes sense to have these four different types of alerts and ways to access the data, each unique in the frequency in which they occur. These alerts and triggers would then simply be concatenated to the end of the N-code generated by the machine translator.

4 Conclusions

In this paper, we have outlined a data mining framework integrated with an existing IDS, namely NFR. The key idea is to apply data mining algorithms to compute and generate misuse and anomaly detection models, according to the observed behavior in audit data.

We extend the basic association rules and frequent episodes algorithms to accommodate the special requirements of analyzing audit data. Our experiments show that the frequent patterns mined from audit data can be used as reliable anomaly detection models, and as guidelines for selecting temporal statistical features to build effect N-code filters.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, 1993.

[2] P. K. Chan and S. J. Stolfo. Toward parallel and distributed learning by meta-learning. In *AAAI Workshop in Knowledge Discovery in Databases*, pages 227–240, 1993.

[3] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process of extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, November 1996.

[4] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[5] S. Kumar and E. H. Spafford. A software architecture to support misuse intrusion detection. In *Proceedings of the 18th National Information Security Conference*, pages 194–204, 1995.

[6] W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, January 1998.

[7] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Garvey. A real-time intrusion detection expert system (IDES) - final technical report. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

[8] Teresa F. Lunt. Detecting intruders in computer systems. In *Proceedings of the 1993 Conference on Auditing and Computer Technology*, 1993.

[9] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the 1st International Conference on Knowledge Discovery in Databases and Data Mining*, Montreal, Canada, August 1995.

[10] Inc. Network Flight Recorder. Network flight recorder. http://www.nfr.com, 1997.