Journal of Computer Science 8 (10): 1649-1659, 2012
ISSN 1549-3636
© 2012 Science Publications

Evaluating the Strengths and Weaknesses of Mining Audit Data for Automated Models for Intrusion Detection in *Tcpdump* and Basic Security Module Data

[1]Mohammed Nazer, G. and [2]A. Arul Lawrence Selvakumar
[1]Department of Computer Applications,
IFET College of Engineering, Villupuram, India
[2]Department of Computer Applications,
Adhiparasakthi Engineering College, Melmaruvathur, Tamilnadu, India

Abstract: Problem statement: Intrusion Detection System (IDS) have become an important component of infrastructure protection mechanism to secure the current and emerging networks, its services and applications by detecting, alerting and taking necessary actions against the malicious activities. The network size, technology diversities and security policies make networks more challenging and hence there is a requirement for IDS which should be very accurate, adaptive, extensible and more reliable. Although there exists the novel framework for this requirement namely Mining Audit Data for Automated Models for Intrusion Detection (MADAM ID), it is having some performance shortfalls in processing the audit data. **Approach:** Few experiments were conducted on *tcpdump* data of DARPA and BCM audit files by applying the algorithms and tools of MADAM ID in the processing of audit data, mine patterns, construct features and build RIPPER classifiers. By putting it all together, four main categories of attacks namely DOS, R2L, U2R and PROBING attacks were simulated. **Results:** This study outlines the experimentation results of MADAM ID in testing the DARPA and BSM data on a simulated network environment. **Conclusion:** The strengths and weakness of MADAM ID has been identified thru the experiments conducted on *tcpdump* data and also on *Pascal* based audit files of Basic Security Module (BSM). This study also gives some additional directions about the future applications of MADAM ID.

Key words: Feature construction, data mining, intrusion detection, denial of service, network security

INTRODUCTION

There was clear evidence from many studies (for example (Durst *et al*., 1999), that the insiders, who have not blocked by firewalls, are the causes for computer security incidents. At the same time, the intruders, so called legitimate users require access with significant privileges to do their day to day work. Moreover, the vast majority of the harm from the insiders are not malicious, rather it is honest people make some honest mistakes. However, there are so many potential outsiders who are very clever and have somehow passed all the screens of firewalls and access and authorization controls and do malicious activities, especially in a network environment. Then, how to prevent them? Although, prevention is very much necessary, it is not a complete solution for computer security. Moreover, it is not practically possible to detect such harmful incidents in advance. Many surveys have been done to control the intrusions and (Halme and Bauer, 1995) identified various range of controls to address intrusion detection.

All these preventive controls can be complimented as the next line of defense, an Id. This intrusion detection system acts as a separate spy computer in a network environment to monitor all the users and system activities, audit the system configuration for vulnerabilities, misconfigurations, accessing the system integrity and data files, recognizing the known attack patterns, violation of user access policy and much more functions. In case if it detects any harmful or suspicious activities, it will alarm the system administrator immediately to take necessary action.

Since the technology has been improved significantly, the modern IDs operates on real time and these ATIDS-Automated IDS monitor all the activities of the user, system and network and alarm the administrator in case if any malicious or suspicious event occurs. Ideally an IDs should be fast, simple, complete and more accurate. This is because, in the

Corresponding Author: Mohammed Nazer, Department of Computer Applications, IFET College of Engineering, Villupuram, India

initial stages, there were a huge false alarm signals and very little positive alarm signals. On attending these signals, there were so much of resources has been applied and much time has been wasted in attending the false or negative alarm signals. Modern commercial IDs tends to be more accurate. But these IDs, detects all known and unknown attacks with limited performance penalty.

Monitoring the use and system activities is appropriate for an attack of initial impact. Indeed, the actual goal of an ID is to check, what resources are being accessed and various attempted attacks are tried. Moreover, recording all the traffic of a given source or destination is very much useful for future audit analysis. This type of approach should be invisible to the user. Finally, IDS should respond an initial defensive action immediately while generating an alarm to the administrator, who can act, only upon receiving an alarm, which takes some time.

Many research works are still in progress on the evolving product of IDS, which has started from the early 1990s. Recent researches (Dickerson and Dickerson, 2000) reveal that, IDS detect a number of serious problems, which are even growing and as the number of problems or attacks increases, so do the signature patterns to the IDS model. Thus, modern IDS are improving in defending continuously. On the other side, avoiding IDS are the first and prominent priority for a number of successful intruders. As we all say that AN ID that is not well defended is useless. Another boom in the IDS technology is that the stealth mode IDS, which is very difficult, even to find on an internal network and is left alone to compromise by itself.

In today's network environment, even though, accuracy of IDS is the essential requirement, its extensibility and adaptability are also very much critical. In a network environment, there exist multiple penetration points for the attackers. For example, in a network level, a well designed malicious IP packets penetrate even through the firewalls and crash the victim host, as well as, at the host side, more vulnerabilities in system software can be exploited to yield an illegal root shell. Since activities at different penetration points are recorded in different audit data sources, an IDS often needs to be extended to incorporate additional modules that specialize in certain components, such as hosts, subnets, etc. of the network system.

Snort, which is another milestone in IDS technology, is a signature based lightweight open source network intrusion prevention and NIDS-Network IDS. This captures and analyze whether there exist a pattern that matches a known signature inside the packet content. Snort has been designed with flexible rules to describe the network traffic to identify which packets to collect or to pass and with a modular plug-in structured detection engine.

A real time alerting capability and generating logs when an attack occurs are the major credits of this Snort. Snort can be distributed to different parts of the network and can send alerts to the central console. Snort's network interface card runs in promiscuous node, which captures all the network traffic that goes by NIC and detect the unexpected events in the traffic to generate real time alerts to the central console.

As a next step, a system for automated network intrusion detection is in progress as a part of JAM project. This ANIDS -Automated Network Intrusion Detection System is designed with many data mining methods, to build network intrusion classifiers which are used to monitor live network stream input to detect the intrusions.

This study discusses about our experiments on the audit data files for building intrusion detection models from the DARPA and Basic Security Module (BSM) Intrusion detection evaluation program and the security related problems. We obtained a set of *tcpdump* data, available at http://iris.cs.uml.edu:8080/network.html. Even though, the output of *tcpdump* data is not intended for security purposes, we had to go through multiple iterations of data pre-processing to extract meaningful features and measures. We studied TCP/IP and its security related problems, for example (Stevens, 1994; Paxson, 1997; 1998; Atkins, 1996; Bellovin, 1989; Porras and Valdes, 1998), for the guidelines of protocol features.

This study is organized as follows. We first give a brief overview of our experiments on *tcpdump* data. We then outline construction of manual and automatic features along with various detection models applied and the performance results of *tcpdump* data. In the last section, we give a brief overview of our experiments on BCM data and their results.

MATERIALS AND METHODS

In this study, we describe in detail about our experiments on DARPA and BSM audit data files for building intrusion detection models. We have applied the classification rules, link analysis and sequence analysis algorithms that has been discussed in (Nazer and Selvakumar, 2011) and we also applied the tools of Mining Audit Data for Automated Models (MADAM) to process the audit data, mine patterns and construction features that has been discussed in (Nazer and Selvakumar, 2012a) and (Nazer and Selvakumar, 2012b) in our simulated network environment. In these experiments on *tcpdump* and BSM data, the strengths and weakness of MADAM ID has been identified and illustrated with their performance results.

Experiments on *tcpdump* data and their results: In order to test the effectiveness of data mining techniques in IDS (Abraham, 2001), we took the user of established and more appropriate data sets and these data sets are more popular and widely used for research work at MIT Lincoln Laboratory @ http://www//ll/mit-edu/IST/ideval. They have collected and distributed the first version of standard corporation for evaluation of network intrusion detection systems. In this evaluation, the probability of detection (whether it detects all intrusions or known attacks) and the probability of false alarm are measured for each system under simulated or testing environments. The objective of these experiments is to study and to analyze the performance shortfalls in the intrusion detection research work.

All these experiments were done on the training data set provided at Lincoln Laboratory of Massachusetts Institute of Technology (DARPA Intrusion Detection Evaluation Dataset) and these are available at 'http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998 data.html'. We have collected 4 gigabytes of compressed *tcpdump* data of 7 weeks of network traffic, for these experiments. These data can be processed into 5 million connection records of 100 bytes each. Each of this data contains the data portion of every ICMP packet transmitted between host inside and outside the simulated net work environment. For testing, we considered and simulated DOS, R2L, U2R and PROBING attack types and in addition to that there was anomaly user behaviour (Ghosh *et al*., 1999), which means a normal user acts as a system administrator with full authentication privileges.

Before commencing our experiments, certain data preprocessing was done, so that, for data packet filtering and reassembling work, we used Bro tool (Paxson, 1998). In our case, in order to avoid the system crashing in ping-of-death and teardrop attacks, we have made new changes in the ICMP packets to its packet fragment inspection modules. This change include a Bro-connection finished event handler so that we get a summarized record for each connection and these records have intrinsic features which are described in the following table (Table 1).

The above table lists the various intrinsic features of network connection records.

Misuse detection: The 'list files' which are included in the training data files were used to identify type of attack, source and destination host and port id and also the timestamp of the files. For building the classification model, we used these information for pattern mining, feature construction and to name each correction record with 'normal' and an attack type to create training data. For our testing purpose, we did not aggregate all the connection records, since the amount of audit data is really very huge, instead, we considered only those connection records that fall within a surrounding time of plus and minus 5 min of each attack. Similarly, we created a dataset for each attack type and for normal dataset, we aggregated only the sequences of normal connection records.

Construction of manual and automatic features: When each ICMP packet data is summarized into the connection records (Nazer and Selvakumar, 2012b) using commonly available packet processing engines, each network connection record contains a set of 'intrinsic' features that are for general network traffic analysis purposes. These features include *service*, *src_host*, *src_port*, *dst_host*, *duration* (duration of the connection), *src_bytes* and *dst_bytes* (number of data bytes from source to destination and vice versa) etc. These intrinsic features were shown in the above mentioned table (Table 1). The frequent sequential patterns from these initial connection records can be viewed as statistical summaries of the network activities. For each attack type, e.g., syn flood, port-scan, we performed pattern mining and comparison using its intrusion data set and the normal data set. But for each attack method, the actual network hosts are irrelevant and moreover there were over a thousand different hosts in the *tcpdump* training data. Hence we did post-processing work on the frequent patterns of each record before we do encode and compare on the training data. The post processing work was done with the following procedure.

In each dataset, check a frequent pattern from left to right, one by one as follows:

- Let the first *src_host* value be s_0
- Let the first *dst_host* value by d_0

In each dataset, whenever a *src_host* value is identified, check whether it is the same as one of the previous *src_host* in the pattern:

- If yes, then replace it with the appropriate s_i
- Otherwise replace it with s_{n+1}
- Perform the same process for the *dst_host* value

For example, a pre-processing dataset pattern:

($service = http$, $src\_host = host_A$),
($service = telnet$, $dst\_host = host_B$) $\rightarrow$
($service = smtp$, $src\_host = host_C$ $dst\_host = host_B$), [0.2,0.1,2s]
The above data is post-processed into
($service = http$, $src\_host = s_0$),
($service = telnet$, $dst\_host = d_0$) $\rightarrow$
($service = smtp$, $src\_host = s_1$, $dst\_host = d_0$), [0.2,0.1,2s]

Table 1: Intrinsic features of network connection records

Feature	Description	Value type
Duration	Length of the connection (number of seconds)	Continuous
Protocol_type	Type of protocol, e.g., tcp, udp,	Discrete
Service	Network service on the destination, eg., http, telnet	Discrete
Src_bytes	Number of data bytes from source to destination	Continuous
Dst_bytes	Number of data bytes from destination to source	Continuous
Flag Normal or error status of the connection	Discrete	
Land	1- connection is from/to the same host/port; 0 - otherwise	Discrete
Wrong-fragment	Number of wrong fragments	Continuous
Urgent	Number of urgent packets	Continuous

As a result of post-processing, the redundancy patterns are reduced and this means, the number of unique patterns within a pattern set is significantly reduced. Moreover, the process of creating a normal pattern set, pattern encoding and pattern comparison becomes very much efficient and all these processes are possible only because of the post processing. In this process, we have created two features namely 'same host' (*same_host*) and 'same service' (*same_srv*) for intrusion only patterns of each attack type. These two intrinsic features are explained as follows:

- The '*same_host*' feature examines only the connections in the past 2 sec that have the same destination host as the current connection record
- The '*same_srv*' feature examines only those connections in the past 2 sec that have same services as the current connection record

We finally summarize the statistical features that are automatically constructed in this process. The statistical feature includes:

- Count of *same_host* and *same_srv* connections
- Percentage of connections having *same_srv* as the current one
- Percentage of different services
- Percentage of different destination hosts
- Percentage of *Serror_%* and *Rerror_%*

These time-based 'traffic' features of connection records are summarized in the Table 2.

Out of all the four attack types that we considered namely DOS, R2L, U2R and PROBING, only the PROBING is very slow that did not produce intrusion-only patterns within the specific time of 2 sec, to say, it can scan the host or the port in a time span of more than a minute. In order to create a 'host based' traffic features, these connection records were sorted based on the destination hosts and applied same pattern mining and feature construction process. Similar to the time based traffic features, we constructed a mirror set of host based traffic feature by using a 'connection' window of 100 connections instead of time span of 2 sec. The R2L and U2R attacks don't have any intrusion only frequent patterns as found in most of the DOS and PROBING attacks. These DOS and PROBING attacks involve a lot of connection to some hosts or ports in a very short period of time and hence they can have more frequent sequential patterns than the normal traffic pattern. In case of R2L and U2R attacks, these are encapsulated within the data portion of ICMP packets which generally appear in a single connection.

Our automatic feature construction model would fail to produce any model or features for these types of attacks since because these attacks don't have any unique frequent patterns of traffic. Also in case of unstructured data contents of IP packets, our current data mining algorithm cannot deal and hence we consider the domain knowledge for defining appropriate current features. To inspect the data exchanges of interactive TCP connection, such as *ftp*, *smtp*, we added some more functions that assign values to a set of content features in order to identify any suspicious behaviour inside the packet data contents. The various features are listed in the following table (Table 3).

The statistical features include number of hot indicators, number of failed login attempts, successful logins, number of compromised conditions, whether root shell is obtained or not, whether a *su* command is attempted and successful or not, number of file creations, number of shell prompts, number of write, delete and create operations on access control files, number of outbound commands in a *ftp* session, root or admin logins or a guest login status. With so much of statistical indicators, the classification program can decide, which minimal set of discriminating features can be used in order to identify the instructions and this is the basic idea behind Table 3.

Table 2: Network traffic features of network connection records

Feature	Description	Value type
Count	Number of connections to the same host as the current connection in the past 2 sec	Continuous
The following features refer to these same-host connections		
Serror_%	% of connections that have ‘SYN’ errors	Continuous
Rerror_%	% of connections that have ‘REJ’ errors	Continuous
Same_srv_%	% of connections to the same service	Continuous
Diff_srv_%	% of connections to the different services	Continuous
Srv_count	Number of connections to the same service as the current connection in the past 2 sec	Continuous
The following features refer to these same-service connections		
Srv_serror_%	% of connections that have ‘SYN’ errors	Continuous
Srv_rerror_%	% of connections that have ‘REJ’ errors	Continuous
Srv_diff_host_%	% of connections at different hosts	Continuous

Table 3: Content features of network connection records

Feature	Description	Value type
hot	Number of ‘hot indicators’	Continuous
failed_logins	Number of failed login attempts	Continuous
logged_in	1- successful login; 0 - otherwise	Discrete
compromised	Number of ‘compromised’ conditions	Continuous
root_shell	1- root shell is obtained; 0 - otherwise	Discrete
su 1 -‘su root’ command attempted; 0-otherwise	Discrete	
file_creations	Number of file creation operations	Continuous
shells	Number of shell prompts	Continuous
access_files	Number of write, delete and create operations on access control files	Continuous
outbound_cmds	Number of outbound commands in a ftp session	Continuous
hot_login	1- the login belongs to the ‘hot’ list (e.g., *root*, *admin*, 0 - otherwise	Discrete
guest_login	1 - the ‘guest’ login; 0 - otherwise	Discrete

Table 4: Example of ‘traffic’ connection records

Label	Service	Flag	Count.	Srv_count	Rerror_%	Diff_srv_%
Normal	ecr_i	SF	1	1	0	1
Smurf	ecr_i	SF	350	350	0	0
Satan	user-level	REJ	231	1	85%	89%
Normal	http	SF	1	0	0	1

Table 5: Example of RIPPER Classifier for DOS and PROBING Attacks

RIPPER rule	Description
smurf :- count = 5, srv_count >= 5, service = ecr_i the same destination host as the current one) is 5 and the number of connections that has the same service as the current one is at least 5, then this is a smurf type of DOS attack.	If the service is ecr_i (icmp echo request) and for the past 2 sec, if the number of connections (that has
satan :- rerror_% >= 83%, diff_srv_% >= 87%. different services is at least 87%, then this of the rejected connection is at least 83%, and the % of is a satan type of PROBING attack	For the past 2 sec, if the number of connections have the same destination host as the current connection, the %

Different detection models: Since different types of intrusion requires different construction features to detect them, we have created three classification models, each of which will be using different set of construction features and these models are explained below:

- Traffic model
- Host-based traffic model
- Content model

The ‘Traffic’ model: In this model, each connection record contains the ‘traffic’ and ‘intrinsic’ features as shown in the following table. Table 4 shows the example labeled connection records.

The appropriate RIPPER classifier detects the DOS and PROBING attacks and the following table (Table 5) shows such an example.

The ‘Host-Based Traffic’ model: In this model, each connection record contains the ‘intrinsic’ and the ‘host-based traffic’ features and the resultant RIPPER classifiers detect the slow PROBING attacks.

The ‘Content’ model: In this model, each connection record contains the ‘intrinsic’ and the ‘content’ features of each ICMP packets and the resultant RIPPER classifier detects the R2L and U2R type of attacks all the above mentioned classification models detects a specific type of intrusion. Instead of having different models individually, we combine all these three classification models into a meta classifier. The advantage of this meta classifier is that each meta level audit record contains the three predictions from the traffic, host-based and content models and additionally one more information of true class label which means ‘normal’ and an attack type. In order to identify whether a connection is normal or an intrusion type, we apply RIPPER rules, so that to detect the R2L and U2R attack type, the meta level classifier uses the content model and to detect the DOS and PROBING attacks, the meta level classifier uses the combination of the traffic and host-based traffic models.

RESULTS AND DISCUSSION

The training audit data were provided by Lincoln Laboratory of Massachusetts Institute of Technology at ‘http://www.ll.mit.edu/mission/communications/ist /corpora /ideval/data/1998data.html’. These *tcpdump* data is about 7 weeks of network traffic and took 2 weeks of unlabelled test data for our experiment. The test data were having so many attack types and we considered 14 types in test data only since our models were not trained to detect of all attack types. These ere reported in the following figure (Fig. 1).

The above figure shows the performance of *tcpdump* misuse detection models and the ROC curves on detection rates and false alarm rates, on all four attack types such as DOS, PROBING, U2R and R2L. The x-axis represents the false alarm rate and the y-axis represents the detection rate. The false alarm rate is calculated as the percentage of normal connections that are classified as an intrusion. The upper left corner data print on each ROC curve shows the low false alarm rate with high detection rate. Group 1 to 3 ROC curves represent the performance of an intrusion detection by other knowledge engineering models.

From the above figure, we can see that our detection model has the best overall performance on detecting intrusion attacks. However, in the case of R2L attacks, all models performed very poorly. The features we built would be general enough so that the models can detect new variations of the known attacks and the new attack refer to those that did not have corresponding instances of our trained data. Moreover, our model can handle a large percentage of PROBING and U2R attacks when compared to DOS and R2L intrusions.

Experiments on BSM data and their results-The Basic Security Module (BSM) (Sunsoft, 1995) audit data were provided by DARPA for a particular host, *pascal*. With this data, we did some experiments in building host-based intrusion detection model. In a host machine, when BSM data is enabled, we get time-bounded sequence of actions that are audited on the system which contains one or more *audit files*.

Each record in the audit file may contain a kernel event such as a system call or a user-level event which is nothing but a system program. *Audit session* is the collection of incoming or outgoing sessions on a particular host such as terminal login, *telnet* login, rlogin (remote login), *rsh*, *ftp* and *sendmail*.

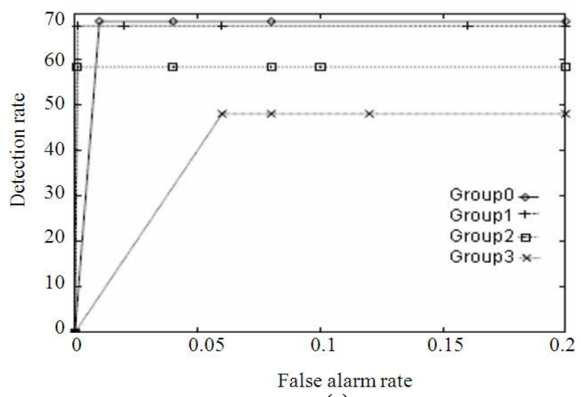
Data Preprocessing-We need to perform a sequence of data preprocessing tasks on the BSM data. Since the BSM data is in the form of binary, it has to be converted into ASCII data and hence we further extended the preprocessing component of USTST (Ilgun, 1993). The following table (Table 6) represents some example BSM event records.

Table 7 shows example of BSM event records and a ‘?’ refers that the value is not given in the original audit record itself. In this each audit record contains various basic features and these features are shown in the following table.

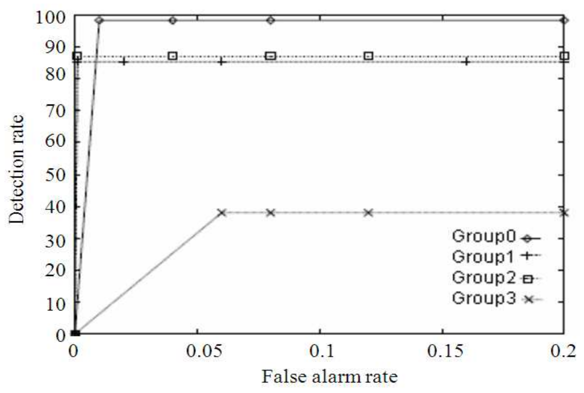
We have created a procedure to process the event data and convert into session records and the procedure is constructed as follows:

- In the beginning of a audit session, we execute
- The *inetd_connect* event (for telnet, rlogin, rsh) or
- The *execve* event on a system program *in.fingerd* (for incoming finger request) or *finger* (outgoing), *mail.loca*l (incoming) or *sendmail* (outgoing), *ftpd* (incoming) or *ftp* (outgoing)
- We record the setaudit event, which assigns the *auid* and *sid* of the session
- We examine all audit records that share the same combination of *auid* and *sid* to consolidate a number of session features
- Finally record the session termination

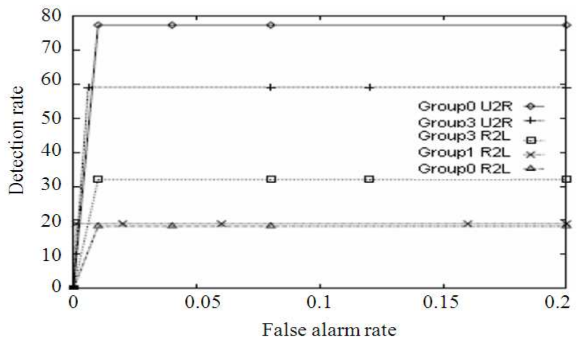
Session features: The various tests with feature construction for session records were analyzed as well and as the first step we computed the frequent patterns from the BSM audit event records. For pattern mining, we prepared the data set in such a way that it contains all the accountable event records of a particular session. The word ‘accountable’ here means an audit record having a meaningful audit user id and a valid session id. From these data sets, we have removed audit user id and session id in order to get a generalized data set and this is because as such the data set is session specific. Then we replaced *ruid* and *euid* features with a flag *same_reid* so that *ruid* agrees with *euid*.



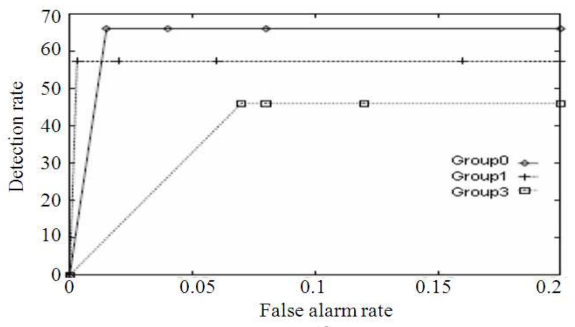
(a)



(b)



(c)



(d)

Fig. 1: Performance of *tcpdump* misuse detection models (a) DOS (b) BROBING (c) U2R and R2L (d) Over all

Table 6: Example records of BSM Event Records

Time	Auid	Sid	Event	Pid	Obname	Euid
08:05:22	0	0	Inet_connect	0	?	0
08:05:22	-2	0	Execve	415	/usr/bin/	0
08:05:31	2104	417	Setaudit	417	?	0
08:05:31	2104	417	Chdir	418	/home/tristank	2104

Table 7: Features of BSM Event Records

Features	Description	Value
Time	Timestamp of the event	Discrete
Auid	Audit user id, inherited by all child processes started by the user's initial process of a session	Discrete
Sid	Audit session id, assigned for each login session and inherited by all dependent processes	Discrete
Event	Audit event name	Discrete
Pid	Process id of the event	Discrete
Obname	Object name, that is full file path that the event operates on	Discrete
Argl_arg4	Arguments of the system call	Discrete
Text	Short event information (e.g., successful login)	Discrete
Error_status	Error status of the event	Discrete
Return_value	Return value of a system call event	Discrete
Tmid	Terminal id	Discrete
Ip_header	Source and destination ip address and ports of the socket used by the event	Discrete
Socket	The local and remote ip addresses and ports of the socket used by the event	Discrete
Ruid	The real user id of the event	Discrete
Rgid	The real group id of the event	Discrete
Euid	Effective user id of the event	Discrete
Egid	The effective group id of the event	Discrete

Table 8: Features for BSM session records

Feature	Description	Value type
Duration	Length (number of seconds) of the session	Continuous
Service	Operating system or network service	Discrete
Logged_in	Whether the user successfully logged in	Discrete
Failed_logins	Number of failed login attempts	Continuous
Process_count	Number of processes in the session	Continuous
Suid_sh	Whether a shell is executed in suid state	Discrete
Suid_p	Whether a suid system program is executed	Discrete
User_p	Whether a user program is executed	Discrete
Su_attempted	Whether a su command is issued	Discrete
Access_files	Number of write, delete and create operations on access control files	Continuous
File_creations	Number of file creations	Continuous
Hot_login	Whether the login belongs to the 'hot' list	Discrete
Guest_login	Whether the login belongs to the 'guest' list	Discrete

Table 9: Example records of BSM Session

Label	Service	Suid_sh	Suid_p	User_p	File_creations
Normal	Smtp	0	0	0	0
Normal	Telnet	0	1	1	3
Normal	Telnet	0	1	0	0
Buffer_overflow	Telnet	1	1	1	2
Normal	Ftp	0	0	0	0
Wraz_master	Ftp	0	0	0	42

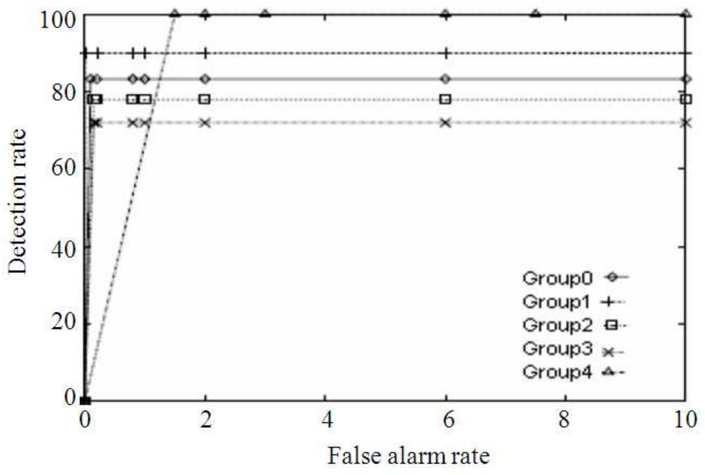
Since *axis* is very important attribute to describe an event data, we represent event as the axis attribute. To compute the number of occurrences of each unique event and the object name, we used relative support of 0.1, so that the patterns of frequent occurrences of object names can be captured using the relative support 0.1. On further proceeding in our testing, we identified the frequent patterns are related a specific object names. For example, although an object name of */usr/bin/nazer* may appear only once or twice in the dataset for a session, we can still identify its patterns using the relative support of 0.1, these occurrences are all of frequent patterns. After a few rounds of initial experiments, we discovered that the patterns are all related to very specific object name or event values. But there are many system calls (kernel events) which cannot be directly linked to user-level commands and hence we reasoned that for intrusion detection purposes, we only need to analyze user-level commands and their operations. For this purpose, we kept only the read, write, create, delete, execute, change ownership, permission, rename and link event records.

The event value of each audit record is replaced by the appropriate type name, for example, *open_r* is replaced by *read* event. We kept only the original object name if the event is *execute*, otherwise we used 'user' to replace all object name values that indicate files in the user directories and 'system' to replace the object name values that indicate files in the system directory. We have also removed '?' (missing) object name values as well and finally we aggregated all event patterns of all normal sessions into a normal pattern set.

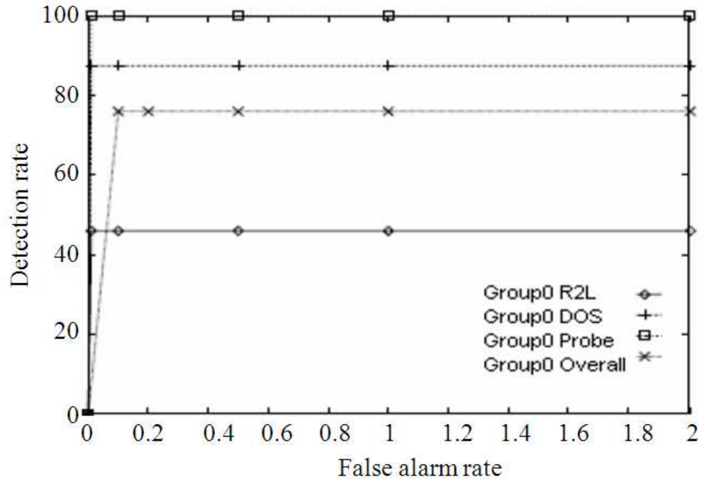
For each U2R session, we further mined its event patterns and compared with the normal pattern set. On encoding, we used the *same_reid*, *event*, *obname* and rest in alphabetical order. On applying the pattern encoding and comparison procedures, we received the top 20% of intrusions-only patterns for each U2R attack. But soon, we came to know that there are many U2R attacks of buffer-overflow, having the same characteristics of intrusion only patterns.

On further investigating, we identified that there is an execution of a user program with a system tool SUID and a shell program. Hence there is a need to build a feature that handles the normal behaviour of the attack. But in the event data, we are having specific operating system information, we have to use domain knowledge (KDD process, Fayyad *et al*., 1996; Lee *et al*., 1999) to acquire the general and abstract information. Here all the limitations of fully automatic feature construction has been analyzed in case of a low level event data, but still the intrusion-only patterns from the pattern mining and comparison gives more helpful information for the manual feature construction. We defined some set of features as shown in the following table for the BSM session records.

Table 8 shows some set of features for BSM session records and some of the features (those in bold) are from the buffer overflow patterns, while others are similar to 'content' features as mentioned above.



(a)



(b)

Fig. 2: ROC curves showing Detection Rates and False Alarm Rates (a) U2R (b) DOS, R2L, PROBING and over all

BSM misuse detection models: In the BSM session records, each record is labeled as 'normal' or an intrusion name. All the BSM session records are aggregated in to a single dataset for this experiment. The following table (Table 9) shows some example records of BSM session.

We then applied the RIPPER rules for BSM session records to know the classification rules. We tested the performance of the rule set on these test data using the DARPA files. Here we compared our model with other models in terms of performance in detecting the U2R attack and the performance of our model in detecting DOS, R2L and PROBING attacks and finally the overall performance of the attacks. These ROC curves of the detection models were shown the following figure (Fig. 2).

Results of BSM data: The following figure shows the performance of BSM Misuse Detection Models. Since there are much fewer attacks contained in the BSM data of a single host, the model has slightly better performance. From the above figure it is very clear that the BSM model has good performance in detecting DOS, U2R and PROBING attack but very poor performance in R2L attacks. When compared with the predictions between the BSM and the *tcpdump* models, we found that they simply agree with other's predictions on the pairs of host session and network connections, because of the following reasons:

- The nature of intrusion detection, one that uses a service that was not modeled can go undetected and
- BSM model feature and *tcpdump* model features are very similar for the evidences with different data sources

Our experiments in meta-learning, where a combined model was computed based on a model for *tcpdump* header-only connection data and a model for BSM host session data, indeed showed that the same level of accuracy was maintained as using a heavyweight *tcpdump* model that also checked the IP data contents.

CONCLUSION

In this study we presented detailed performance evaluation experimentation results on network *tcpdump* data and on operating system BSM audit data set. The experiments on *tcpdump* data showed the effectiveness of MADAM ID's automatic pattern mining and comparison and feature construction procedures. These patterns were mechanically parsed to construct a set of temporal and statistical 'traffic' features for the detection models. There are no intrusion-only patterns from connection records of R2L and U2R attacks since they involve in a single connection. We have used domain knowledge to define a set of 'content' features for these attacks. Where as in the case of BSM data experiments, we found that the intrusion-only patterns of buffer overflow attacks contain specific program names that are not inherent to the attack method. This is because, compared with connection records which are more general and their semantics well understood, the BSM audit records are contains low level details of system events. Hence we need to use domain knowledge to interprete these patterns. And the most general information is aggregated into BSM session records.

This shows the advantages of using MADAM ID to process huge volume of audit data, construct features and inductively lean more classification rules. However, since our models were intended for misuse detection and were trained using only the available data sets, a number of new attacks in the test may not be identified and these attacks need to be detected as well.

REFERENCES

Abraham, T., 2001. IDDM: Intrusion Detection using Data Mining techniques. DTIC.

Atkins, D., 1996. Internet Security Professional Reference. 1st Edn., New Riders Publishing, Indianapolis, ISBN-10: 1562055577, pp: 908.

Bellovin, S.M., 1989. Security problems in the TCP/IP protocol suite. Comput. Commun. Rev., 19: 32-48. DOI: 10.1145/378444.378449

Dickerson, J.E. and J.A. Dickerson, 2000. Fuzzy network profiling for intrusion detection. Proceedings of the NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society, (NAFIPS' 00), IEEE Xplore Press, Atlanta, Atlantan, pp: 301-306. DOI: 10.1109/NAFIPS.2000.877441

Durst, R., T. Champion, B. Witten, E. Miller and L. Spagnuolo, 1999. Testing and evaluating computer intrusion detection systems. Commun. ACM, 42: 53-61. DOI: 10.1145/306549.306571

Fayyad, U.M., G. Piatetsky-Shapiro and P. Smyth, 1996. The KDD process for extracting useful knowledge from volumes of data. Commun. ACM., 39: 27-34. DOI: 10.1145/240455.240464

Ghosh, A.K., A. Schwartzband and M. Schatz, 1999. Learning program behavior profiles for intrusion detection. Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring, (WIDNM' 99), USENIX Technical Program, pp: 51-62.

Halme, L.R. and R.K. Bauer, 1995. Intrusion Detection FAQ: AINT Misbehaving: A Taxonomy of Anti-Intrusion Techniques. Arca Systems, Inc.

Ilgun, K.U., 1993. USTAT: a real-time intrusion detection system for UNIX. Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, May 24-26, IEEE Xplore press, Oakland, CA, pp: 16-28. DOI: 10.1109/RISP.1993.287646

Lee, W., S.J. Stolfo and K.W. Mok, 1999. A data mining framework for building intrusion detection models. Proceedings of the IEEE Symposium on Security and Privacy, (SSP’ 99), IEEE Xplore Press, Oakland, pp: 120-132. DOI: 10.1109/SECPRI.1999.766909

Nazer, G.M. and A.A.L. Selvakumar, 2011. Current intrusion detection techniques in information technology-a detailed analysis. Eur. J. Sci. Res., 65: 611-624.

Nazer, G.M. and A.L. Selvakumar, 2012a. A systematic framework for analyzing audit data and constructing network ID models. CIIT Int. J. Data Mining Knowledge Manag., 4: 178-185.

Nazer, G.M. and A.L. Selvakumar, 2012b. Intelligent data mining techniques for intrusion detection models on network. Eur. J. Sci. Res., 71: 36-45.

Paxson, V., 1997. End-to-end Internet packet dynamics. Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication, (SIGCOMM’ 97), ACM, USA, pp: 139-152. DOI: 10.1145/263105.263155

Paxson, V., 1998. A system for detecting network intruders in real-time. Proceedings of the 7th USENIX Security Symposium, Jan. 26-29, San Antonio, Texas, pp: 1-22.

Porras, P.A. and A. Valdes, 1998. Live traffic analysis of TCP/IP gateways. Proceedings of the Internet Society's Networks and Distributed Systems Security Symposium, (ISNDSSS’ 98), pp: 2-13.

Stevens, W.R., 1994. TCP/IP Illustrated. Addison-Wesley Publishing Company.

Sunsoft, S.D., 1995. Basic security Module Guide. SunSoft.