International Journal of Engineering Inventions
ISSN: 2278-7461, www.ijeijournal.com
Volume 1, Issue 4 (September2012) PP: 68-76

Integrated Conditional Random Fields with the Layered Approach for Intrusion Detection

K Ranganath[1], Shaik Shafia[2]
[1]*Computer Science and Engineering Department Hyderabad Institute of Technology And Management Management R.R.Dist*
[2]*Hyderabad Institute of Technology And Computer Science and Engineering Department R.R.Dist*

Abstract— Both Conditional Random Fields (CRFs) and Layered Approach have some things in common. They can be used for solving two issues of Accuracy and Efficiency. They solely do have certain disadvantages and advantages which almost completely disappear by combining both concepts. Intrusion detection (ID) is a type of security management system for computers and networks. An ID system gathers and analyzes information from various areas within a computer or a network to identify possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization).The CRFs can effectively model such relationships among different features of an observation resulting in higher attack detection accuracy. Another advantage of using CRFs is that every element in the sequence is labelled such that the probability of the entire labelling is maximized, i.e., all the features in the observation collectively determine the final labels. Hence, even if some data is missing, the observation sequence can still be labelled with less number of features. A layered model is to reduce computation and the overall time required to detect anomalous events. The time required to detect an intrusive event is significant and can be reduced by eliminating the communication overhead among different layers. To improve the speed of operation of the system. Hence, we implement the LIDS and select a small set of features for every layer rather than using all the 41 features. This results in significant performance improvement during both the training and the testing of the system. This project presents high attack detection accuracy can be achieved by using CRFs and high efficiency by implementing the Layered Approach. Finally, we show that our system is robust and is able to handle noisy data without compromising performance

Keywords— CERT, CRF, Intrusion, Session

I. INTRODUCTION

In this project we address three significant issues which severely restrict the utility of anomaly and hybrid Intrusion Detection systems in present networks and applications. The three issues are; limited attack detection coverage, large number of false alarms and inefficiency in operation. Present anomaly and hybrid intrusion detection systems have limited attack detection capability, suffer from a large number of false alarms and cannot be deployed in high speed networks and applications without dropping audit patterns. Hence, most existing intrusion detection systems such as the USTAT, IDIOT, EMERALD, Snort and others are developed using knowledge engineering approaches where domain experts can build focused and optimized pattern matching models [1]. Though such systems result in very few false alarms [2], they are specific in attack detection and often tend to be incomplete. As a result their effectiveness is limited. We, thus, address these shortcomings and develop better anomaly and hybrid intrusion detection systems which are accurate in attack detection, efficient in operation and have wide attack detection coverage.

The objective of an intrusion detection system is to provide data security and ensure continuity of services provided by a network [3]. Present networks provide critical services which are necessary for businesses to perform optimally and are, thus, a target of attacks which aim to bring down the services provided by the network. Additionally, with more and more data becoming available in digital format and more applications being developed to access this data, the data and applications are also a victim of attackers who exploit these applications to gain access to data. With the deployment of more sophisticated security tools, in order to protect the data and services, the attackers often come up with newer and more advanced methods to defeat the installed security systems [4], [5].According to the Internet Systems Consortium (ISC) survey, the number of hosts on the Internet exceeded 550,000,000 in July 2008 [6]. Earlier, a project in 2002, estimated the size of the Internet to be 532,897 TB [7]. Increasing dependence of businesses on the services over the Configuration errors and vulnerabilities in software are exploited by the attackers who launch powerful attacks such as the Denial of Service (DoS) [8] and Information attacks [9].

According to the Computer Emergency Response Team (CERT), the number of vulnerabilities in software has been increasing and many of them exist in highly deployed software [10].Considering that it is near to impossible to build 'perfect' software, it becomes critical to build effective intrusion detection systems which can detect attacks reliably. The prospect of obtaining valuable information, as a result of a successful attack, subside the threat of legal convictions. The problem becomes more profound since authorized users can misuse their privileges and attackers can masquerade as authentic users by exploiting vulnerable applications. Given the diverse type of attacks (DoS, Probing, Remote to Local, User to Root and others), it is a challenge for any intrusion detection system to detect a wide variety of attacks with very few false alarms in real time environment. Ideally, the system must detect all intrusions with no false alarms. The challenge is, thus, to build a system which has broad attack detection coverage and at the same time which results in very few false alarms. The system must also be efficient enough to handle large amount of audit data without affecting performance at the deployed environment. However, this is in no way a solution for securing today's highly networked computing environment and, hence, the need to develop better intrusion detection systems

II. BACKGROUND

Detection Intrusion in networks and applications has become one of the most critical tasks to prevent their misuse by attackers. Intrusion detection started in 1980's and since then a number of approaches have been introduced to build intrusion detection systems [1]. However, intrusion detection is still at its infancy and naive attackers can launch powerful attacks which can bring down an entire network [5]. To identify the shortcoming of different approaches for intrusion detection, we explore the related research in intrusion detection. We describe the problem of intrusion detection in detail and analyse various well known methods for intrusion detection with respect to two critical requirements viz. accuracy of attack detection and efficiency of system operation. We observe that present methods for intrusion detection suffer from a number of drawbacks which significantly affect their attack detection capability. Hence, we introduce conditional random fields for effective intrusion detection and motivate our approach for building intrusion detection systems which can operate efficiently and which can detect a wide variety of attacks with relatively higher accuracy, both at the network and at the application level.

A. Intrusion Detection and Intrusion Detection System

The intrusion detection systems are a critical component in the network security arsenal. Security is often implemented as a multi layer infrastructure and different approaches for providing security can be categorized into the following six areas

Attack Deterrence – Attack deterrence refers to persuading an attacker not to launch an attack by increasing the perceived risk of negative consequences for the attacker. Having a strong legal system may be helpful in attack deterrence. However, it requires strong evidence against the attacker in case an attack was launched.

Attack Prevention – Attack prevention aims to prevent an attack by blocking it before an attack can reach the target. However, it is very difficult to prevent all attacks. This is because, to prevent an attack, the system requires complete knowledge of all possible attacks as well as the complete knowledge of all the allowed normal activities which is not always available. An example of attack prevention system is a firewall.

Attack Deflection – Attack deflection refers to tricking an attacker by making the attacker believe that the attack was successful though, in reality, the attacker was trapped by the system and deliberately made to reveal the attack.

Attack Avoidance – Attack avoidance aims to make the resource unusable by an attacker even though the attacker is able to illegitimately access that resource. An example of security mechanism for attack avoidance is the use of cryptography.

Attack Detection – Attack detection refers to detecting an attack while the attack is still in progress or to detect an attack which has already occurred in the past. Detecting an attack is significant for two reasons; first the system must recover from the damage caused by the attack and second, it allows the system to take measures to prevent similar attacks in future.

Attack Reaction and Recovery – Once an attack is detected, the system must react to an attack and perform the recovery mechanisms as defined in the security policy. Tools available to perform attack detection followed by reaction and recovery are known as the intrusion detection systems. However, the difference between intrusion prevention and intrusion detection is slowly diminishing as the present intrusion detection systems increasingly focus on real time attack detection and blocking an attack before it reaches the target. Such systems are better known as the Intrusion Prevention Systems.

B. Principles and Assumptions in Intrusion Detection

The principle states that for a system which is not under attack, the following three conditions hold true:

- Actions of users conform to statistically predictable patterns.
- Actions of users do not include sequences which violate the security policy.
- Actions of every process correspond to a set of specifications which describe what the process is allowed to do.

Systems under attack do not meet at least one of the three conditions. Further, intrusion detection is based upon some assumptions which are true regardless of the approach adopted by the intrusion detection system. These assumptions are:

- There exists a security policy which defines the normal and (or) the abnormal usage of every resource.
- The patterns generated during the abnormal system usage are different from the patterns generated during the normal usage of the system; i.e., the abnormal and normal usage of a system results in different system behavior. This difference in behavior can be used to detect intrusions.

C. Components of Intrusion Detection Systems

An intrusion detection system typically consists of three sub systems or components:

Data Preprocessor – Data preprocessor is responsible for collecting and providing the audit data (in a specified form) that will be used by the next component (analyser) to make a decision. Data preprocessor is, thus, concerned with collecting the data from the desired source and converting it into a format that is comprehensible by the analyser Data used for detecting intrusions range from user access patterns to network packet level features (such as the source and destination IP addresses, type of packets and rate of occurrence of packets) to application and system level behaviour (such as the sequence of system calls generated by a process.) We refer to this data as the audit patterns.

Analyzer (Intrusion Detector) – The analyser or the intrusion detector is the core component which analyses the audit patterns to detect attacks. This is a critical component and one of the most researched. Various pattern matching, machine learning, data mining and statistical techniques can be used as intrusion detectors. The capability of the analyser to detect an attack often determines the strength of the overall system.

Response Engine – The response engine controls the reaction mechanism and determines how to respond when the analyzer detects an attack. The system may decide either to raise an alert without taking any action against the source or may decide to block the source for a predefined period of time. Such an action depends upon the predefined security policy of the network. The authors define the Common Intrusion Detection Framework (CIDF) which recognizes a common architecture for intrusion detection systems. The CIDF defines four components that are common to any intrusion detection system. The four components are; Event generators (E-boxes), event Analyzers (A-boxes), event Databases (D-boxes) and the Response units (R-boxes). The additional component, called the D-boxes, is optional and can be used for later analysis.

D. Challenges and Requirements for Intrusion Detection Systems

The purpose of an intrusion detection system is to detect attacks. However, it is equally important to detect attacks at an early stage in order to minimize their impact. The major challenges and requirements for building intrusion detection systems are:

- The system must be able to detect attacks reliably without giving false alarms. It is very important that the false alarm rate is low as in a live network with large amount of traffic, the number of false alarms may exceed the total number of attacks detected correctly thereby decreasing the confidence in the attack detection capability of the system. Ideally, the system must detect all intrusions with no false alarms, i.e. it can detect a wide variety of attacks and at the same time which results in very few false alarms.
- The system must be able to handle large amount of data without affecting performance and without dropping data, i.e. the rate at which the audit patterns are processed and decision is made must be greater than or equal to the rate of arrival of new audit patterns. In addition, the system must be capable of operating in real time by initiating a response mechanism once an attack is detected.
- A system which can link an alert generated by the intrusion detector to the actual security incident is desirable. Such a system would help in quick analysis of the attack and may also provide effective response to intrusion as opposed to a system which offers no after attack analysis. Hence, it is not only necessary to detect an attack, but it is also important to identify the type of attack.
- It is desirable to develop a system which is resistant to attacks since, a system that can be exploited during an attack may not be able to detect attacks reliably.

III. IMPLEMENTATION

Ever increasing network bandwidth poses a significant challenge to build efficient network intrusion detection systems which can detect a wide variety of attacks with acceptable reliability. In order to operate in high traffic environment, present network intrusion detection systems are often signature based. As a result, anomaly and hybrid intrusion detection systems must be used to detect novel attacks. However, such systems are inefficient and suffer from a large false alarm rate. To ameliorate these drawbacks, we first develop better hybrid intrusion detection methods which are not based on attack signatures and which can detect a wide variety of attacks with very few false alarms. Given the network audit patterns where every connection between two hosts is presented in a summarized form with 41 features, our objective is to detect most of the anomalous connections while generating very few false alarms. In our experiments, we used the KDD 1999 data set. Conventional methods, such as decision trees and naive Bays, are known to perform well in such an environment; however, they assume observation features to be independent. We propose to use conditional random fields which can capture the correlations among different features in the data

The KDD 1999 data set represents multiple features, a total of 41, for every session in relational form with only one label for the entire record. However, we represent the audit data in the form of a sequence and assign label to every feature in the sequence using the first order Markov assumption instead of assigning a single label to the entire observation. Though, this increases complexity, it also improves the attack detection accuracy. Figure 3.1 represents how conditional random fields can be used for detecting network intrusions.

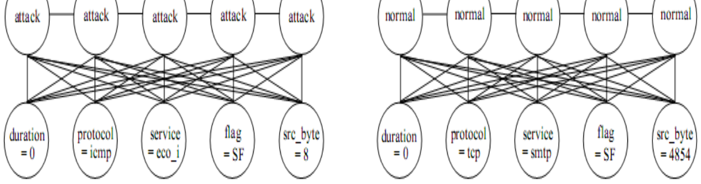


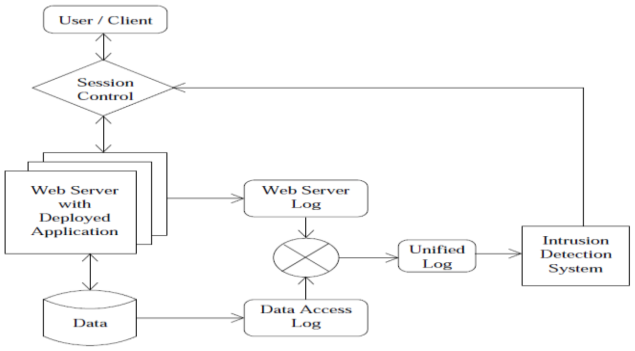
Figure3.1: Conditional Random Fields for Intrusion Detection

In the figure, observation features ‘duration’, ‘protocol’, ‘service’, ‘flag’ and ‘source bytes’ are used to discriminate between attack and normal events. The features take some possible value for every connection which are then used to determine the most likely sequence of labels < attack, attack, attack, attack, attack > or < normal, normal, normal, normal, normal >. Custom feature functions can be defined which describe the relationships among different features in the observation. During training, feature weights are learnt and during testing, features are evaluated for the given observation which is then labeled accordingly. It is evident from the figure that every input feature is connected to every label which indicates that all the features in an observation determine the final labeling of the entire sequence. Thus, a conditional

random field can model dependencies among different features in an observation. Present intrusion detection systems do not consider such relationships. They either consider only one feature, as in case of system call modeling, or assume independence among different features in an observation, as in case of a naive Bays classifier.
We also note that in the KDD 1999 data set, attacks can be represented in four classes; Probe, DoS, R2L and U2R. In order to consider this as a two class classification problem, the attacks belonging to all the four attack classes can be relabeled as attack and mixed with the audit patterns belonging to the normal class to build a single model which can be trained to detect any kind of attack. The problem can also be considered as a five class classification problem, where a single system is trained with five classes (normal, Probe, DoS, R2L and U2R) instead of two. As we will see from our experimental result, considering every attack class separately not only improves the attack detection accuracy but also helps to improve the overall system performance when integrated with the layered framework. Furthermore, it also helps to identify the class of an attack once it is detected at a particular layer in the layered framework. However, a drawback of this implementation is that it requires domain knowledge to perform feature selection for every layer.

E. Description of our Framework

We present our unified logging framework in Figure 3.2 which can be used for building effective application intrusion detection system



.

Figure 3.2: Framework for Building Application Intrusion Detection System

In our framework, we define two modules; session control module and logs unification module, in addition to an intrusion detection system which is used to detect malicious data accesses in an application. The logs unification module provides input audit patterns to the intrusion detection system and the response generated by the intrusion detection system is passed on to the session control module which can initiate appropriate intrusion response mechanisms. In our framework, every request first passes through the session control which is described next.

Session Control Module

The prime objective of an intrusion detection system is to detect attacks reliably. However, it must also ensure that once an attack is detected, appropriate intrusion response mechanisms are activated in order to mitigate their impact and prevent similar attacks in future. The session control module serves dual purpose in our framework. First, it is responsible for establishing new sessions and for checking the session id for previously established sessions. For this, it maintains a list of valid sessions which are allowed to access the application. Every request to access the application is checked for a valid session id at the session control and anomalous requests can be blocked depending upon the installed security policy. Second, the session control also accepts input from the intrusion detection system. As a result, it is capable of acting as an intrusion response system. If a request is evaluated to be anomalous by the intrusion detection system, the response from the application can be blocked at the session control before data is made visible to the user, thereby preventing malicious data accesses in real time.

The session control can either be implemented as a part of the application or can also be implemented as a separate entity. Once the session id is evaluated for a request, the request is sent to the application where it is processed. The web server logs every request. All corresponding data accesses are also logged. The two logs are then combined by the logs unification module to generate unified log which is described next.

Logs Unification Module

Analyzing the web assesses logs and the data access logs in isolation are not sufficient to detect application level attacks. Hence, we propose using unified log which can better detect attacks as compared to independent analysis of the two logs. The logs unification module is used to generate the unified log. The unified log incorporates features from both the web access logs and the corresponding data access logs. Using the unified log, thus, helps to capture the user application interaction and the application data interactions. Hence, we first process the data access logs and represent them using simple statistics such as 'the number of queries invoked by a single web request' and 'the time taken to process them' rather than analyzing every data access individually. We then use the session id, present in both, the application access logs and the associated data access logs, to uniquely map the extracted statistics (obtained from the data access logs) to the corresponding web requests in order to generate a unified log. Figure 3.3, represents how the web access logs and the corresponding data access logs can be uniquely mapped to generate a unified log. In the figure, f1, f2, f3...fn and $g_{1'}, g_{2'}, \dots g_{m'}$ represent the features of web access logs and the features extracted from the reduced data access logs respectively.

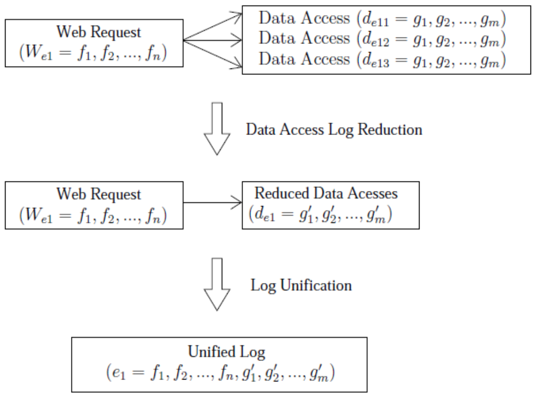

Web Request
$(W_{e1} = f_1, f_2, ..., f_n)$
Data Access $(d_{e11} = g_1, g_2, ..., g_m)$
Data Access $(d_{e12} = g_1, g_2, ..., g_m)$
Data Access $(d_{e13} = g_1, g_2, ..., g_m)$
Data Access Log Reduction
Web Request
$(W_{e1} = f_1, f_2, ..., f_n)$
Reduced Data Acesses
$(d_{e1} = g'_1, g'_2, ..., g'_m)$
Log Unification
Unified Log
$(e_1 = f_1, f_2, ..., f_n, g'_1, g'_2, ..., g'_m)$

Figure 3.3: Representation of a Single Event in the Unified log

From Figure 3.3, we observe that a single web request may result in more than one data accesses which depend upon the logic encoded into the application. Once the web access logs and the corresponding data access logs are available, the next step involves the reduction of data access logs by extracting simple statistics as discussed before. The session id can, then, be used to uniquely combine the two logs to generate the unified log.

Issues in Implementation

Experimental results show that our approach based on conditional random fields can be used to build effective application intrusion detection systems. However, before deployment, it is critical to resolve issues such as the availability of the training data and suitability of our approach for a variety of applications. We now discuss various methods which can be employed to resolve such issues.

Availability of Training Data

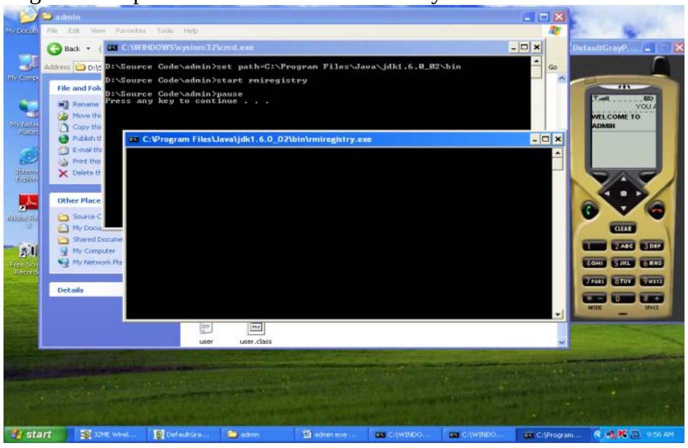
Though our system is application independent and can be used to detect malicious data access in a variety of applications, it must be trained before the system can be deployed online to detect attacks. This requires training data which is specific to the application. To obtain such data may be difficult However; training data can be made available as early as during the application testing phase when the application is tested to identify errors. Logs generated during the application testing phase can be used for training the intrusion detection system. However, this requires security aware software engineering practices which must ensure that necessary measures are taken to provide training data during the application development phase, which can be used to train effective application intrusion detection systems.

Suitability of Our Approach for a Variety of Applications

As we already discussed, our framework is generic and can be deployed for a variety of applications. It is particularly suited to applications which follow the three tier architecture which have application and data independence. Furthermore, our framework can be easily extended and deployed in the Service Oriented Architecture. Our proposed framework can be considered as a special case for the service oriented architecture which defines only one service. The challenge is to identify such correlations automatically and this provides an interesting direction for future work.

IV. RESULTS

The Programs Developed has been validated by testing them with variety of inputs. Here we can see the outputs for different inputs which are given as input to the Intrusion Detection system.

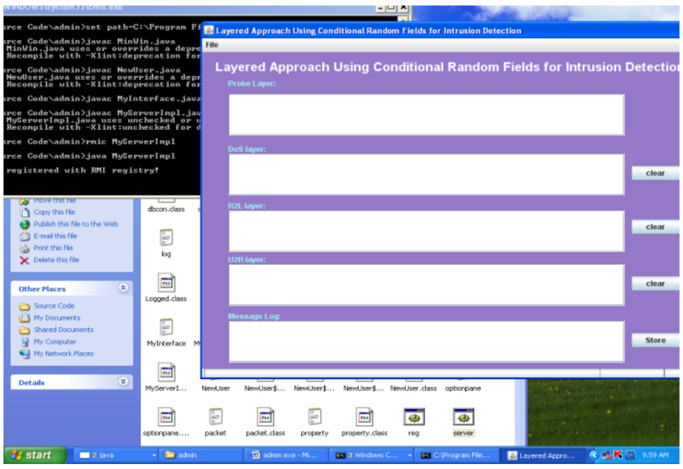

C:\WINDOWS\system32\cmd.exe

```
D:\Source Code\admin>set path=C:\Program Files\Java\jdk1.6.0_02\bin

D:\Source Code\admin>start rmiregistry

D:\Source Code\admin>pause
Press any key to continue . . .
```

C:\Program Files\Java\jdk1.6.0_02\bin\rmiregistry.exe

Figure 4.1: RMI Registry window

This form shows RMI Registry, by using which we can invoke one machine can be invoked by other method of the object running inside other machine

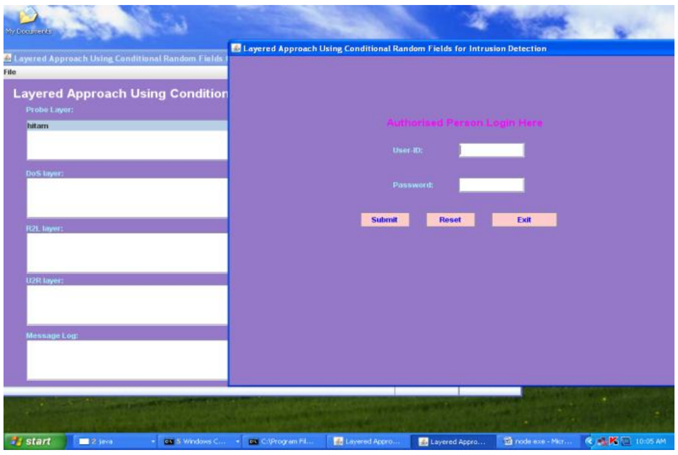


```
rce Code\admin>set path=C:\Program Fi

rce Code\admin>javac MinWin.java
MinWin.java uses or overrides a depre
Recompile with -Xlint:deprecation for

rce Code\admin>javac NewUser.java
NewUser.java uses or overrides a depr
Recompile with -Xlint:deprecation for

rce Code\admin>javac MyInterface.java

rce Code\admin>javac MyServerImpl.jav
MyServerImpl.java uses unchecked or u
Recompile with -Xlint:unchecked for d

rce Code\admin>rmic MyServerImpl

rce Code\admin>java MyServerImpl

registered with RMI registry!
```

Figure 4.2 Intrusion Detection System Windows.

This form shows the main screen of the Intrusion Detection System

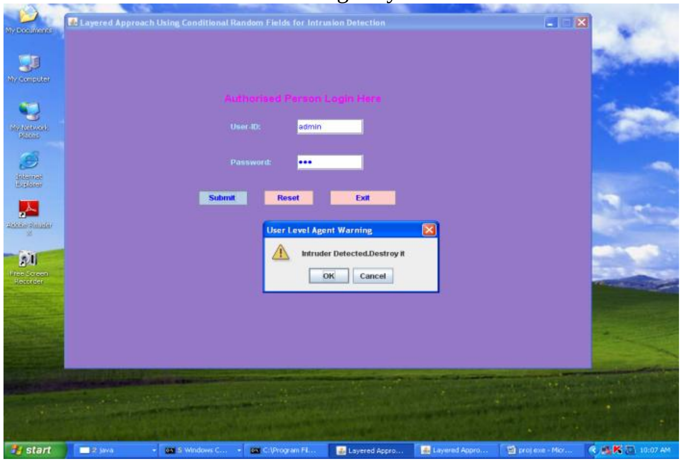
Figure 4.3: Authorized Person Login System Window This form shows the main screen of the Authorized Person Login System



Layered Approach Using Conditional Random Fields for Intrusion Detection
Probe Layer:
hitam
DoS layer:
R2L layer:
U2R layer:
Message Log:
Authorised Person Login Here
User-ID:
Password:
Submit
Reset
Exit

Figure 4.4: Home Page of actual Intrusion Detection System Window

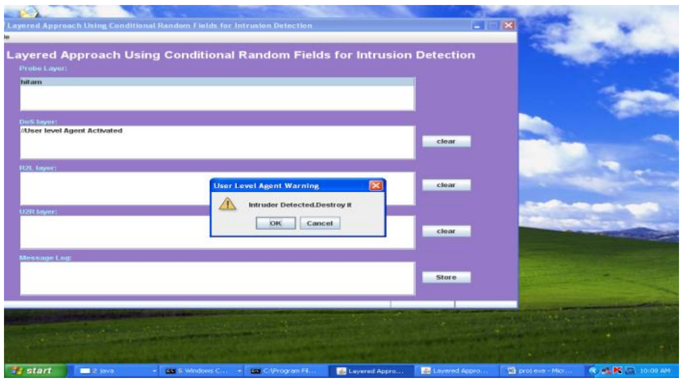
This form shows the main screen of the Authorized Person Login System and Intrusion Detection System



Layered Approach Using Conditional Random Fields for Intrusion Detection
Authorised Person Login Here
User-ID:
admin
Password:
Submit
Reset
Exit
User Level Agent Warning

Intruder Detected.Destroy it

Figure 4.5(a): password mismatch window


Layered Approach Using Conditional Random Fields for Intrusion Detection

Probe Layer:
hitam

DoS layer:
//User level Agent Activated

R2L layer:

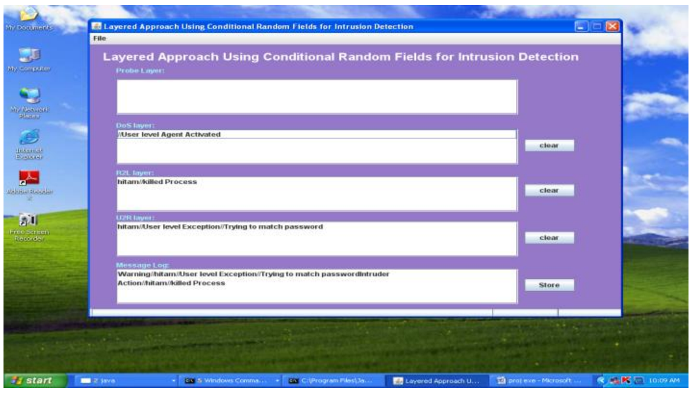
U2R layer:

Message Log:

User Level Agent Warning
Intruder Detected.Destroy it

Figure 4.5(b): IDS detecting unauthorized user window

This Forms shows Intrusion is detected when the user enter the flaw credentials


Layered Approach Using Conditional Random Fields for Intrusion Detection

Probe Layer:

DoS layer:
//User level Agent Activated

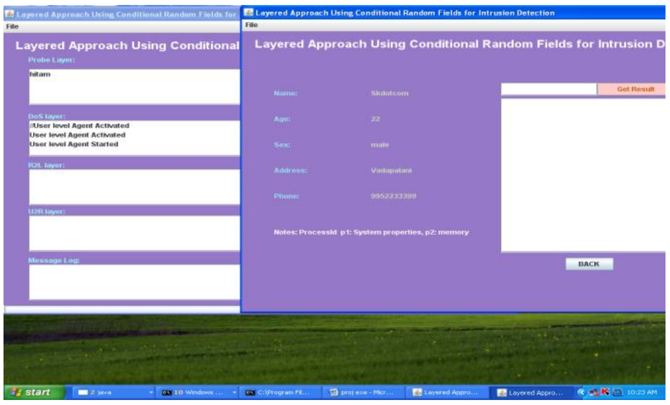
R2L layer:
hitam//killed Process

U2R layer:
hitam//User level Exception//Trying to match password

Message Log:
Warning//hitam//User level Exception//Trying to match passwordIntruder
Action//hitam//killed Process

Figure 4.6: Unauthorized user detection window

This Form shows what type of operation unauthorized user performed and what type of actions taken by the Intrusion Detection System performed


Layered Approach Using Conditional Random Fields for Intrusion Detection

Probe Layer:
hitam

DoS layer:
//User level Agent Activated
User level Agent Activated
User level Agent Started

R2L layer:

U2R layer:

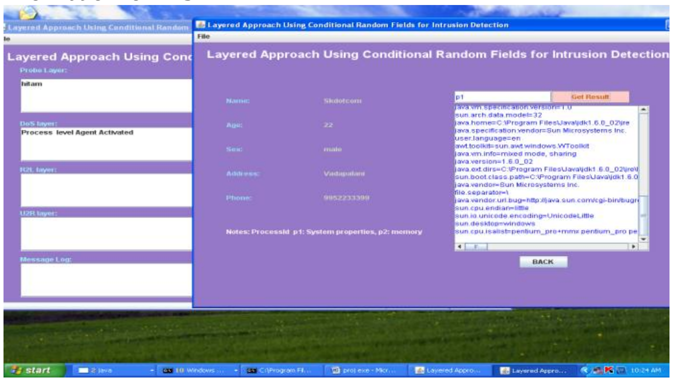
Message Log:

Name: Skdotcom
Age: 22
Sex: male
Address: Vadapalani
Phone: 9952233399

Notes: ProcessId p1: System properties, p2: memory

Figure 4.7: Authorized user IDS window

This form shows valid user Information for IDS


Layered Approach Using Conditional Random Fields for Intrusion Detection

Probe Layer:
hitam

DoS layer:
Process level Agent Activated

R2L layer:

U2R layer:

Message Log:

Name: Skdotcom
Age: 22
Sex: male
Address: Vadapalani
Phone: 9952233399

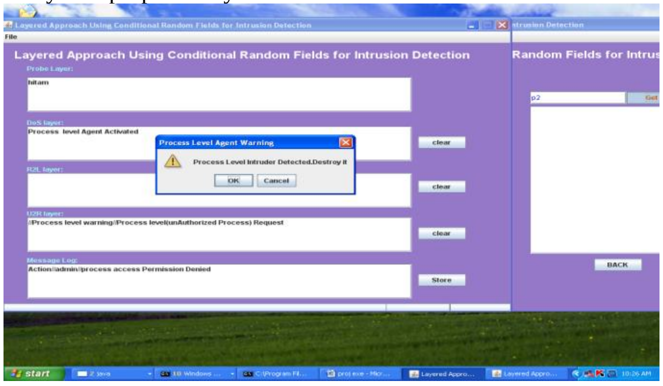
Notes: ProcessId p1: System properties, p2: memory

p1

```
java.vm.specification.version=1.0
sun.arch.data.model=32
java.home=C:\Program Files\Java\jdk1.6.0_02\jre
java.specification.vendor=Sun Microsystems Inc.
user.language=en
awt.toolkit=sun.awt.windows.WToolkit
java.vm.info=mixed mode, sharing
java.version=1.6.0_02
java.ext.dirs=C:\Program Files\Java\jdk1.6.0_02\jre\
sun.boot.class.path=C:\Program Files\Java\jdk1.6.0
java.vendor=Sun Microsystems Inc.
file.separator=\
java.vendor.url.bug=http://java.sun.com/cgi-bin/bugr
sun.cpu.endian=little
sun.io.unicode.encoding=UnicodeLittle
sun.desktop=windows
sun.cpu.isalist=pentium_pro+mmx pentium_pro pe
```

Figure 4.8: Authorized user retrieving system properties window

This Form shows retrieving the system properties by the authorized user


Layered Approach Using Conditional Random Fields for Intrusion Detection
File
Layered Approach Using Conditional Random Fields for Intrusion Detection
Probe Layer:
DoS layer:
R2L layer:
U2R layer:
Message Log:
clear
clear
clear
Store
Random Fields for Intrus
Get
BACK
OK
Cancel
start

Probe Layer: hitam

DoS layer: Process level Agent Activated

Process Level Agent Warning: Process Level Intruder Detected.Destroy it

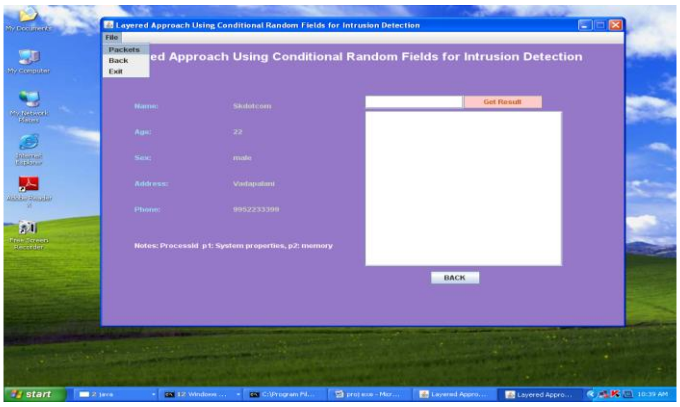
U2R layer: //Process level warning//Process level(unAuthorized Process) Request

Message Log: Action//admin//process access Permission Denied

p2

Figure 4.9: Process level intruder detection window

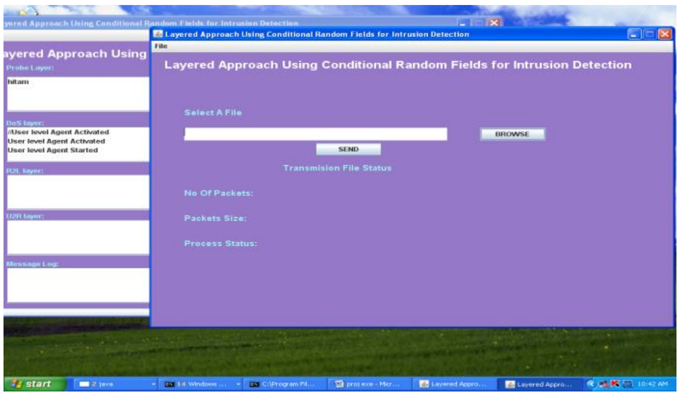
This form shows if authorized user trying to retrieve the processes level information then IDS detected and gives the alert information as dialog box


Layered Approach Using Conditional Random Fields for Intrusion Detection
File
Packets
Back
Exit
ed Approach Using Conditional Random Fields for Intrusion Detection
Get Result
BACK
start

Name:	Skdotcom
Age:	22
Sex:	male
Address:	Vadapalani
Phone:	9952233399

Notes: Processid p1: System properties, p2: memory

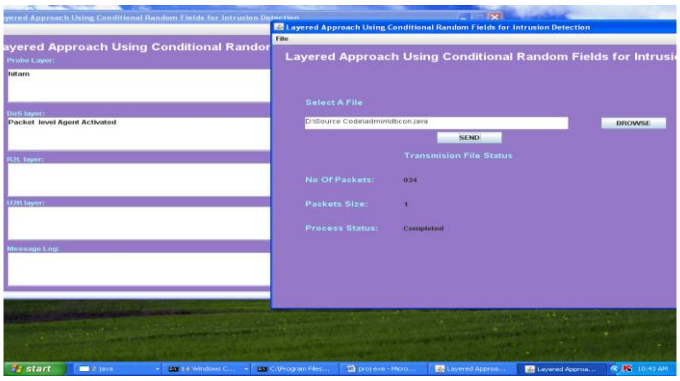
Figure 4.10(a): how to send data window



Probe Layer: hitam

DoS layer: //User level Agent Activated
User level Agent Activated
User level Agent Started

Figure 4.10(b): how to add file window


Layered Approach Using Conditional Random Fields for Intrusion Detection
File
Layered Approach Using Conditional Random Fields for Intrusi
Select A File
BROWSE
SEND
Transmision File Status
Probe Layer:
DoS layer:
R2L layer:
U2R layer:
Message Log:
start

Probe Layer: hitam

DoS layer: Packet level Agent Activated

Select A File: D:\Source Code\admin\dbcon.java

No Of Packets:	934
Packets Size:	1
Process Status:	Completed

Figure 4.11: Transmission status window

This form shows no. of packets are transferred, size of the packets and process status

V. CONCLUSION

We explored the suitability of conditional random fields for building robust and efficient intrusion detection systems which can operate, both, at the network and at the application level. In particular, we introduced novel frameworks and developed models which address three critical issues that severely affect the large scale deployment of present anomaly

VI. FUTURE ENHANCEMENT

The critical nature of the task of detecting intrusions in networks and applications leaves no mar- gin for errors. The effective cost of a successful intrusion overshadows the cost of developing intrusion detection systems and hence, it becomes critical to identify the best possible approach for developing better intrusion detection systems. Every network and application is custom designed and it becomes extremely difficult to develop a single solution which can work for every network and application. In this thesis, we proposed novel frameworks and developed methods which perform better than previously known approaches. However, in order to improve the overall performance of our system we used the domain knowledge for selecting better features for training our models. This is justified because of the critical nature of the task of intrusion detection. Using domain knowledge to develop better systems is not a significant disadvantage; however, developing completely automatic systems presents an interesting direction for future research. From our experiments, it is evident that our systems performed efficiently. However, developing faster implementations of conditional random fields particularly for the domain of intrusion detection requires further investigation.

REFERENCES

1. Stefan Axelsson. Research in Intrusion-Detection Systems: A Survey. Technical Report 98-17, Department of Computer Engineering, Chalmers University of Technology, 1998.

2. SANS Institute - Intrusion Detection FAQ. Last accessed: Novmeber 30, 2008. http: //www.sans.org/resources/idfaq/.

3. Kotagiri Ramamohanarao, Kapil Kumar Gupta, Tao Peng, and Christopher Leckie. The Curse of Ease of Access to the Internet. In Proceedings of the 3rd International Confer- ence on Information Systems Security (ICISS), pages 234–249. Lecture Notes in Computer Science, Springer Verlag, Vol (4812), 2007.

4. Overview of Attack Trends, 2002. Last accessed: November 30, 2008. http://www. cert.org/archive/pdf/attack_trends.pdf.

5. Kapil Kumar Gupta, Baikunth Nath, Kotagiri Ramamohanarao, and Ashraf Kazi. Attacking Confidentiality: An Agent Based Approach. In Proceedings of IEEE International Conference on Intelligence and Security Informatics, pages 285–296. Lecture Notes in Computer Science, Springer Verlag, Vol (3975), 2006.

6. The ISC Domain Survey. Last accessed: Novmeber 30, 2008. https://www.isc. org/solutions/survey/.

7. Peter Lyman, Hal R. Varian, Peter Charles, Nathan Good, Laheem Lamar Jordan, Joyojeet Pal, and Kirsten Swearingen. How much Information. Last accessed: Novmeber 30, 2008. http://www2.sims.berkeley.edu/research/ projects/how-much-info-2003.

8. Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems. ACM Computing Surveys,39(1):3, 2007. ACM.

9. Animesh Patcha and Jung-Min Park. An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. Computer Networks, 51(12): 3448–3470, 2007.

10. CERT/CC Statistics. Last accessed: Novmeber 30, 2008. http://www.cert.org/

AUTHORS

Mr.K.Ranganath, Graduated in Computer Science and Engineering from Osmania University Hyderabad, India, in 2006 and M.Tech in Software Engineering from Jawaharlal Nehru Technological University, Hyderabad, A.P., India in 2010. He is working presently as Assistant Professor in Department of C.S.E in Hyderabad Institute of Technology and Management (HITAM), R.R.Dist, INDIA, A.P. He has 4 years of Experience. His research interests include Network security and Data Mining.

Ms.Shaik Shafia, Graduated in Computer Science and Engineering from JNTU Hyderabad, India, in 2006 and M.Tech in Computer Science from Jawaharlal Nehru Technological University, Hyderabad, A.P., India in 2011. she is working presently as Assistant Professor in Department of C.S.E in Hyderabad Institute of Technology And Management (HITAM), R.R.Dist, INDIA, A.P. She has 6 years of Experience. Her research interests include Secure Computing