

Network Security Project

Prevention and Obstruction of Attacks On Systems through Honeypots



Rheeya Uppaal

Fourth Year, Computer Science

TABLE OF CONTENTS

1. Introduction and Problem Statement
2. UML Diagrams
3. Modules
 - a. Client
 - b. Server (Honey Pot)
 - c. Backend Server
4. Resources Utilised
5. Intelligent Intrusion Detection using Machine Learning
6. Conclusion
7. Individual Contribution

REFERENCES

APPENDIX

1. Introduction and Problem Statement

1.1 Extended Aim

The **POOH** (Prevention and Obstruction of attacks On systems through Honeypot) project aims to use honeypots as a security measure for detecting intrusions to a system.

A *honeypot* is an "an information system resource whose value lies in unauthorized or illicit use of that resources"(from the www.securityfocus.com forum). The computer system that is set up to act as a decoy to lure cyber attackers, and to detect, deflect or study attempts to gain unauthorized access to information systems. These honeypots are placed inside a production network with other production servers in the role of a decoy as part of a network intrusion detection system (IDS). They are designed to appear real and contain information or a resource of value with which to attract and occupy hackers. This decoy ties up the attacker's time and resources, hopefully giving administrators enough time to assess and mitigate any vulnerability in their actual production systems. The information gathered from the honeypot can also be useful in catching and prosecuting those behind an attack.

1.2 Introduction and Current Scenario

Many existing systems require manual definitions of normal and abnormal behaviour (intrusion signatures). It is unfeasible to identify these intrusion signatures automatically using machine learning or data mining techniques. These techniques analyze network or system activity logs to generate models or rules, which can then used by the system to detect intrusions. This prevents unauthorised access and protects the system integrity or reliability. However, most of the existing work on intrusion detection focuses on activities generated by a single source, resulting in many false positives and undetected intrusions. In the existing scenario, an intruder can easily enter into system and access the system. To prevent the same, we are proposing the solution of using Honeypots as a security measure.

1.3 Characteristics

Honeypots are a powerful, new technology with incredible potential. They can do everything from detecting new attacks never seen in the wild before, to tracking automated credit card fraud and identity theft. In the past several years we have seen the technology rapidly develop, with new concepts such as honeypot farms, commercial and open source solutions, and documented findings released.

- Luring possible attackers
- Incognito monitoring of possible attacker in stealth
- If attack detected, identifying the type of attack
- Reporting attack details, including host details, open ports on potential attacker, operating system fingerprint among more

1.4 Scope of the Project

The scope of this project is vast in terms of network security, and security application for Distributed Systems. The only user of the said system, is the *Network Administrator*, the

person overseeing the entire network, and continuously patching any loopholes in the security of the network. Hence, the scope of this project shall be specified from the view point of the Network Administrator.

Intrusion Detection Systems (IDS) generate huge amount of alerts, which corresponds to large volume of data. This data is time-consuming, resource intensive and costly to review. In contrast, Honeypots collect data only when someone (attacker) interacts with them, making this model easier & cost-effective to identify and act on authorized activity.

Honeypots sidestep the problem of large false positive detection, as any activity with them is by definition, unauthorized. This helps the Network Administrator to reduce, if not eliminate, false alerts. IDS, on the other hand, have the drawback of generating too many false positives.

Any activity with Honeypots, is anomalous, hence helping the Network Administrator to identify any new or previously unknown attacks instantly, and solving the problem of false negatives.

Honeypots, require minimal resources, even on large networks. According to Lance Spitzner, founder of the HoneyNet Project, a single Pentium computer with 128MB of RAM can be used to monitor millions of IP addresses.

1.5 Assumptions

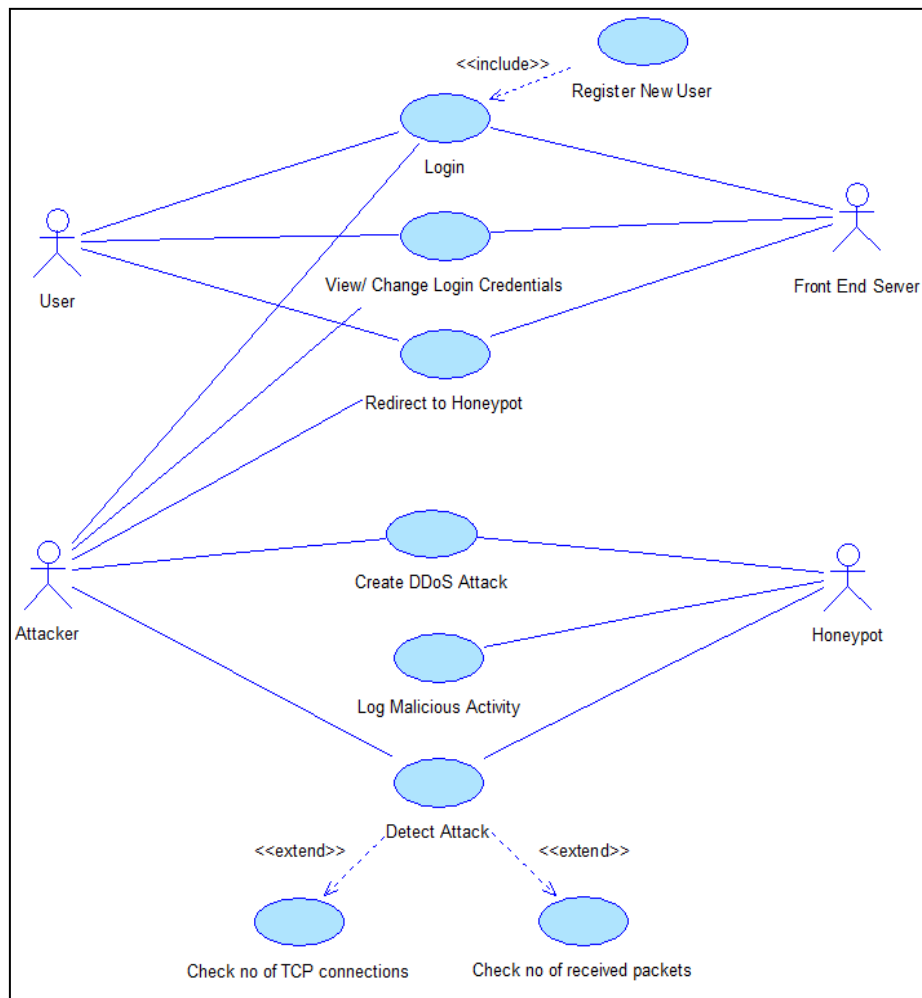
- Number of systems: 4
- Platform requirement:
- Software:
 - Front end - JDK 1.7 or above
 - Back end - MySQL
- Hardware:
 - Memory - 512 MB RAM minimum
 - Processor - Intel i3 processor or above

1.6 Limitations

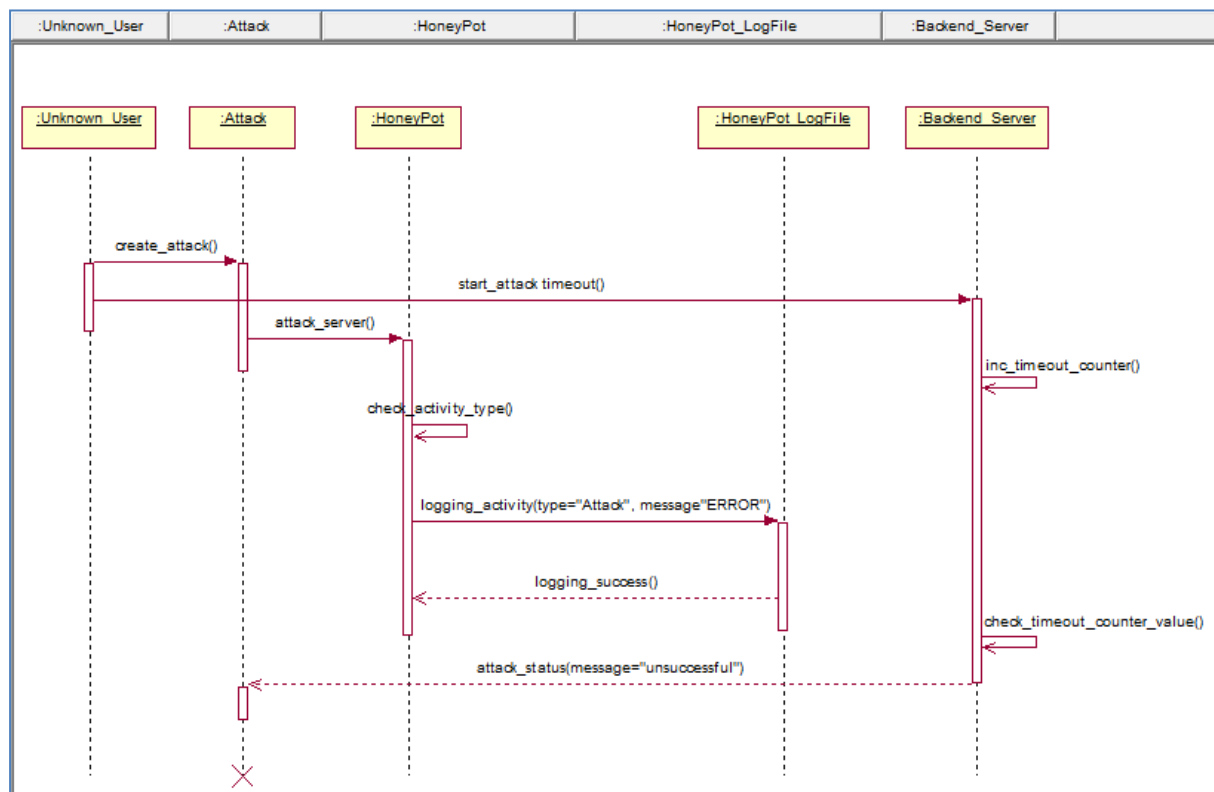
- The project shall utilize pseudo-live traffic generated artificially. Can be extended to live traffic in future iterations.
- The test environment consists of an intranet of four systems, wherein the following roles shall be played. Multiple instances of a particular type shall not be implemented.
 - Honeypot server
 - Backend server
 - Trusted user
 - Unknown user

2. UML Diagrams

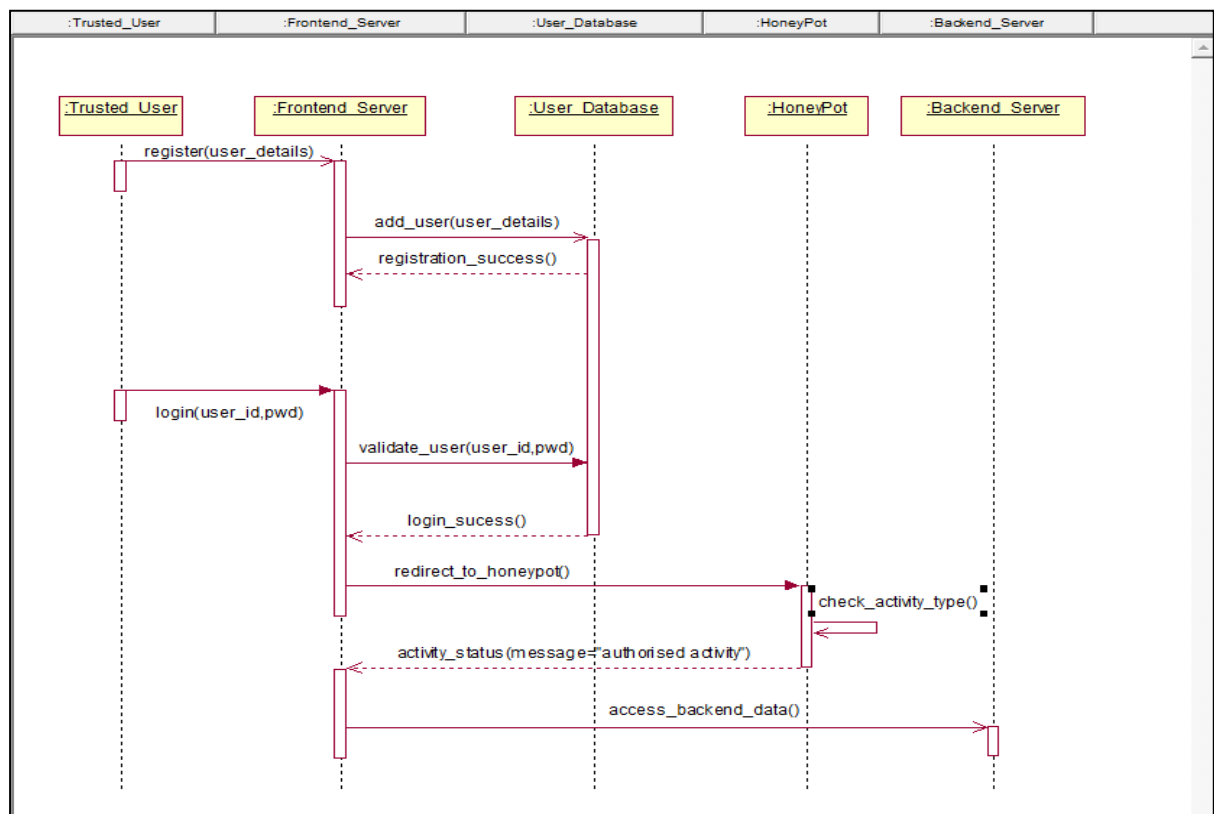
2.1 Use Case Diagram



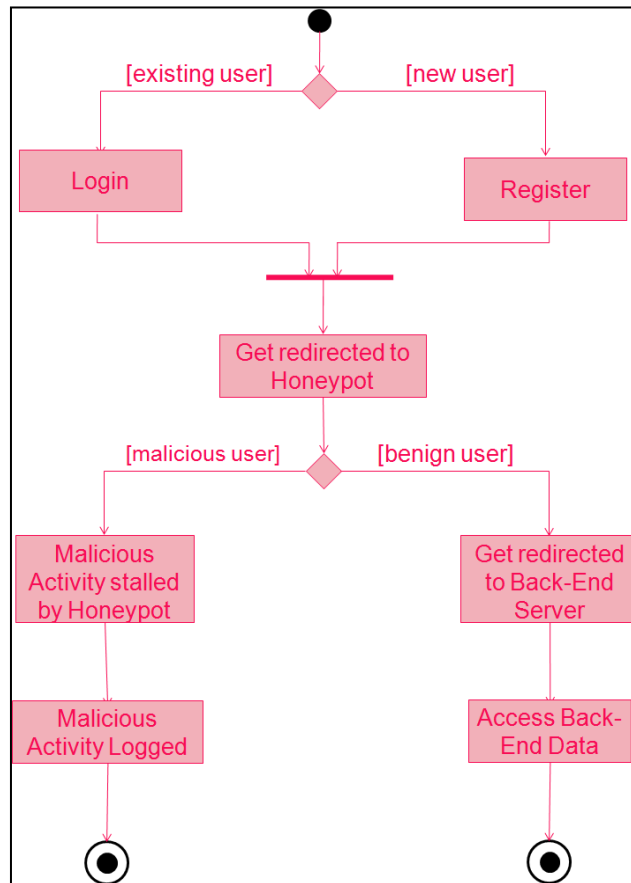
2.2 Sequence Diagram for Attacker



2.3 Sequence Diagram for Trusted User



2.4 Activity Diagram



3. Modules

3.1 Client

- This module is the POOH client. End User attempt to login to the Backend Server using this component.
- A new user can be registered using the “Register” Button which redirects to the Registration Module.
- A registered user can attempt access to the Backend Server by providing the proper credentials.

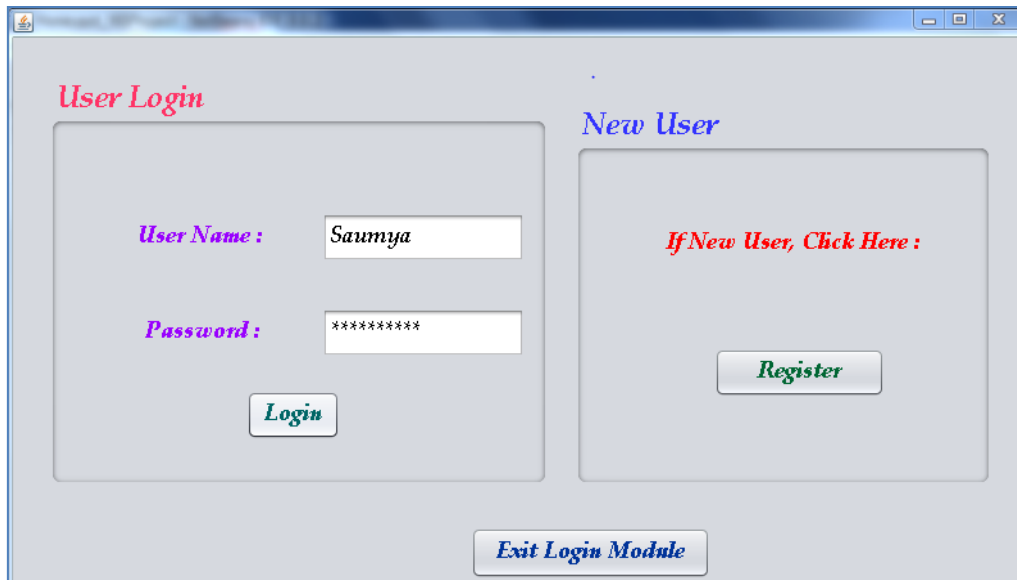


Fig 3.1 Client Module

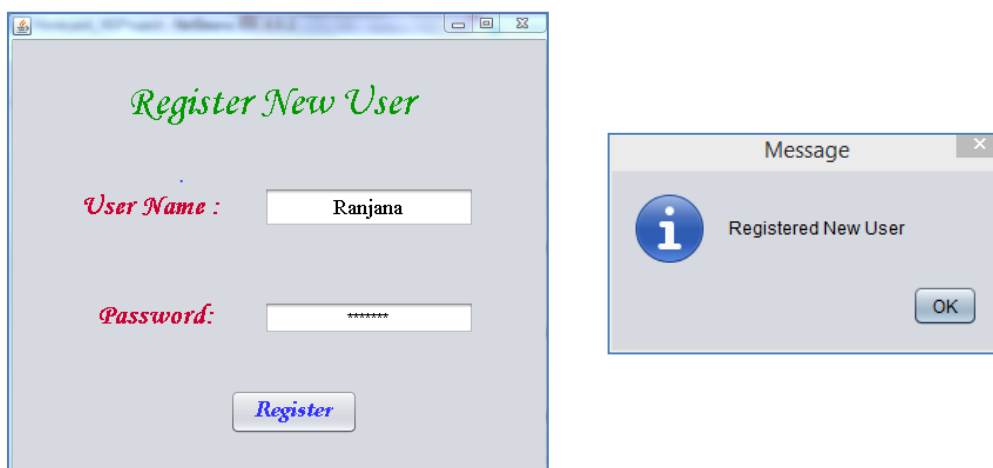


Fig 3.2 Login and Registration of New User

3.2 POOH Server (Honeypot)

- This module captures the user credentials and the activity of the user.
- Firstly, the system verifies them against the registered user database.
 - If there is a username/password mismatch appropriate error message is displayed.
 - If the user is not registered, the error message generated is "Invalid User".
 - If proper credentials are supplied, then the activity of the user is checked in the next step.
- The second step of granting access to any user is checking the activity generated by the user.
- This activity checking has two steps as follows:
 - Check if the login instance is opening multiple TCP connections on the honeypot machine. The threshold for detecting an attack was determined empirically for our

- set-up. The value finalised on was 50 open TCP connections. This step detects only SYN flooding attacks.
- If the preliminary check was passed, i.e. an “Operations are Normal” was identified, then the system intelligently detects the presence of an attack, and the Attack Type, if one is found.
 - All Login Activity is logged into the log file, Attack_Log.txt with the general format as:
 - Timestamp of the Login Attempt.
 - Numbers of TCP Connections open.
 - Source IP of the Login Attempt.
 - If the activity of the user is found to be Malicious, then the user is blocked out of the system, the entire details of the source of attack are logged into the log file along with the general format as specified below:
 - Attack Type
 - Attack Group
 - Attacker’s IP Address
 - If the User is Benign, then user is granted access to the backend server, where he can perform the operations on the backend data.
 - Screenshots of all elements related to this module are attached in the Appendix under the header Server Side Screenshots.
 - The Output which is shown on the user end on detection of an Attack is:

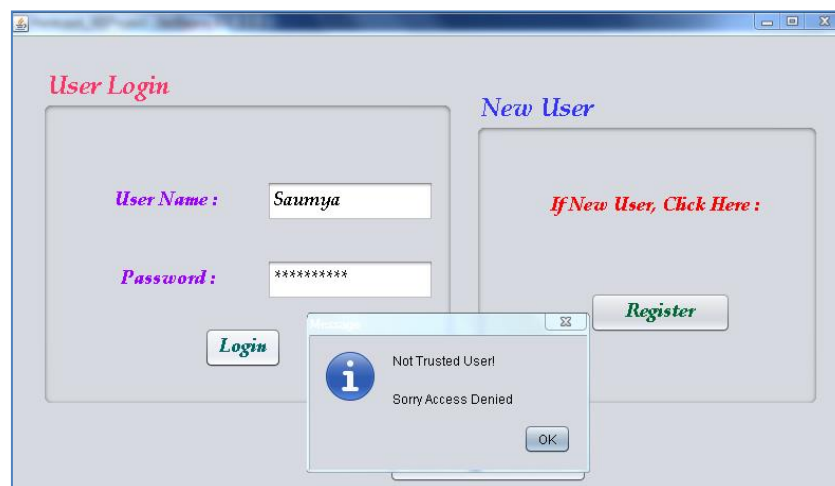


Fig 3.3 Attack Being Blocked by Honeypot

3.3 Backend Server

- Contains the backend data.
- Protected by Honeypot
- No interaction with the Attacker.

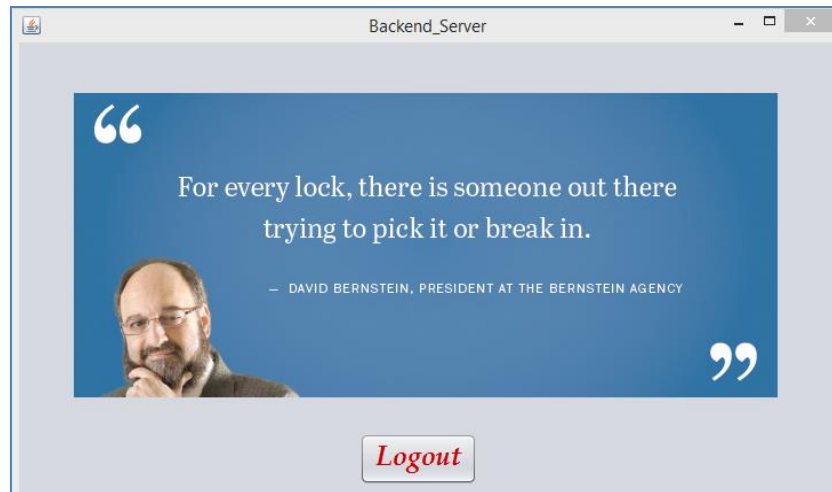


Fig 3.4 Backend Data, if User is Benign

4. Resources Utilized

There were several tools which were used in the course of building this system. To contribute to the novelty of our project, minimal external resources have been utilized. We have fruitfully attempted to create a majority of the employed tools ourselves. We have used the Apache POI library, for manipulating various file formats based upon the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2). Apache POI is your Java Excel solution (for Excel 97-2008).

4.1 Apache POI

- A major use of the Apache POI API is for Text Extraction applications such as web spiders, index builders, and content management systems. In our use case, it is used to handle the data being read from/written to XLSX files for the purpose of Attack Detection and Blocking of Malicious Users. There were two classes in the Apache POI library which were extensively used in the project:
 - **XSSF:** This class's instance was used to perform the read/write functions and parsing of the XLSX files.
 - **HSSF:** This class's instance was used to create the CSV file, Machine_Python.csv, which contained the input for the Learning Agent.
- Overview to these Classes:
 - HSSF is the POI Project's pure Java implementation of the Excel '97(-2007) file format. XSSF is the POI Project's pure Java implementation of the Excel 2007 OOXML (.xlsx) file format. HSSF and XSSF provide ways to read spreadsheets create, modify, read and write XLS spreadsheets. They provide:
 - Low level structures for those with special needs
 - An eventmodel API for efficient read-only access
 - A full usermodel API for creating, reading and modifying XLS files
 - To read the spreadsheet data, eventmodel API in either the org.apache.poi.hssf.eventusermodel package, or the org.apache.poi.xssf.eventusermodel package, is used depending on the file format.

- To modify the spreadsheet data then the usermodel API is used. Note that the usermodel system has a higher memory footprint than the low level eventusermodel, but has the major advantage of being much simpler to work with. Also, the new XSSF supported Excel 2007 OOXML (.xlsx) files are XML based, the memory footprint for processing them is higher than for the older HSSF supported (.xls) binary files.
- JAR files utilised:
 - poi-3.15.jar
 - poi-ooxml-3.15.jar
 - poi-ooxml-schemas-3.15.jar
 - xmlbeans-2.6.0.jar

4.2 Apache Commons

- Apache Commons is an Apache project focused on all aspects of reusable Java components.
- The Apache Commons project is composed of three parts:
 - The Commons Proper - A repository of reusable Java components.
 - The Commons Sandbox - A workspace for Java component development.
 - The Commons Dormant - A repository of components that are currently inactive.
- The Apache Commons source code repositories are writable for all ASF committers. While Apache Commons is a Commit-Then-Review community, we would consider it polite and helpful for contributors to announce their intentions and plans on the dev mailing list before committing code. All contributors should read our contributing guidelines.
- JAR files utilised:
 - commons-codec-1.10.jar
 - commons-collection-4-4.1.jar
 - commons-logging-1.2.jar
 - junit-4.12.jar
 - log4j-1.2.17.jar

5. Intelligent Intrusion Detection Using Machine Learning

Machine learning (ML) is a subfield of artificial intelligence that evolved from the study of pattern recognition and computational learning theory. In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed". Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look. It explores the study and construction of algorithms that can learn from and make predictions on data. Such algorithms operate by building a model from an example *training set* of input observations in order to make data-driven predictions or decisions expressed as

outputs, rather than following strictly static program instructions. A computer program is said to learn from experience with respect to some class of tasks and performance measure if its performance at tasks, as measured, improves with experience.

5.1 Learning Model

After testing a variety of methods for classification, Random Forests were found to be the most suitable for this project. **Random forests** or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

5.2 Dataset

The NSL-KDD dataset has been used for the task of training our model. The training data consists of 5 million records (4 GB). The dataset consists of live traffic recorded over a period of a week and consists records of the following types of attacks.

- Denial of Service Attack (DoS):
 - back
 - land
 - neptune
 - pod
 - smurf
 - teardrop
- User to Root Attack (U2R): unauthorized access to local superuser (root) privileges
 - buffer_overflow
 - loadmodule
 - perl
 - rootkit
- Remote to Local Attack (R2L): unauthorized access from a remote machine
 - ftp_write
 - guess_passwd
 - imap
 - multihop
 - phf
 - spy
 - warezclient

- warezmaster
- Probing Attack: surveillance and other probing
 - ipsweep
 - nmap
 - portsweep
 - satan

5.3 Performance

The model performed with a 99.8% accuracy on the live test data.

REFERENCES

Alata, E., &Nardi, L. Modelling attack processes on the Internet, based on honeypot collected data.

López, M. H., &Reséndez, C. F. L. (2008). Honeypots: basic concepts, classification and educational use as resources in information security education and courses. In Proceedings of the Informing Science and IT Education Conference.

Peter, E., & Schiller, T. (2011).A practical guide to honeypots. Washington Univerity.

Spitzner, L. (2003). Honeypots: tracking hackers(Vol. 1). Reading: Addison-Wesley.

Tavallaee, M., Bagheri, E., Lu, W., &Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009.

Zhang, F., Zhou, S., Qin, Z., & Liu, J. (2003, August). Honeypot: a supplemented active defense system for network security. In Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on (pp. 231-235). IEEE.

<http://www.wikipedia.org>

<https://www.kaggle.com/>

<https://poi.apache.org/>

<http://www.listendata.com/2014/11/random-forest-with-r.html>

<https://www.sans.org/security-resources/idfaq/what-is-a-honeypot/1/9>

<https://commons.apache.org/>

APPENDIX

A. Client Module

```
Output
Honeypot_NSProject (run)  Honeypot_NSProject (run) #2  Honeypot_NSProject (run) #3

run:
Successful credentials.
Socket :Socket[addr=/127.0.0.1,port=6876,localport=14995]
Data sent is:Saumya Veritasoft
Username sent to the server
BUILD SUCCESSFUL (total time: 5 minutes 40 seconds)
|
```

B. Server (Honey Pot)

```
Output - Honeypot_NSProject (run)

run:
Server Started and listening to the port 6876
Client Socket Received is:/192.168.43.126
Netstat Output File is present at F:\College Work\7th Sem\NS\Honeypot_NSProject\src\Resources\Netstat_Output.txt
Operations are normal.
Attack Log File is present at F:\College Work\7th Sem\NS\Honeypot_NSProject\src\Resources\Attack_Log.txt
Successful credentials.
Traffic Data is: 2 row.
Data written in Sample File successfully.
Enter continue to continue or exit if file transfer is done :
continue
Server Started and listening to the port 3000
Server ready .....
In Util
Machine_Python.csv file sent .....
Server Started and listening to the port 3001
The data of the file is:[B@534df152
The data of the file is:true,Probe,IPsweep Probe

ML_Output.txt file received .....
Enter continue to continue or exit if file transfer is done :
exit
-----
Attack Detected!!!!!! :(
Attack Details are logged into F:\College Work\7th Sem\NS\Honeypot_NSProject\src\Resources\Attack_Log.txt
```

C. Attack Detected by the HoneyPot, consequent Logging of the same

```
-----
Thu Nov 10 15:50:59 IST 2016
1. TCP connections are: 37
2. TCP + CLOSE_WAIT connections are: 0
3. TCP + TIME_WAIT connections are: 0
4. Source IP of Login by User is: 192.168.43.126:1395

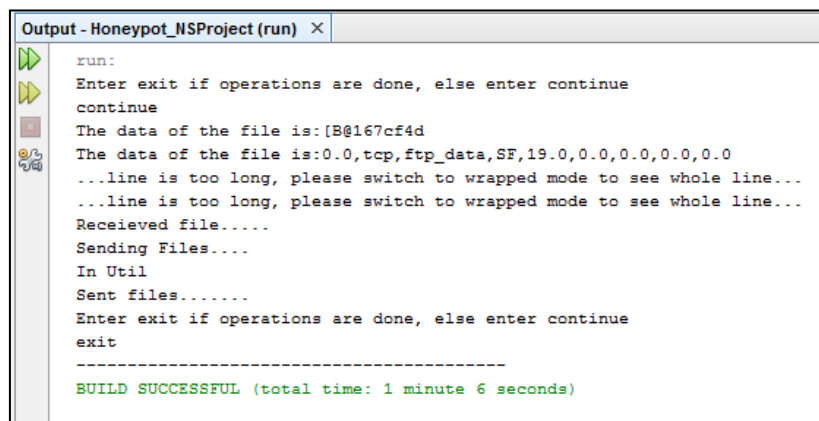
Attack was Detected. The details are:
1.Attack Type: Probe
2.Attack Group: IPsweep Probe
3.IP Address: 192.168.43.126:1395
```

D. Attack Log

```
-----
Thu Nov 10 15:50:59 IST 2016
1. TCP connections are: 37
2. TCP + CLOSE_WAIT connections are: 0
3. TCP + TIME_WAIT connections are: 0
4. Source IP of Login by User is: 192.168.43.126:1395

Attack was Detected. The details are:
1.Attack Type: Probe
2.Attack Group: IPsweep Probe
3.IP Address: 192.168.43.126:1395
```

E. Load Balancing Server for Honeypot



```
Output - Honeypot_NSProject (run) X
run:
Enter exit if operations are done, else enter continue
continue
The data of the file is:[B@167cf4d
The data of the file is:0.0,tcp,ftp_data,SF,19.0,0.0,0.0,0.0,0.0
...line is too long, please switch to wrapped mode to see whole line...
...line is too long, please switch to wrapped mode to see whole line...
Receieved file.....
Sending Files....
In Util
Sent files.....
Enter exit if operations are done, else enter continue
exit
-----
BUILD SUCCESSFUL (total time: 1 minute 6 seconds)
```