Integration of Intrusion Detection System with Mobile Cloud Offloading to Securely Offload Mobile Applications to the Cloudlet

[*]Abeselom Befekadu

Abstract

Nowadays mobile device has become a useful tool to compute the day to day activities of human being. However, this tool lacks the resources to handle the increasing need for computing intensive tasks. To overcome those limitations, the computation offloading mechanism has been widely considered. Security measures using encryption, user authentication, password, firewall and digital certificates are not enough to secure the data during transmission and the code itself. To reduce the risks behind mobile application offloading, this research proposes an integrated intrusion detection system with mobile application offloading to securely offload mobile computation to the cloudlet. The system denies the service request of the mobile device during intrusion. This means the system stops the offloading process immediately at the time of attack. This process secures the client's data from being accessed by unauthorized intruder. Moreover, the system will generate report to the cloudlet administrator about why the service request of the client is denied. After removing the intruder from the network the services will start immediately.

General Terms:

Mobile Cloud Computing, Network Security, Machine Learning, Data mining

Keywords

Mobile Cloud Computing, Computation Offloading, Intrusion Detection in cloud, Stop offloading, Notify the cloudlet administrator.

Introduction

Mobile cloud computing is a new paradigm that uses cloud computing resources to overcome the limitations of mobile computing[1]. Computation off-loading sends bulky computations and difficult processing from mobile device to the resourceful cloud servers. This contributes to efficient usage of battery, memory, storage and CPU of the mobile device. There are two core sources of vulnerability in computation offloading [2]. The first one is the propagated vulnerability, which happens when the possibility of an

[*]Lecturer, Assosa University, Assosa, Ethiopia abeselom2010@gmail.com

attack initiates from other objects and propagates to the object over call interactions. The other one is cloud originated vulnerability, which is triggered by relationship between an object of the mobile application and the cloud server which accommodates it. In order to minimize the risk of being caught by the attackers during transmission, there is a need to secure these objects by using some security mechanisms. Among the security mechanisms, intrusion detection system is the more popular and effective means of security.

1.1 Scope and Limitations of the Study

The integrated system has the following aspects:

- Identification of possible intrusions
- Report generation about the intrusions
- Reading the generated report before accepting service requests from mobile device
- Denying service request if there is an attack on the cloudlet network
- Alarm the cloudlet administrator why the service is denied.

The integrated system has the following limitations:

- The research only focused on the intrusions that were caused by network based vulnerabilities.
- To identify the computation intensive methods, the system used the previous execution time and call relationship of the methods.
- The designed solution was not 100% accurate. There was false positive and false negative in the result.
- The experiment was conducted only on android based mobile device.
- The research only considered attacks which were found on NSL-KDD datasets.

1.2 Methodology

This research was expected to improve the limitations of the existing intrusion detection system in cloud computing environment. To achieve the main objective of this research, design-oriented research methodology was used. Design-oriented research method is basically a problem-solving model [3]. The five steps of design oriented method followed in this research work were problem identification, solution suggestion, development of the system, evaluation and conclusion.

I. Problem Identification

The first thing to do was to conduct a comprehensive review of literatures to acquire a deeper understanding of the research area and its problem domains. Through this literature the researcher identified the importance and limitations of the previous works done in the area of intrusion detection in mobile cloud computing. Existing works related to this research work were assessed to identify and point out direction in providing solution to identified problems. The problems that were identified in this research were:

- The performance problem which was caused by deploying computation intensive intrusion detection system on the mobile device.
- The communication vulnerability problem which happened when an attacker compromised the communication channel between the mobile device and cloudlet server and made the system to look like normal operation time.
- The problem around the time gap between detection of intrusion and taking care of the intrusion by cloudlet administrator.
- The problem around the need for high network bandwidth to create the replica of mobile device on virtual machines of the cloud server in which intrusion detection was executed.

II. Solution Suggestion

The second phase of this research was to determine different solutions for each research problems. The suggested solutions were:

- Deploying the heavy computing components of the system in the cloud environment.
- Deny service request by the mobile device during attacks and make the system to compute tasks locally.
- Alert the cloudlet administrator about the attacks.
- Using the round trip time (RTT) value in offloading decisions.

III. Development of the System

The integrated network based intrusion detection system with computation offloading engine was developed to improve the performance, communication vulnerability and time gap between detection and taking action problems of the existing systems. The system used combined algorithm which was Naïve Bayes and Decision Tree for classifying and learning process of the intrusion detection. Lighter offloading decision engine was developed for the client side to reduce the computation overhead. The offloading process of the system offloaded application at method level granularity. For communication between mobile device and cloudlet server, the system used remote procedure call (RPC) technology. To partition the application, the system used critical path method. For integration, the network based intrusion detection module generated report and the report was used by the cloudlet server to take action.

IV. Evaluation and Improvement

Finally, the integrated system was implemented and evaluated by using the integrated intrusion detection system with computation offloading architecture. A mobile application that used image processing algorithms to apply some effects was used to evaluate the offloading engine, and the NSL-KDD testing data set was used to evaluate the network based intrusion detection system. To test the integration, the response of the system during attack was observed.

V. Discussion and Conclusion

The results which were found during evaluation phase and the achievements of the integrated system were discussed. The success that made the system to achieve the objectives of the research was deliberated. Finally, the findings of the research were summarized.

1.3 Tools and Techniques

Eclipse Neon was used to develop the prototype. This version of Eclipse supports Java Platform Standard Edition (Java SE) specification Version 8.2 with Java Development Kit (JDK) 8.2 and Java SE Runtime Environment (JRE) 8.2. In order to build, test and debug applications for Android, the Android SDK was installed. The researcher also installed the ADT plugin on Eclipse Neon to use the packages which allowed the creation of android projects, used the emulator to test the application and for designing the user interface. The java programming was also used as a programming language during development. Waikato Environment for Knowledge Analysis (WEKA) was a toolkit which was composed of machine learning and data mining. This toolkit can be used as it is or it can be called from Java code. This API was called from java code for the purpose of data mining and machine learning during developing the network based intrusion detection system. Because of cost, the researcher decided to use Connectify Hotspot to connect the mobile device and the cloudlet during the development of the system, which turned a computer with wireless adapter into a wireless router. Maven was utilized to manage the different parts of the system and their dependency.

2. Related Works

Substantial numbers of researches have been done and several solutions have been proposed to improve the performance of network intrusion detection system in mobile cloud computing environment. However, none of them have considered the computation offloading. This research has considered the security of mobile cloud computing from the computation offloading perspective.

Portokalidis et al [4] portray a scheme that executes an intrusion detection for Android based system called Paranoid Android. Paranoid Android depends on a cloud organization model where intrusion detection is offered as a package. By imitating the entire device in virtual machines in the cloud, it is conceivable to apply asset concentrated anomaly detection mechanisms. This would not be possible to do on smart phones due to the exceptionally limited accessible resources. The clone is kept in a state of harmony with the smart phone. Activities performed on the device are replayed by the emulator and also where the security checks are implemented using the emulator's data. They additionally demonstrate that it is basic to improve the synchronization and tracing processes because collecting and transmitting information can prompt unbalanced exhaustion of the battery. Despite this, the system uses extremely loose synchronization model where the device synchronizes with its replica only when it is recharging. After an attack occurs the phone has to plugged-in to personal computer to get back to the normal state.

Manish Kumar and Dr. Hanumanthappa [5] propose a cloud based Intrusion Detection System for smart phones to reverse the issues of smartphone resource requirements and to identify any trouble making or abnormal movement adequately. It comprises of a cloud-based administration which would enable clients to install a lightweight agent on their smartphone and enlist to an online cloud-service by indicating their working platform, applications introduced on their telephone and other important data about their smartphone. A while later, this particular smartphone is copied in a virtual machine on the cloud utilizing an intermediary proxy which copies the incoming traffic to the device and forwards the traffic to the emulation platform, where intrusion detection is executed. In this paper, the authors failed to specify how the intrusion detection engine communicates with the client host agent and how to handle the much needed mobile resources during creating the replica of the device.

Fang Yuan, Lidong Zhai, Yanan Cao and Li Guo [6] proposed an intrusion detection system for identifying anomaly activity on Android smartphones. The intrusion detection system constantly screens and gathers the data of smartphone under ordinary conditions and attack state. It extracts different features acquired from the Android framework, for example, the network traffic of smartphones, battery utilization, CPU use and the measure of running processes. At that point, it applies Bayes classifying algorithm to decide if there is an attack. Analysis of the network traffic, CPU usage, the amount of running processes, and so on is carried out on the device. But, this kind of deployment hurts the performance of the smart phone.

Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer and Yael Weiss[7] proposed a framework which recognizes a Host-based Malware Detection System that continuously screens different components and events which is acquired from the phone and afterward applies machine learning anomaly identifiers to group the gathered information as normal (benign) or abnormal (malicious). The researchers did not demonstrate their proposed solution on the data which was obtained from the real world. Since no malicious applications were yet accessible for Android, they created four malicious applications, and assessed Andromaly's capacity to distinguish new malware in view of samples of known malware. They evaluated a few blends of anomaly detection algorithms; included feature selection technique, and the number of top features to discover the mix that yielded the best execution in distinguishing new malware on Android.

3. Design of the System

This system viewed the security of mobile cloud computing from the perspective of computation offloading since the main aim and purpose of mobile cloud computing was to relieve the mobile device resource limitations by offloading computation-intensive mobile applications to the cloudlet server. The following figure shows the high level architecture of the system.

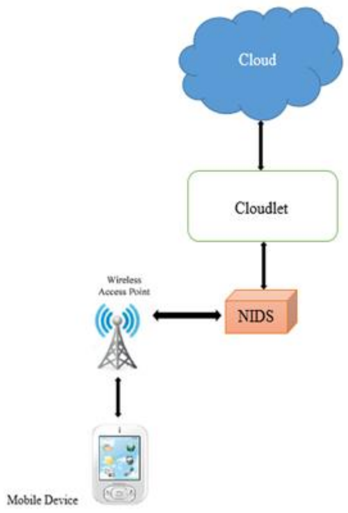


Fig 1: High Level Architecture of the Integrated System

3.1 Detailed Architecture of the System

The system architecture consisted three modules which were the mobile module, the network based intrusion detection system module, and the cloudlet server module. The mobile module and cloudlet server module consisted of four layers, several components and subcomponents in each layer. Each of those components and subcomponents performed specific tasks and had different functionalities. The network based intrusion detection system module also had several components and functions. The architecture of the system is depicted in the Figure 2.

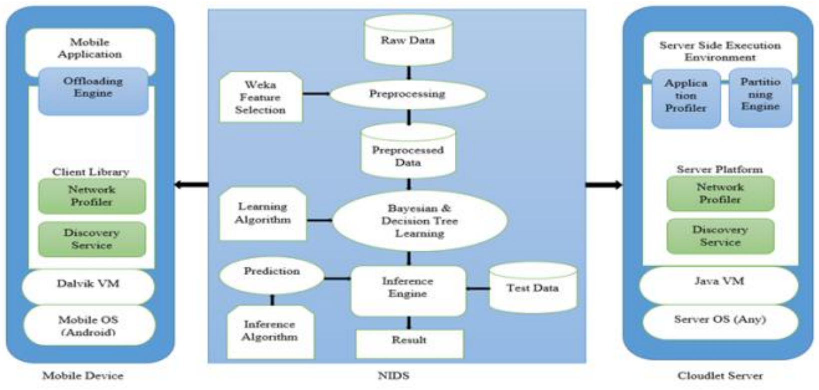


Fig 2: Architecture of the Integrated System

3.1.1 Mobile Module Components of the System

I. Discovery Service

The discovery service is responsible for finding a running cloudlet server within the same network such as WLAN, WIFI, and base station. This component was also responsible for creating a communication session with the server. At the time of launching mobile application, the mobile broadcasted its presence to wireless network by multicasting message. By using the server's platform listening service in the wireless network, the server listen the multicast message and replayed a message back to the mobile device with the services that are available for the mobile application. The component was taken from a Multi-Platform Offloading System (MpOS) framework[8] with some minor adaptation for the purpose of integration with the system.

II. Network Profiler

This component of the system uses the round trip time (RTT) to monitor and measure the quality of the network which is available between the mobile device and the server. A ping request is sent to the cloudlet server and the round trip time (RTT) was measured. The result is used by the offloading engine as parameter for the purpose of making a decision about whether to offload a method or not to offload to a remote server.

III. Offloading Engine

The offloading engine performs decisions whether to offload the execution to the cloudlet or execute. It is locally based on parameters from the network profiler and partition priority of the methods given by partitioning engine. The offloading engine is the heart of the engine which decides which code to offload. It makes an estimate of round trip time (RTT) and decides to offload only when a certain criterion is met. The estimation is carried out only if a connection exists and a server is available. If no server exists, estimation is redundant since it will be executed in the android runtime whatever the cost. The round-trip time is defined as the elapsed time between the instant packet is sent by the source node to the instant corresponding ACK packet is received by the source node [9]. The system uses the round trip time (RTT) for offloading decisions since it requires simple computation and easy to analyze. Moreover, using the round trip time (RTT) for offloading decisions reduce the computation overhead encountered because of analysis on the resource constraint mobile device. For offloading decisions different RTT boundaries are used by various scholars. In [8] offloading is executed when RTT is lower than 40 ms (40 millisecond). However, Tseghai et al [10] observed offloading decisions for different RTT values ranging from 10 m/s to 100 m/s. Based on the result, computation offloading was worthwhile when RTT is lower than ≈32ms. Since this RTT value is better than the other's result, the system uses it for offloading decisions. Specifically the system performs offloading when RTT is lower than 32ms. The Pseudo code of the computation offloading algorithm is described as follows.

```
Input: Method, RTT, Method_Annotation, Partition_Priority, Partitions
Process:
BEGIN
If RTT < = 32 && Method_Annotation = = Remotable Then
   Offload the method to remote server
Else If RTT < = 32 Then
   Compute RTT difference (RTTD = 32 - RTT)
   Sort the partitions ascending depending on Partition_Priority
      For each Partition Do
         If Partition_Priority < = RTTD Then
            Offload the partition to remote server
         End If
      End For
Else
Execute the method on the mobile device
End If
END
Output: Perform offloading if it is worthy else execute the method locally
```

Pseudo code 1: Pseudo code for offloading engine

3.1.2 Cloudlet Server Module Components of the System

I. Application Profiler

The main responsibility of this component is to estimate the execution time of the methods. In offloading, decision making on whether to offload or not is based on the estimated android runtime, estimated offloading time and estimated server runtime. To offload a method to the remote server the estimated server runtime must be less than the addition result of android runtime and estimated offloading time. The system deploys application profiler and partitioning engine components on the server despite most of the other systems which deploy them on the mobile side. This means of component deployment allows the mobile device side to reduce the overhead encountered during application profiling and partitioning. The java instrumentation API is used to develop both application profiler and partitioning engine components. Accessing a class that is loaded by the Java Class Loader from the JVM and modifying its byte code at runtime become easy by using java instrumentation API. The application profiler uses the call graph to determine the cost of execution of each method and the number of calls that is happened between each method. The application partitioning engine uses this cost to partition the application later.

II. Partitioning Engine

Partitioning engine partitions the mobile application based on the application profiler output result. This component uses critical path method to identify the longest path in the call graph which is generated by the application profiler. The application partitioning engine algorithms identify the first critical path which is the longest path from the start node to the exit node in the call graph. The length of a path is computed as the sum of the execution time of the methods and the number of calls along that path. After this process the engine partitions, the methods in the first critical path, assign partition priority for each method and offload it to the remote server on next execution. The engine doesn’t stop on the first critical path instead it will continue by identifying the next critical path by pruning the first critical path from the call graph. The next figure demonstrate example of the first critical path of the call graph.

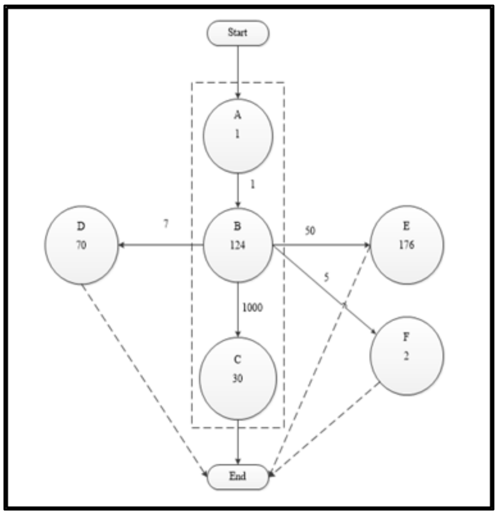


Fig 3: The First Critical Path in the Call Graph

3.1.3 Network Based Intrusion Detection System Module Components

1. Raw Data

The input data for training phase was the offline dataset which was found on the web for educational research purpose. It was labeled dataset that could be easily learnt by the system. For this case, the research used simulated dataset called NSL-KDD for this phase. This dataset was selected because it was the latest version of all simulated dataset in the area of network security; redundant records were eliminated from training set and it was affordable to use for experiment purpose as it consisted of reasonable number of instances both in the training and testing set[11].

I. Preprocessing

Data preprocessing was required to remove unwanted attributes from the dataset and build a dataset for Naïve Bayes and Decision Tree algorithms. Dataset feature extraction was analyzed based on the attacks nature and extra domain information. The preprocessing phase was responsible for preparing the NSL-KDD dataset for the next phase which was Naïve Bayes and Decision Tree learning process. In this research, WEKA attributed filtering was used with other pre-processing techniques. The operations such as attribute selection, attribute filtering and instances filtering were applied in this phase. Those techniques improved the efficiency of the algorithm to classify the data correctly.

II. Naïve Bayes and Decision Tree Learning

Today's network environment is very crowded and uncertain. Detecting intrusions in this uncertain network environment is very difficult. Finding the best algorithm is a very challenging task in developing network based intrusion detection system. This research compared and contrasted every algorithm that was used to develop intrusion detection system and finds the combination of Naïve Bayes and Decision Tree algorithms are the best way to develop the system.

Pseudo code 2: Pseudo Code for Training Phase

III. **Inference Engine**

After the Naïve Bayes and Decision tree model were built or trained by the network traffic dataset and ready to predict attacks in the incoming network traffic, the Inference Engine provided the predicting algorithms with test data which was used to compare with the learnt knowledge. The Inference Engine also classified each record of the input data (Test Dataset) to normal connection or to a relevant attack based on the Naïve Bayes and Decision tree model. The integration of the algorithms happened after both algorithms made prediction. Then when both made same prediction, the integrator code took the result but when they predicted different, the integrator code took the worst possibility which was the anomaly.

```
Input: Test Dataset
Process:
BEGIN
Read test dataset
Call Naïve Bayes and Decision Tree to classify as
anomaly or normal
If status is anomaly then
Sent report
End If
```

Pseudo code 3: Pseudo code for detection stage

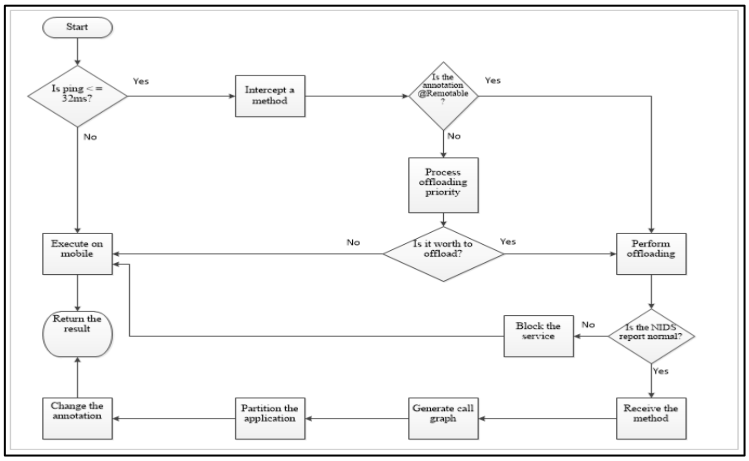


Fig 4: Execution flow of the system of the system

3.2 Communication Model of the System

To allow communication between the mobile device and the cloudlet server this research used the conventional client-server communication model. Because of its support to Dalvik virtual machine, language independence and best performance, this research chose RPC over RMI.

3.3 Execution Flow of the System

There are different ways of partitioning an application during offloading. The partitioning can be done on different levels such as thread level, method level, task level, object level, class level, module level and the whole application level. Anyone can choose from this level as per their system specification and compatibility issues. This research considered the compatibility issue with the RPC communication technique and decided the offloading process at method level was the best since RPC supports only method level remote calls. Specifically, the system offloaded computations at method level granularity.

The system started its implementation by examining the network quality. If the network quality meets the standard (The ping result < = 32ms), the system intercepts a method which is in execution locally and starts the process of offloading. But at the server side, the server receives the offloaded methods based on the report produced by the network based intrusion detection system (NIDS). If the report by the NIDS contains attacks, the server immediately denies service for the mobile device and makes the methods to be executed locally. After appropriate measures are taken by the cloud administrator, the services of the cloudlet server will start immediately.

4. Implementation and Evaluation

4.1 Implementation of the System

The system was developed and implemented for android mobile device and the intrusion detection was implemented for network based attacks. The Android API 16 (Android 4.1.2 Jelly Bean) was used as the target system for the development of the application. The WEKA API was also used for data mining and machine learning purpose in developing the intrusion detection system.

3.1.1 Configuration of the System

In order to use the offloading system, the mobile application had to be configured inside the code and the cloudlet file must be configured. The configuration is depicted below:

- The methods with computation intensive operation can be annotated with @ Remotable annotation. The annotation must be implemented by using interfaces. The interfaces allows the system to create proxy of objects in which enables the system to access the methods [8].

- The IP addresses of the cloudlet must be configured manually and the configured file, the server platform jar and the application profiler and partitioning jar file must be in the same folder.

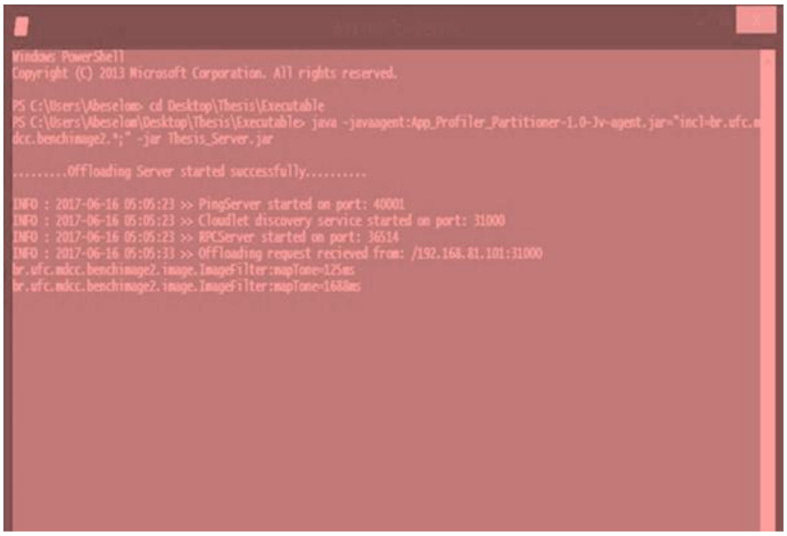
- The client library must be referenced as a library by the mobile application.

4.2 Evaluation of the System

The evaluation of the system was viewed from the perspectives of identification of network based cyber-attacks and deterrence of data theft by stopping the computation offloading process immediately at the time of network based intrusions. The system used NSL-KDD datasets for simulating the detection of intrusion and a computation-intensive mobile application called Bench Image was used for testing the offloading engine.

3.2 Results and Discussion

The experiments were broadly divided into two parts. The first part consisted of the experiment which was done on network based intrusion detection. The second part consisted of experiments performed on integration of the network based intrusion detection system with mobile application offloading to securely offload mobile computation to the cloudlet server.



```
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\Abeselom> cd Desktop\Thesis\Executable
PS C:\Users\Abeselom\Desktop\Thesis\Executable> java -javaagent:App_Profiler_Partitioner-1.0-Jv-agent.jar="incl=br.ufc.m
dcc.benchimage2.*;" -jar Thesis_Server.jar

.........Offloading Server started successfully..........

INFO : 2017-06-16 05:05:23 >> PingServer started on port: 40001
INFO : 2017-06-16 05:05:23 >> Cloudlet discovery service started on port: 31000
INFO : 2017-06-16 05:05:23 >> RPCServer started on port: 36514
INFO : 2017-06-16 05:05:33 >> Offloading request recieved from: /192.168.81.101:31000
br.ufc.mdcc.benchimage2.image.ImageFilter:mapTone=125ms
br.ufc.mdcc.benchimage2.image.ImageFilter:mapTone=1688ms
```

Fig: Handling of Service Request by the Cloud Let Server

4.3.1 Experiment on Network Based Intrusion Detection System

The performance of the network based intrusion detection system was evaluated based on the accuracy, precision, recall, True Positive Rate (TPR) and False Positive Rate (FPR). To measure the performance of the NIDS a standard metrics which were confusion matrix values were used. The effectiveness of network based intrusion detection system was measured in terms of accuracy in which it identified how much did the IDS classified the incoming packet as normal and attack. The accuracy of the system was calculated using the following equation.

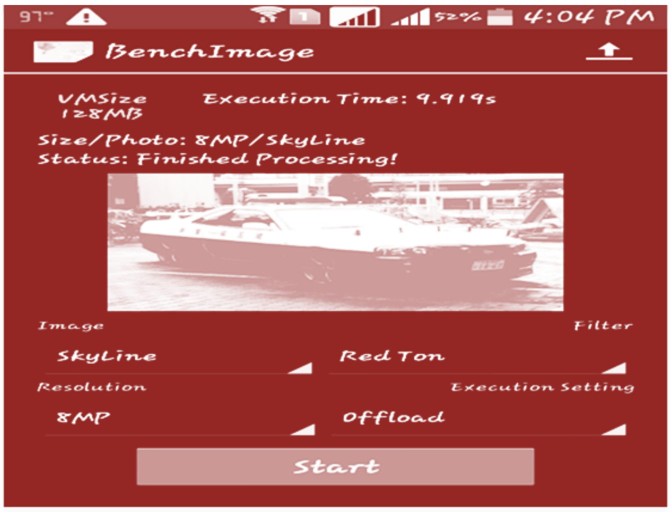
$$\text{Accuracy} = \frac{(\text{correctly predicted} * 100)}{(\text{correctly predicted } + \text{ incorrectly predicted})}$$

This researcher first calculated the accuracy of each algorithm which was Naïve Bayes and Decision Tree. After that the researcher compared the result with the combined algorithm. For both algorithms the researcher used 22544 instances. The experiment got 17160 correctly predicted and 5384 incorrectly predicted instances for Naïve Bayes algorithm and 17913 correctly predicted and 4631 incorrectly predicted instances for Decision Tree algorithm after calculating the accuracy of each algorithm. The result for Naïve Bayes was 76% and for Decision Tree 79%. But in contrast to the results from the Naïve Bayes and Decision Tree algorithms the combined algorithm produced 18543 correctly prediction and 4001 incorrectly prediction. From this result the accuracy of combined algorithm was 82%. Therefore, the accuracy of the combined algorithm was better than that of the single algorithm.

4.3.2 Experiment on Integration of NIDS with Mobile Application Offloading

The main purpose of integrating network based intrusion detection system with mobile application offloading was to reduce the security risks encountered during offloading computation intensive part of mobile application to the cloudlet. Figure 5 demonstrate how the system handles the offloaded methods during the normal time of operations.

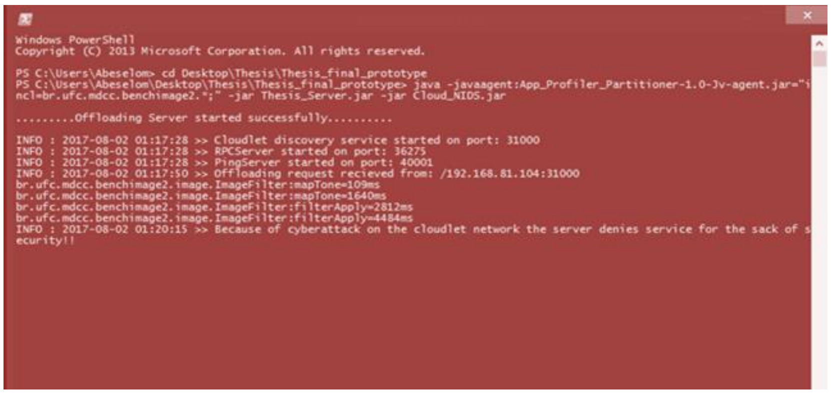
After receiving service request by the cloudlet server from mobile device, the cloudlet server processed the image and returned the result to the mobile application. The result of applying Red Tone filter on the image is shown in figure 6.



```
BenchImage
VMSize        Execution Time: 9.919s
128MB
Size/Photo: 8MP/SkyLine
Status: Finished Processing!
Image                                   Filter
SkyLine                 Red Ton
Resolution                    Execution Setting
8MP                     Offload
Start
```

Fig 6: The Result of Applying Red Ton Filter on the Image

When cyber-attack happened the integrated system responded by denying service request by the mobile device and closed the stream immediately. When the system denied service, the mobile application immediately processed the task locally. The system also showed information about why the request was denied for the purpose of alarming the cloudlet administrator. Moreover, after the removal of intruder from the network by the administrator, the services of the cloudlet started immediately. This mechanism secures the mobile application user from attackers (Figure 7) demonstrate how the integrated system responds during cyber-attack.



```
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\Abeselom> cd Desktop\Thesis\Thesis_final_prototype
PS C:\Users\Abeselom\Desktop\Thesis\Thesis_final_prototype> java -javaagent:App_Profiler_Partitioner-1.0-Jv-agent.jar="i
ncl=br.ufc.mdcc.benchimage2.";" -jar Thesis_Server.jar -jar Cloud_NIDS.jar

.........Offloading Server started successfully..........

INFO : 2017-08-02 01:17:28 >> Cloudlet discovery service started on port: 31000
INFO : 2017-08-02 01:17:28 >> RPCServer started on port: 36275
INFO : 2017-08-02 01:17:28 >> PingServer started on port: 40001
INFO : 2017-08-02 01:17:50 >> Offloading request recieved from: /192.168.81.104:31000
br.ufc.mdcc.benchimage2.image.ImageFilter:mapTone=109ms
br.ufc.mdcc.benchimage2.image.ImageFilter:mapTone=1640ms
br.ufc.mdcc.benchimage2.image.ImageFilter:filterApply=2812ms
br.ufc.mdcc.benchimage2.image.ImageFilter:filterApply=4484ms
INFO : 2017-08-02 01:20:15 >> Because of cyberattack on the cloudlet network the server denies service for the sack of s
ecurity!!
```

Fig 7: The Integrated System Responses during Attack

5. Conclusion

Intrusion detection in mobile cloud computing has been improving from time to time. However, these improvements have not been enough to secure the mobile users from attacks. When thinking about mobile cloud security, we have to consider two things. The first one is about losing the customers forever because of data theft by the intruders which happens by letting them to use the services of cloud during cyber-attacks and the second is gaining the confidence of the customers by denying services till taking care of the attacks by the administrator. This research argues that losing mobile cloud customers forever is the worst choice for the cloud providers.

Since cloudlets are deployed ones wireless hop away for the purpose of fast response time, they support only minimum number of clients as per the wireless capacity. So protecting the mobile cloud customers by denying service request until the cloud administrator removes the intruder is the best means of security for both customers and cloud provider because the integrated system affects only customers who use the attacked cloudlet. This security operation does not disrupt the activities of other clients who are using another cloudlet of the provider. The current intrusion detection system in the mobile cloud environment lacks these capabilities.

Finally, the integrated intrusion detection with mobile computation offloading system was evaluated and the obtained result shows the system can deter data theft by the intruders during attack. This shows that the system can be effective means of security for mobile cloud computing.

References

1. R. Matos, J. Araujo, D. Oliveira, P. Maciel, and K. Trivedi, “Sensitivity Analysis of a Hierarchical Model of Mobile Cloud Computing,” Elsevier B.V., 2014.
2. H. Zhu, C. Huang, and J. Yan, “Vulnerability Evaluation for Securely Offloading Mobile Apps in the Cloud,” IEEE, pp. 108–116, 2013.
3. A. Hevner, “Design Science in Information Systems Research,” MIS Q., Vol. 28, No. 1, pp. 75–105, 2004
4. G. Portokalidis and H. Bos, “Paranoid Android : Versatile Protection for Smartphones,” 2010.

5. M. Kumar and M. Hanumanthappa, “Cloud Based Intrusion Detection Architecture for Smartphones,” IEEE, 2015.
6. F. Yuan, L. Zhai, Y. Cao, and L. Guo, “Research of Intrusion Detection System on Android,” IEEE Ninth World Congr. Serv. Res., pp. 312–316, 2013.
7. [A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, “‘ Andromaly ’: A Behavioral Malware Detection Framework for Android Devices,” *J Intell Inf Syst*, pp. 161–190, 2012.
8. P. B. Costa, P. A. L. Rego, L. S. Rocha, and J. N. De Souza, “MpOS : A Multiplatform Offloading System,” ACM, pp. 577–584, 2015.
9. Q. Wang, “Restart in Mobile Offloading,” 2015.
10. A. Tseghai, “A Lightweight Computation Offloading System for Mobile Cloud Computing,” 2017.
11. D. Tigabu, “Constructing Predictive Model for Network Intrusion Detection,” 2012.