IJIRR

International Journal of Information Research and Review
Vol. 04, Issue, 12, pp.4810-4818, December, 2017

RESEARCH ARTICLE

INTEGRATION OF INTRUSION DETECTION SYSTEM WITH MOBILE CLOUD OFFLOADING TO SECURELY OFFLOAD MOBILE APPLICATIONS TO THE CLOUDLET

[1]Abeselom Befekadu and *,[2]Ravindra Babu

[1]Lecturer, Assosa University Assosa, Ethiopia
[2]Adama Science and Technology University Adama, Ethiopia

ARTICLE INFO

Article History:

Received 11th September, 2017
Received in revised form
06th October, 2017
Accepted 20th November, 2017
Published online 30th December, 2017

Keywords:

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

ABSTRACT

This Nowadays mobile devices have 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 deny 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 be started immediately.

General Terms: Mobile Cloud Computing, Network Security, Machine Learning, Data mining.

INTRODUCTION

Mobile cloud computing is a new paradigm that uses cloud computing resources to overcome the limitations of mobile computing (Matos, 2014). Computation offloading migrates bulky computations and difficult processing from mobile device to the resourceful cloud servers. This contributes the efficient usage of battery, memory, storage and CPU of the mobile device. There are two core sources of vulnerability in computation off loading Zhu, 2013. The first one is the propagated vulnerability, which happens when the possibility of an 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 accommodating it. In order to minimize the risk of being caught by the attackers during transmission, there is a need to secure this objects by using some security mechanisms. Among the security mechanisms intrusion detection system is the more popular and effective means of security.

Corresponding author: **Ravindra Babu,**
Adama Science and Technology University Adama, Ethiopia.

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 have the following limitations:

- The research only focuses on the intrusions that are caused by network based vulnerabilities.
- To identify the computation intensive methods, the system uses the previous execution time and call relationship of the methods.
- The designed solution is not 100% accurate. There is false positive and false negative in the result.
- The experiment is conducted only on android based mobile device.

- The research only considered attacks which is found on NSL-KDD datasets.

MATERIALS AND METHODS

This research is 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 (Hevner, 2004). The five steps of design oriented method followed in this research work are problem identification, solution suggestion, development of the system, evaluation and conclusion.

Problem Identification

The first thing to do is to conduct a comprehensive review of literatures to acquire a deeper understanding of the research area and its problem domains. Through this literature the research has 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 assessed to identify and point direction in providing solution to identified problems. The problems that were identified in this research are:

- The performance problem which is caused by deploying computation intensive intrusion detection system on the mobile device.
- The communication vulnerability problem which is happened when an attacker compromise the communication channel between the mobile device and cloudlet server and make 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 is executed.

Solution Suggestion

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

- 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.

Development of the system

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

Evaluation and Improvement

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

Design Oriented Research Methods

The results which are found during evaluation phase and the achievements of the integrated system has been discussed. The successes that makes the system to achieve the objectives of the research is deliberated. Finally, the findings of the research are summarized.

Tools and Techniques

Eclipse Neon is 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 apps for Android, the Android SDK is installed. The research also install the ADT plugin on Eclipse Neon to use the packages which allows the creation of android projects, use the emulator to test the application and for designing the user interface. The java programming is also used as a programming language during development. Waikato Environment for Knowledge Analysis (WEKA) is a toolkit which is 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 is 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 research decided to use Connectify hotspot to connect the mobile device and the cloudlet during the development of the system, which turns a computer with wireless adapter into a wireless router. Maven is utilized to manage the different parts of the system and their dependency.

Related Works

Substantial number 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 considers the security of mobile cloud computing from the computation offloading perspective.

Portokalidis *et al* (Portokalidis, 2010), 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 smartphones due to the exceptionally limited accessible resources. The clone is kept in a state of harmony with the smartphone. 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 occur the phone have to plugged-in to personal computer to get back to the normal state.

Manish Kumar and Dr. Hanumanthappa (Kumar, 2015) propose a cloud based Intrusion Detection System for smartphones 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 fails 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.

Fangfang Yuan, Lidong Zhai, Yanan Cao and Li Guo (Yuan, 2013), 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 are 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 (Shabtai, 2012) proposed a framework which recognize 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 is obtained from the real world. Since no malicious applications are 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, include feature selection technique and the number of top features to discover the mix that yields the best execution in distinguishing new malware on Android.

Design of the system

This system views the security of mobile cloud computing from the perspective of computation offloading since the main aim and purpose of mobile cloud computing is 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.

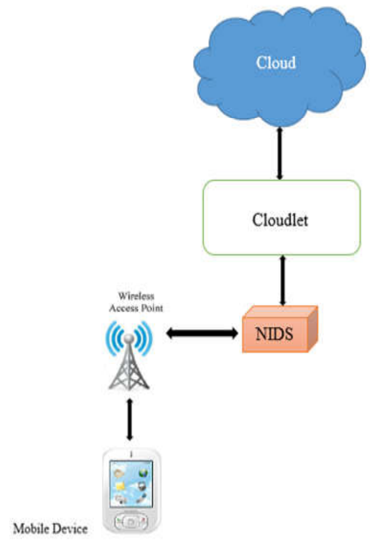

Fig 1. High level architecture of the integrated system

Detailed Architecture of the System

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

Mobile Module Components of the System

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 is also responsible for creating a communication session with the server. At the time of launching mobile application, the mobile broadcasts 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 replays a message back to the mobile device with the services that are available for the mobile application. The component is taken from A Multi-Platform Offloading System (MpOS) framework

(Costa, 2015), with some minor adaptation for the purpose of integration with the system.

uses it for offloading decisions. Specifically the system performs offloading when RTT is lower than 32ms.

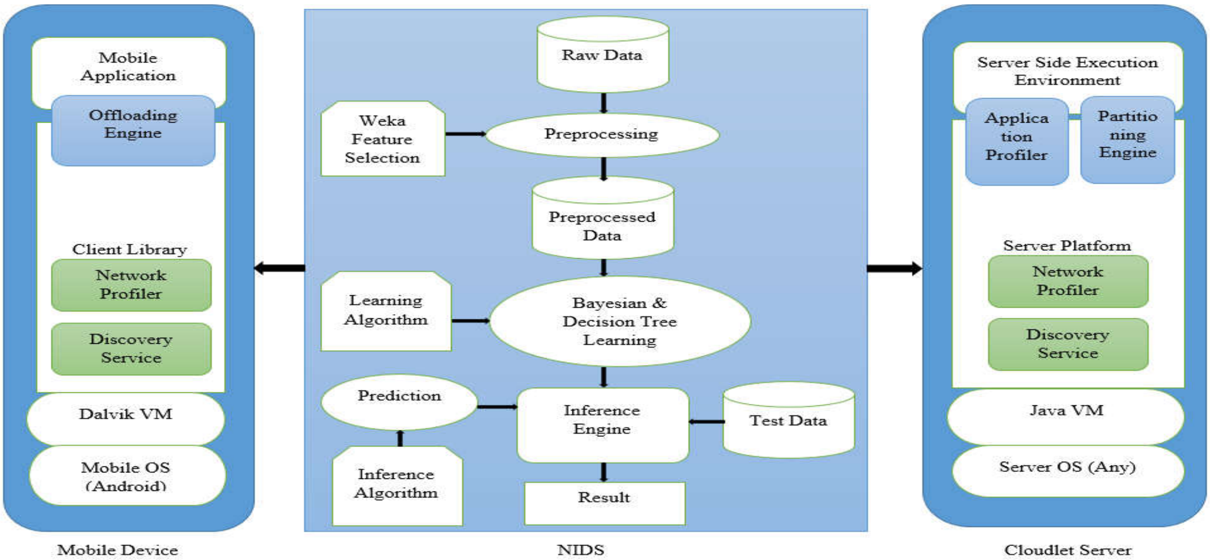

Fig 2. Architecture of the integrated system

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) is measured. The result will be 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.

Offloading Engine

The offloading engine performs decisions whether to offload the execution to the cloudlet or execute it 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 a packet is sent by the source node to the instant the corresponding ACK packet is received by the source node (Wang, 2015). 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 (Costa, 2015), offloading is executed when RTT is lower than 40ms (40 millisecond). However, Tseghai *et al.*, (Tseghai, 2017), observed offloading decisions for different RTT values ranging from 10ms to 100ms. 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.

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

Cloudlet Server Module Components of the System

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 bytecode 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.

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 algorithm 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 did not 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.

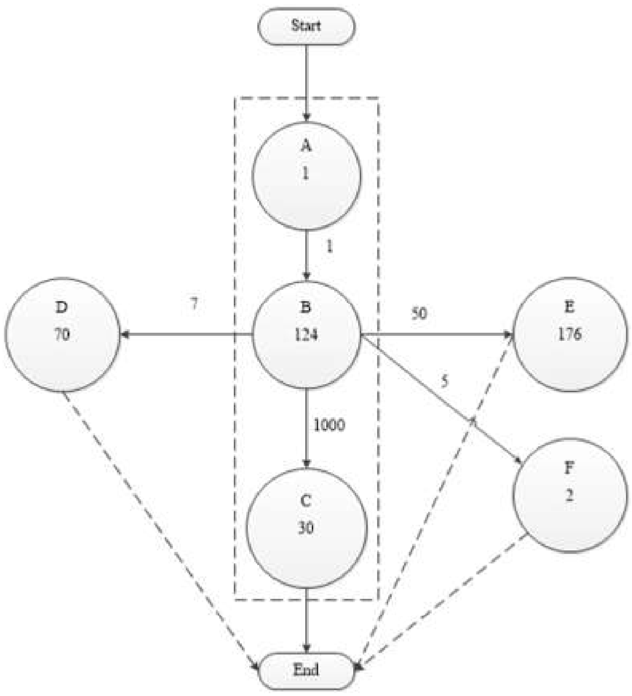

Fig. 3. The first critical path in the call graph

Network based Intrusion Detection System Module Components

Raw Data

The input data for training phase is, the offline dataset which is found on the web for educational research purpose. It is labeled dataset that can be easily learnt by the system. For this case, the research has used simulated dataset called NSL-KDD for this phase. This dataset is selected because it is the latest version of all simulated dataset in the area of network security; redundant records are eliminated from training set and it is affordable to use for experiment purpose as it consists of reasonable number of instances both in the training and testing set (Tigabu, 2012).

Preprocessing

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

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 compare and contrasts every algorithms that are used to develop intrusion detection system and finds the combination of Naïve Bayes and Decision Tree algorithms are best way to develop the system.

```
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
END
Output: Report
```

Pseudo Code 3. Pseudo code for detection stage

Inference Engine

After the Naïve Bayes and Decision tree model are built or trained by the network traffic dataset and ready to predict attacks in the incoming network traffic, the Inference Engine

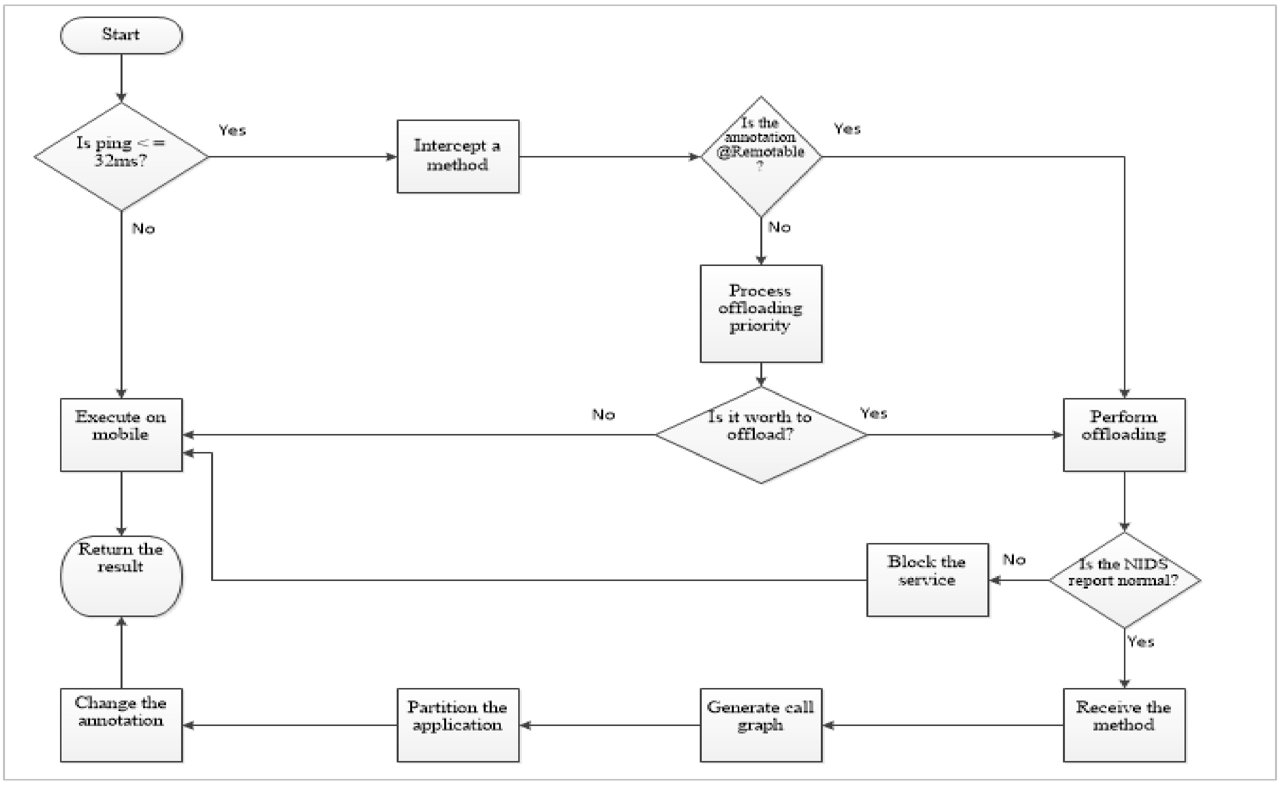

Fig 4: Execution flow of the system

provides the predicting algorithms with test data which is used to compare with the learnt knowledge. The Inference Engine also classify 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 are happened after both algorithms made prediction. Then when both made same prediction, the integrator code takes the result but when they predict different the integrator code takes the worst possibility which is the anomaly.

Communication Model of the System

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

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 considers the compatibility issue with the RPC communication technique and decide the offloading process at method level is the best since RPC supports only method level remote calls. Specifically, the system offloads computations at method level granularity. The system starts its implementation by examining the network quality. If the network quality meets the standard (The ping result < = 32ms), the system intercept a method which is in execution locally and starts the process of offloading. But at the server side, the server receive 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 deny service for the mobile device and make the methods to be executed locally. After appropriate measures are taken by the cloud administrator the services of the cloudlet server will start immediately.

IMPLEMENTATION AND EVALUATION

Implementation of the System

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

Configuration of the System

In order to use the offloading system, the mobile application have 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 (Costa, 2015).
- The IP addresses of the cloudlet must be configured manually and the config 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.

Evaluation of the System

The evaluation of the system is 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 uses NSL-KDD datasets for simulating the detection of intrusion and a computation-intensive mobile application called BenchImage has been used for testing the offloading engine.

RESULTS AND DISCUSSION

Experiments are broadly divided into two parts. The first part consists of the experiments which is done on network based intrusion detection. The second part consists 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.

Experiment on Network based Intrusion Detection System

The performance of the network based intrusion detection system is 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 is confusion matrix values are used. The effectiveness of network based intrusion detection system is measured in terms of accuracy in which it identifies how much do the IDS classify the incoming packet as normal and attack. The accuracy of the system is calculated using the following equation.

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

This research first calculate the accuracy of each algorithm which is Naïve Bayes and Decision Tree. After that the research compares the result with the combined algorithm.

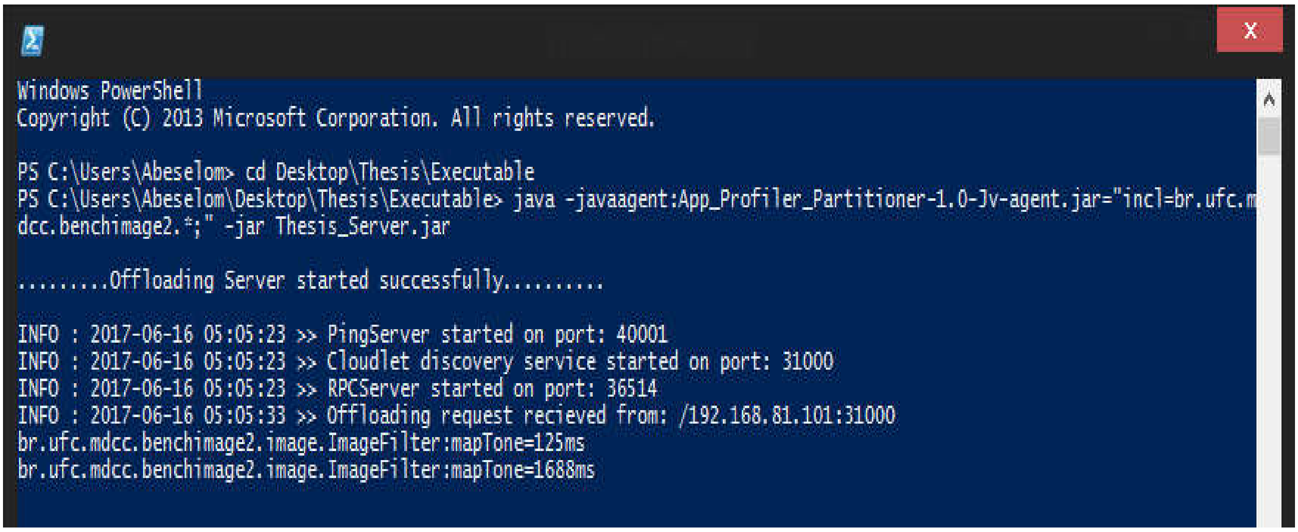

Fig. 5. Handling of service request by the cloudlet server

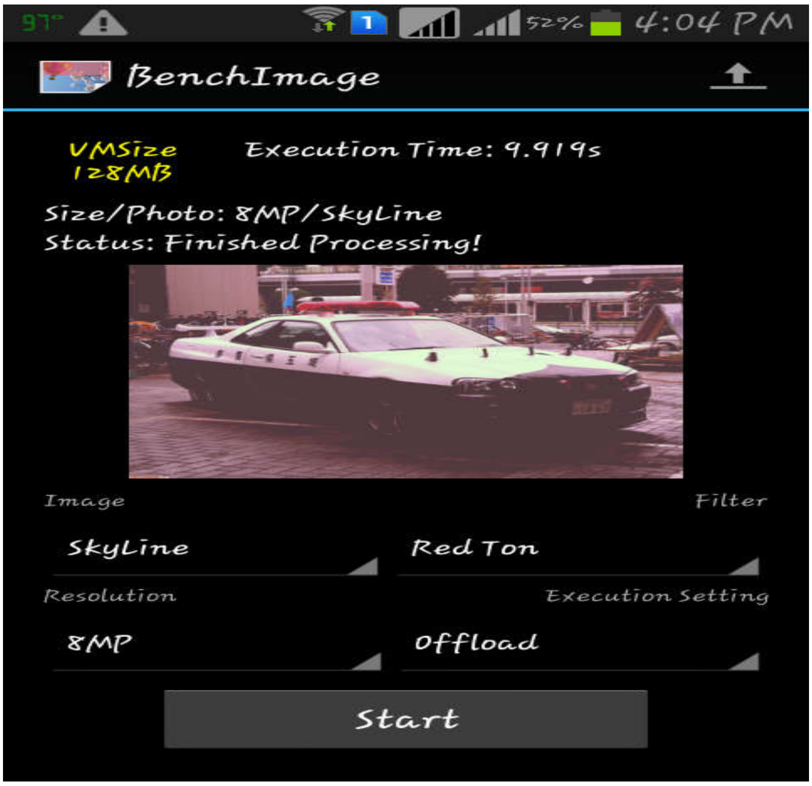

Fig. 6. The result of applying Red Ton filter on the image

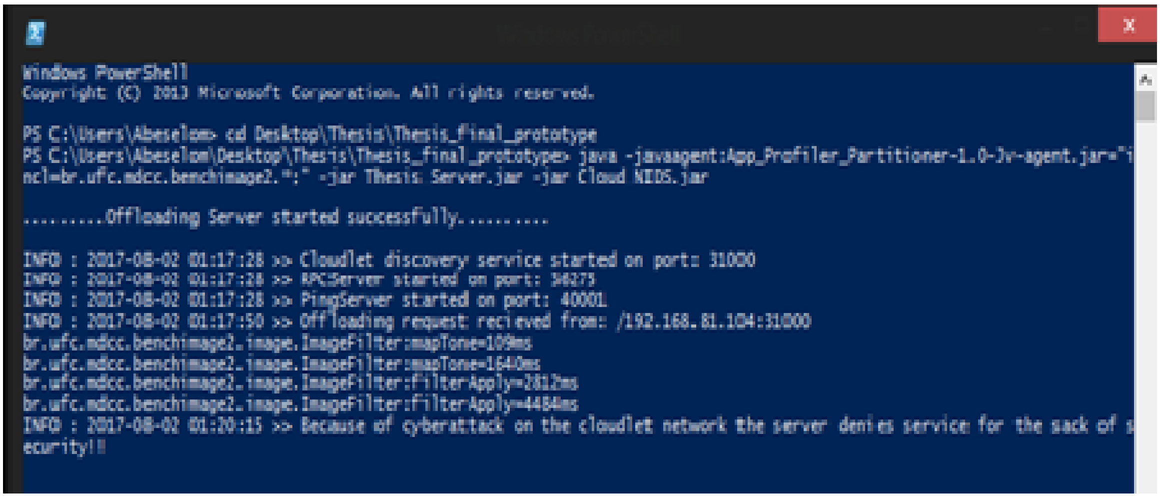

Fig. 7. The integrated system response during attack

For both algorithms the research uses 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 is 76% and for Decision Tree 79%. But in contrast to the results from the Naïve Bayes and Decision Tree algorithms the combined algorithm produces 18543 correctly prediction and 4001 incorrectly prediction. From this result the accuracy of combined algorithm is 82%. Therefore the accuracy of the combined algorithm is better than that of the single algorithm.

Experiment on Integration of NIDS with Mobile

Application Offloading

The main purpose of integrating network based intrusion detection system with mobile application offloading is 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 return back the result to the mobile application. The result of applying Red Tone filter on the image is shown in above Figure 6. When cyberattack happens the integrated system will respond by denying service request by the mobile device and close the stream immediately. When the system deny service, the mobile application immediately process the task locally. The system also shows information about why the request is 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 will start immediately. This mechanism secures the mobile application user from attackers. Figure 7 demonstrate how the integrated system responds during cyber-attack.

Conclusion

Intrusion detection in mobile cloud computing has improved from time to time. However, this 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 is happened by letting them to use the services of cloud during cyberattacks 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 one 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 which uses the attacked cloudlet. This security operation did not disrupt the activities of other clients which are using another cloudlet of the provider. The current intrusion detection systems in the mobile cloud environment lacks these capabilities. Finally, the integrated intrusion detection with mobile computation offloading system is 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

Congr. Serv. Res., pp. 312–316,

Costa, P. B. Rego, P. A. L. Rocha, L. S. and De Souza, J. N. 2015. "MpOS : A Multiplatform Offloading System," ACM, pp. 577–584,

Hevner, A. 2004. "Design Science In Information Systems Research," MIS Q., vol. 28, no. 1, pp. 75–105,

Kumar M. and Hanumanthappa, M. 2015. "Cloud Based Intrusion Detection Architecture for Smartphones," IEEE,

Portokalidis, G. and Bos, H. 2010. "Paranoid Android : Versatile Protection For Smartphones,"

Shabtai, A. Kanonov, Elovici, U. Glezer, C. and Weiss, Y. 2012. "'Andromaly': a behavioral malware detection framework for android devices," *J Intell Inf Syst,* pp. 161–190,

Tigabu, D. 2012. "Constructing Predictive Model for Network Intrusion Detection,"

Tseghai, A. 2017. "A Lightweight Computation Offloading System for Mobile Cloud Computing," Wang, Q. "Restart in Mobile Offloading," Matos, R. Araujo, J. Oliveira, D. Maciel, P. and Trivedi, K. 2014. "Sensitivity analysis of a hierarchical model of mobile cloud computing," Elsevier B.V.,

Yuan, F. Zhai, L. Cao, Y. and Guo, L. 2013. "Research of Intrusion Detection System on Android," IEEE Ninth World

Zhu, H. Huang, C. and Yan, J. 2013. "Vulnerability Evaluation for Securely Offloading Mobile Apps in the Cloud," IEEE, pp. 108–116.

****