
**ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY**

**SCHOOL OF ELECTRICAL ENGINEERING & COMPUTING**

**DEPARTMENT OF COMPUTING**


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

A THESIS SUBMITTED TO
THE DEPARTMENT OF COMPUTING OF ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SYSTEMS


**By:**

**Abeselom Befekadu**



September, 2017

Adama, Ethiopia

# Declaration

This thesis is my original work, and it has not been presented for a degree in any other university. All sources of materials used for the thesis have been acknowledged.

_____

Abeselom Befekadu

This thesis has been submitted for examination with my approval as university advisor

_____

Ravindra Babu (PhD)

**September, 2017**

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING

DEPARTMENT OF COMPUTING

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

By:

Abeselom Befekadu

## Names and Signature of Members of the Examining Board

| | |
|---|---|
| Ravindra Babu (PhD) | _____ |
| Advisor | Signature |
| _____ | _____ |
| External Examiner's name | Signature |
| _____ | _____ |
| Internal Examiner's name | Signature |
| _____ | _____ |
| Chair Person's name | Signature |
| _____ | _____ |
| Department Head's name | Signature |
| _____ | _____ |
| School Dean's name | Signature |

## ACKNOWLEDGMENT

First and foremost, I would like to thank my almighty God for making this possible. My deepest gratitude also goes to Dr. Ravindra Babu, my thesis advisor for providing me with many valuable ideas and insights. He has constantly supported me throughout the thesis work. It was an enormous pleasure for me to work under his guidance. The last but not the least thanks goes to my family and friends for giving me invaluable hope, courage, ideas, and comments.

## ABSTRACT

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 and digital certificates are not enough to secure the data during transmission and the code itself. To alleviate this problem many scholars have been trying to develop cloud based intrusion detection system.

The existing cloud based intrusion detection system works by creating replica of the mobile device on the virtual machine in the cloud. The operation of intrusion detection executed on the replica of the mobile device on the cloud where the intrusion detection system is deployed. After detecting intrusion the users will be notified about the attacks. The main limitation of this security mechanism is the communication way of notifying the user about the attack. During communication between the mobile device and the cloud, the attacker can compromise the channel and make the system to look like normal working time to the mobile device. This allows the attacker to access the data which is transmitted or offloaded to the cloudlet. This mechanism also needs more network bandwidth for coping the network data to the emulation platform on the cloud.

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 proposed 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. Furthermore, the evaluation results of the proposed integrated system showed that it can stop the offloading process immediately during attack and notifies the administrator about the attack.



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

# Table of Contents

# List of Figures

# List of Tables

# List of Pseudo Codes

# List of Abbreviations and Acronyms

| | |
|---|---|
| MCC | Mobile Cloud Computing |
| CC | Cloud Computing |
| NIDS | Network based Intrusion Detection System |
| HIDS | Host based Intrusion Detection System |
| BTS | Base Transceiver Station |
| GUI | Graphical User Interface |
| ADT | Android Development Tools |
| IDE | Integrated Development Environment |
| RPC | Remote Procedure Call |
| RMI | Remote Method Invocation |
| WLAN | Wireless Local Area Network |
| MNO | Mobile Network Operator |
| SaaS | Software as a Service |
| IaaS | Infrastructure as a Service |
| API | Application Programming Interface |
| VM | Virtual Machine |
| DVM | Dalvik Virtual Machine |
| JVM | Java Virtual Machine |
| RTT | Round Trip Time |
| SDK | Software Development Kit |
| MP | Megapixel |

| | |
|---|---|
| ms | Millisecond |
| Javassist | Java Programming Assistant |
| AT&T | American Telephone & Telegraph |
| WEKA | Waikato Environment for Knowledge Analysis |
| ACK | Acknowledgment |
| AES | Advanced Encryption Standard |
| Wi-Fi | Wireless Fidelity |
| IDRS | Intrusion Detection and Response System |
| CPU | Central Processor Unit |
| IT | Information Technology |

# CHAPTER ONE: INTRODUCTION

## 1.1 Overview

Mobile cloud computing is a novel model which practices cloud computing means to eliminate the problem around the curbs of mobile computing[1]. It is one of the focus area because of its speedy advance on mobile application and rise of cloud computing. Nowadays mobile devices are a way of simplified lifestyle. This trend shows how mobile based computation processing is necessary in our day to day life. Despite that mobile devices lacks the capabilities to satisfy the rapid needs for the technology. This limitations occurs because of lower computation power, limited storage and energy efficiency problems of the mobile device. To alleviate this kind of problems mobile application offloading is the key solution[2]. Computation offloading migrates bulky computations and difficult processing from mobile device to the resourceful cloud servers. Mobile application offloading eases the workload of mobile devices. This contributes the efficient usage of battery, memory, storage and CPU.

Although the offloading of computation have many benefits, it also have its own problems. Among them the main problem is the security threat that surrounds during offloading parts of application to the cloudlet server. A significant amount of researches have been done on protecting the mobile cloud computing mainly by using intrusion detection system but none of them consider the security of mobile cloud computing from the perspective of computation offloading. The main focus of some researches is increasing the performance of intrusion detection by deploying the intrusion detection system on cloud server and monitor the activities of the mobile device. The detection process is executed on the replica of the mobile device which is created on virtual machines of the cloud server[3][4][5]. Most of this systems have security vulnerability on their communication channel with mobile device during communicating the result of detection and also needs high network bandwidth to replicate the mobile device on emulating platforms. Others recommend the deployment of the intrusion detection system on the mobile device[6], [7]. However, this way of deployment decreases the performance of the mobile device despite the purposes of mobile cloud computing. Another researcher recommend to use encryption mechanisms to secure the data[8] but security mechanisms that depends on encryption rises computation overhead and only applicable on the data. The redundancy during encryption and decryption process upsets both the

performance and energy efficiency. Current attackers also are smart enough to easily decrypt the encrypted data.

There are two core sources of vulnerability in computation offloading[9]. 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. The main aim of this thesis is to reduce the vulnerability surrounding application offloading by integrating network based intrusion detection system with computation offloading to securely offload mobile applications to the cloudlet.

### 1.2 Motivation

The main motivation that encourages to conduct this research is; mobile application offloading provides a way for the mobile users to accomplish their tasks without worrying about the computation power, battery status and memory storage limitations of mobile devices. However, this significant way of easing work for the mobile users came under threat because of security issues during offloading tasks. To deter this threat and to contribute in this developing area this study propose the research which is conducted under the title integration of intrusion detection system with mobile cloud offloading to securely offload mobile applications to the cloudlet.

### 1.3 Statement of the Problem

Mobile application offloading is well known for its energy efficiency and computation effectiveness even though it has many challenges to be solved around the area of security. In cloud mobile application offloading security threat comes through objects that the cloud uses to pass message from mobile application to the cloud server[9]. 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 more popular and effective means of security. To achieve the goals of securing the mobile cloud computing the existing intrusion detection systems performs the detection process either on the mobile device[6], [7] or on the replica of the mobile device on virtual machines in the cloud[3][4][5]. These security mechanisms introduce an extra computation overhead to the resource constrained mobile devices, which affects the performance gain from computation offloading and have their own security vulnerability during communicating the detection result with the mobile device. Moreover, the

intrusion detection system which is deployed on the cloud server needs more network bandwidth to create the replica of the mobile device on emulating platforms.

To overcome those problems the proposed system views the security of mobile cloud computing from the perspective of computation offloading in contrast with other system. These allows the proposed system to alleviate the limitations around the communication, the bandwidth and the performance of existing systems. More specifically the following questions has been addressed in this research.

1. How to detect network based cyber-attacks?
2. How to alert the cloudlet administrator if there is intrusion?
3. How to prevent mobile application methods from passing to the cloudlet server if there is an attack? Or how to deny service request of mobile device when intrusion happens?

## 1.4 Objectives

### 1.4.1 General Objective

The main objective of this research is to develop and integrate network based intrusion detection system with computation offloading to securely offload mobile applications to the cloudlet.

### 1.4.2 Specific Objectives

The specific objectives of this research are:

- To conduct a detailed literature review to understand the application domain for machine learning in network based intrusion detection and the computation offloading concepts.
- To select the best detection algorithm that can examine unknown attacks.
- To select offloading algorithm that can increase the performance of the mobile device.
- To develop a lighter computation offloading engine for mobile cloud computing using the selected offloading algorithm.
- To develop network based intrusion detection system using the selected detection algorithm and integrate it with computation offloading engine.
- To test and evaluate the proposed integrated network based intrusion detection system with mobile computation offloading engine.

## 1.5 Scope and Limitations of the Study

The proposed 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 proposed 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 proposed system uses the previous execution time and call relationship of the methods.
- A new offloading and detection algorithm is not designed.
- 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 did not implement the integrated system in public cloud environment.
- The research only considered attacks which is found on NSL-KDD datasets.

## 1.6 Methodology

To carry out the overall works of this thesis, different methods and techniques was employed. The detailed description of these methods is presented as follows:

### 1.6.1 Research Design

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[10]. The five steps of design oriented method followed in this thesis work are problem identification, solution suggestion, development of the proposed system, evaluation and conclusion.

### I. 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 cloud 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.

### II. 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 proposed 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 Proposed 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 proposed 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.

**IV. Evaluation and Improvement**

Finally, the proposed 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.

**V. Discussion and Conclusion**

The results which are found during evaluation phase and the achievements of the proposed 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.

### 1.6.2 Tools and Techniques

To write the documentation Microsoft Office is used. In order to draw the execution flow, call graph and critical path selection Microsoft Visio was used. For development of the system Eclipse neon development environment is utilized. To develop the android application android development tool (ADT) plugin was installed and used on the Eclipse neon. Java programming language is also used to develop the system. Waikato Environment for Knowledge Analysis (WEKA) is utilized for the purpose of data mining and machine learning during developing the network based intrusion detection system.

## 1.7 Significance of the Study

This thesis focuses on integrating network based intrusion detection system with computation offloading to securely offload mobile application to the cloudlet. This helps to reduce problems associated to network attacks caused by intruders in different mobile users. The result which is

gained from this research can be applied or used to secure the mobile cloud computing specifically cloud application offloading. Any cloud service provider can use the developed integrated system to secure cloudlet and clients' resource from being attacked by cyber criminals. The main beneficiaries of this research are all mobile application users which have low CPU, limited storage and short battery life time to install and use security application on their mobile devices as the study solves the problems around the security risks during offloading computation to the cloudlet server which is more resourceful. Moreover, it would contribute to grant the confidentiality, integrity and availability of cloud resources and it simplifies cloudlet management.

### 1.8 Organization of the Thesis

This thesis work is organized in six chapters. The first chapter presents the introduction part of the thesis. The rest of this thesis is organized as follows, Chapter Two covers literature review. Chapter Three presents the existing works that are related to the objectives of this thesis. The design of proposed integration of network based intrusion detection system with mobile computation offloading is discussed in Chapter Four followed by implementation and evaluation in Chapter Five. Finally, the conclusion made on the thesis result and future work of this thesis are presented in Chapter Six.

## CHAPTER TWO: LITERATURE REVIEW

### 2.1 Mobile Cloud Computing (MCC)

Before MCC, there is a need top to bottom comprehension of mobile computing and cloud computing (CC).Mobile computing can encourage the clients by giving a tool regularly regardless of their moment. Conversely, with mobile computing and portability, smart device confronts its characteristic issue including limited battery, resource inadequacy and low connectivity[11]. In latest research, this issue has been resolved by CC. CC can be characterized as a paradigm that conveys computing and IT as a service. The cloud resource on-demand idea has pulled in smartphone clients to use different CC services, for example, "Software, Platform, and Infrastructure" as-a- service ("SaaS, PaaS, and IaaS") at low expenses[12], [13]. The idea driving MCC is to permit resource-constrained portable smart device to utilize data storage and data processing of intense and brought together computational cloud to address the innate issue of mobile computing[14]. The MCC framework must guarantee portability support and service accessibility so clients can have full access and connection with the cloud without disturbance[15]. Additionally, MCC offers better comfort to smartphone clients by reducing their stresses over equipment and programming complexities. The pay-as-you-use idea has attracted smartphone clients to profit from low cost and on-demand cloud benefits that encourage the reliable, scalable, aggregation of computing, sharing, and assignment of storage and systems networking assets [16].

The cloud resources overwhelmed the issue of resource shortage for smart devices[16] and to be a promising answer for mobile computing because of various reasons (portability, communication, scalability, and mobility). Smart device clients can get the cloud resources simply by connecting to the internet through wireless technology and give the ability to send data which needs high computation to the cloud[16]. Anyhow, in spite of every one of these features, MCC is defenseless against attacks as a result of its distributed infrastructure and the way that it can be easily accessed by anybody. Defensive components, for example, firewall, cryptography, and access control can just give constrained security to MCC. Accordingly, a cloud-based IDS is expected to secure MCC and smart device.


*Figure 2. 1 General idea of mobile cloud computing[17]*

### 2.2 Mobile Cloud Computing (MCC) Architecture

The general architecture of mobile cloud computing is delineated in figure 2.2. Components of mobile cloud computing architecture and their purposes are also discussed after figure 2.2.

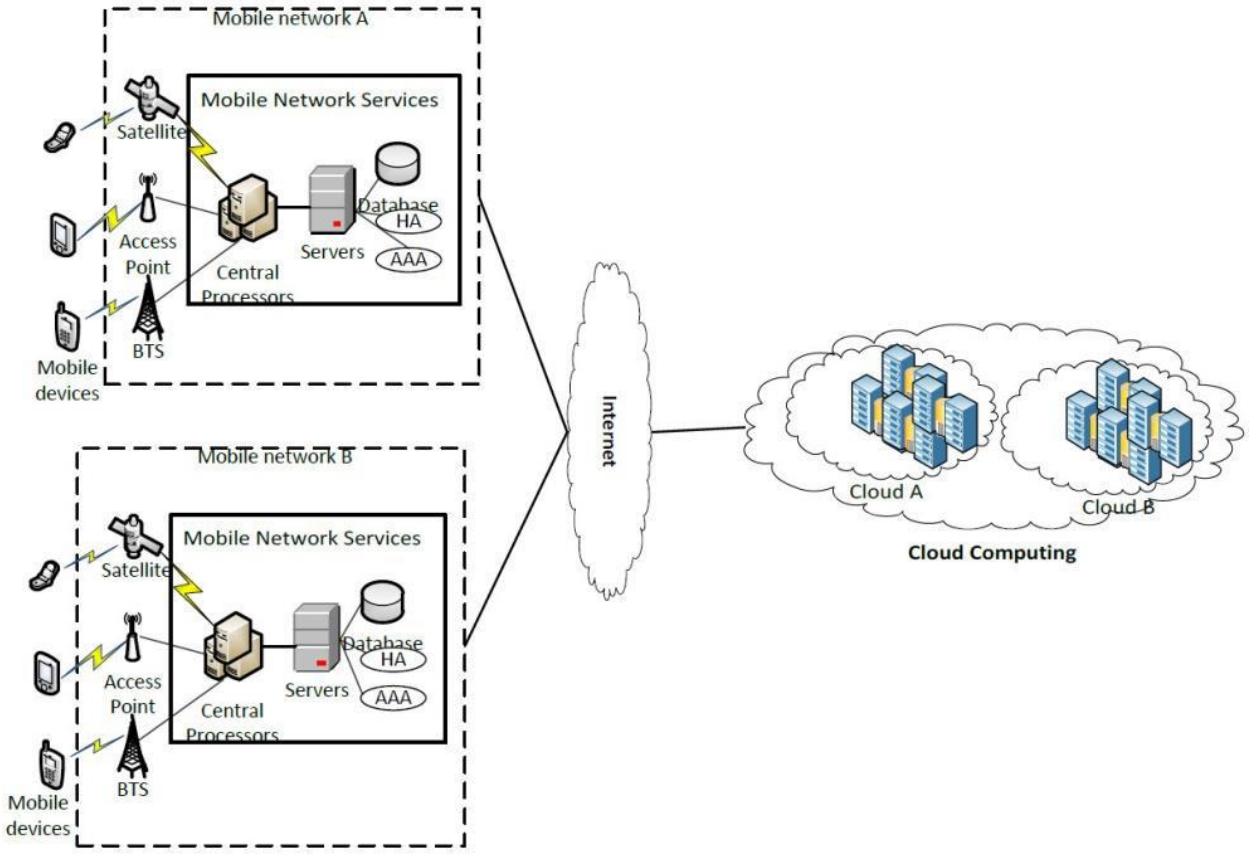


*Figure 2. 2 Architecture of MCC[18]*

### A. Mobile Devices

Mobile Devices are linked with the network through base stations that setup and control the connotations (air interface) and practical interfaces between the network and device. These base stations can be Base Transceiver Station (BTS), access Point or Satellite[19].

### B. Network Operator

It has Base Station through which cell phones are linked. These Base Stations are connected with the central processors which prepare the mobile client's demand. Mobile clients get AAA (Authentication, Authorization and Accounting) service from the mobile network administrator. Mobile clients' requests and information are transmitted to the cloud controllers that are connected

with servers giving mobile network services. The subscribers' request are sent to a cloud through the Internet[16].

### C. Cloud Controllers

The cloud controller gets the subscribers' requests from the network administrator by means of the Internet and process the request to give a corresponding cloud services to the mobile clients. It utilizes the resources in the datacenter to give diverse services for mobile clients[19]. These services are produced and given the thoughts of utility computing, virtualization, and service-oriented architecture (e.g., web, application, and database servers)[16].

## 2.3 Mobile Computation Offloading

Mobile devices, for example, smart mobile phones and tablets have turned into the most demanding computing devices because of their ubiquity. These devices give a wide collection of helpful services (e.g., web surfing, video calls, messages, gaming, navigation and guidance, healthcare, banking, education and so forth.) from anyplace at any time. However, smartphones are confined by their processor performance, data transfer capacity, storage limit and battery lifetime[20]. These imperatives impede smartphones from performing complex jobs and running computation-intensive mobile applications. A noticeable answer to overthrow these confinements is computation offloading, which is the concentration of mobile cloud computing (MCC)[21].

Computational offloading indicates to a system, in which a computational-intensive application parts are caught from a neighborhood execution workflow and relocated to a remote server in the cloud environment for execution, and the result of the processing is synchronized once more into the nearby workflow[22]. Computation Offloading can be performed at various levels of granularity. It can be performed at the entire application level, class level, method level, thread level, object level and so forth. Additionally, computation offloading is not quite the same as the traditional client-server architecture where a thin customer regularly moves computation to a remote server[23]. Though in computation offloading, a computation-intensive piece of an application is moved to a remote server just when offloading is advantageous.

## 2.4 Computation Offloading Architectures of MCC

Computation offloading architecture illustrates the kind of resources used to streamline the capacity of smartphones and the route in which mobile devices are linked with and connect with

cloud assets. In light of the cloud resource organization, existing computation offloading architectures can be arranged into the subsequent four classes[17]:

### A. Public cloud

The entire or segments of the mobile application is offloaded to public computing resources comprised of stationary servers through the Internet which is located in different geographical location[17]. Some offloading structures[24], [25] Utilize this architecture to optimize the execution of the applications and to decrease the energy utilization of the mobile device. The uniqueness in computing capacities between the bunch of server's public cloud and smart mobile phones is high and persistent. Public cloud is exceedingly accessible, scalable, and resource elastic with intense processing abilities while smartphones have limited resources[17]. Along these lines, utilizing public cloud to execute smartphone tasks can exceedingly upgrade the execution performance of the mobile application. In any case, the broadcast time and communication cost must be considered for offloading on the grounds that access to the public cloud may incur higher delay due to the lower quality of the wireless network.

### B. Cloudlet

A cloudlet is a trustworthy, resource rich Computer or assembly of Computers which is all around linked with the Internet and accessible for use by close-by smart devices. Consequently, when smartphones would prefer not to offload to the cloud (possibly because of deferral and cost), they can discover and utilize a close-by cloudlet. Along these lines, mobile clients may take care of the demand for real-time interactive response by low-latency, one-hop, and high-data transmission remote access to the cloudlet. In the event that no cloudlet is accessible close-by, the smartphone may allude to the default mode that will send preconditions to a far off cloud, or in the most pessimistic scenario, exclusively its own particular resource[26].

### C. Cloud of Mobile Network Operator (MNO)

A mobile network operator (MNO) is a telecommunications service provider organization that provides wireless voice and data communication for its subscribed mobile users. It possesses or controls the majority of the things important to offer and convey services to an end client, for example, radio spectrum allocation, wireless network infrastructures, and provisioning of

computer systems and marketing. By executing the cloud ideas including SaaS, IaaS, MNO can be a cloud vendor. Moreover, MNO can be a cloud representative which plays as a service intermediary between other cloud vendor and mobile clients. There are a few MNOs that give cloud service, for example, the China Mobile and AT&T. Cloud of MNOs spreads a vast region and its more protected and reliable[17].

### D. Cloud of Mobile Devices

In the other three structures, the powerful Computers in the cloud are the computing resource suppliers to the resource compelled mobile device. However, there may be a circumstance in which services by those resources is not accessible and additionally it is excessively costly, making it impossible to get to them. A virtual cloud made from the gathering of smartphones in vicinity and in stable form can be an answer for such circumstances[17]. The virtual cloud gives computing resources and process to the offloaded mobile applications. Hyrax[15] introduced an idea of utilizing the collection of mobile devices as computing resource vendor. With cloud mobile device, clients can share resources and execute tasks among the devices to increase better involvement and execution[17]. But, the variation of processing power may occur because of the portability of the mobile device, which influences the execution of the computation offloading system.

## 2.5 Security Issues in Cloud Computing

Because of the increasing number of attacks on cloud infrastructure, security in cloud computing environment became one of the focus area. Security dangers in cloud can be classified as follows [27]:

### 1. Cloud information secrecy issue

Secrecy of information over cloud is one of the obvious security concerns. Encryption of information can be possible with the ordinary techniques. But, encrypted information can be secured from a malicious client however the protection of information even from the administrator of information at service provider’s end couldn't be covered up. Searching and indexing on encrypted data remains a state of worry in that case. Previously mentioned cloud security issues are a few and dynamicity of cloud architecture are confronting new difficulties with fast implementation of new service paradigm.

2. **Network and host based attacks on remote Server**

On remote hypervisors host, network intrusion attacks are a noteworthy security worry, as cloud vendors' usage of virtual machine technology in their cloud infrastructure. Denial of service and distributed denial of service attacks are attacks that makes the service to refuse accessibility to end clients.

3. **Cloud security auditing**

Cloud auditing is a difficult task to check the consistency of all the security arrangements by the provider. Cloud service provider has the control of sensitive client information and processes, so an automated or outsider auditing mechanism for data integrity check and measurable investigation is required. Protection of information from outsider reviewer is another worry of cloud security.

4. **Lack of data interoperability standards**

It comes about into cloud client information secure state. In the event that a cloud client needs to move to other service provider because of some specific reasons it would not have the capacity to do as such, as cloud client's data and application may not be perfect with other provider's data storage configuration or platform. Security and secrecy of data would be in the hands of cloud service provider and cloud client would be reliant on a solo service provider.

### 2.6 Intrusion Detection System

An intrusion detection system (IDS) screens network traffic and monitors for suspicious action and alarms the system or system manager. In some situations, the IDS may likewise react to strange or malicious activities by making a move, for example, obstructing the client or source IP address from getting to the system. IDS comes in range of "flavors" and approach the objective of distinguishing suspicious activity in various ways.

There are network based and host based intrusion detection systems. There are also IDS that identify in light of searching for particular signatures of known threats like the way antivirus software ordinarily detects and safeguards against malware (Signature based IDS) and there are IDS that identify based on comparing traffic patterns against a baseline and searching for irregularities (Anomaly based IDS)[28].

## I. NIDS

Network Intrusion Detection Systems are set at a vital point or deployed inside the system to screen network traffic to and from all mobile device on the network. In a perfect world, you can filter all inbound and outbound network traffic, though doing as such may make a bottleneck that would impede the general speed of the network.

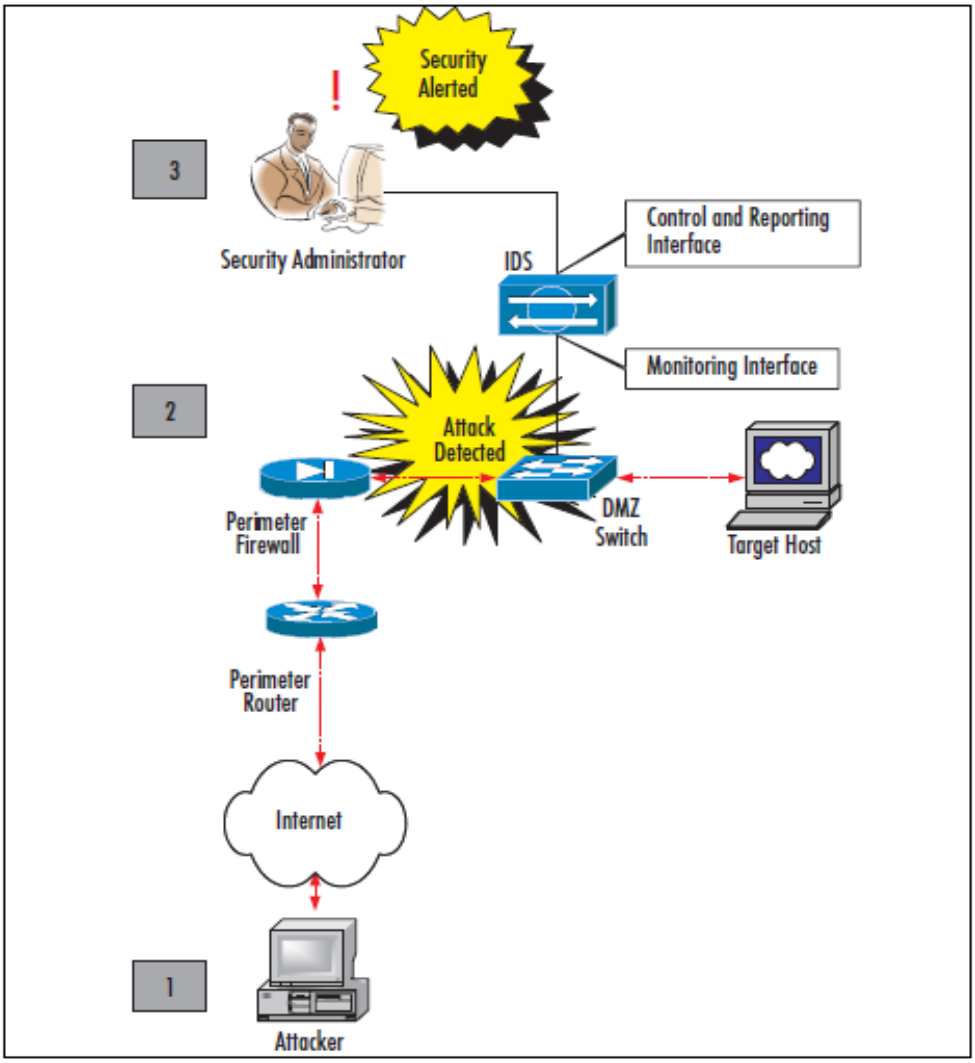


*Figure 2. 3 A typical NIDS deployment*[29]

## II. HIDS

Host based Intrusion Detection System are keep running on individual hosts or devices on the network. An HIDS screens the inbound and outbound bundles from the mobile device simply and will warns the client or administrator of suspicious action is noticed.

### A. Signature Based

A signature based IDS will screen packets with respect to the network and look at them against a database of signature or attributes from known malicious risks.

This is like the way most antivirus software identifies malware. The issue is that there will be a delay between another threat being found in the wild and the signature for identifying that risk being applied to your IDS. For the period of delay time your IDS would not be able to identify the new threat.

**Advantages**

- Signature detectors are very effective in detecting attacks without generating a large number of false alarms.
- They can quickly and accurately diagnose the use of a specific attack technique. This can help those responsible for security to easily follow security problems and to prioritize corrective actions.

**Disadvantages**

- Signature detectors only detect the attacks they previously know, so they must be constantly updated with signatures of new attacks.
- Many signature detectors are designed to use very tight patterns that prevent them from detecting variants of common attacks.

### B. Anomaly Based

An IDS which is Anomaly based will screen network traffic and look at it against well-known baseline. The baseline will identify what is "normal" for that network what kind of bandwidth is for the most part utilized, what protocols are employed, what ports and devices generally connect with each other and aware the administrator or client when traffic is identified which is abnormal, or fundamentally extraordinary, than the baseline.

**Advantages**

- The IDSs based on anomaly recognition detect unusual behavior. Thus, they have the ability to detect attacks for which they have no specific knowledge.
- Anomaly detectors produce information that is very useful to define new patterns for signature detection.

**Disadvantages**

- The detection of anomalies produces a high number of false alarms due to the unpredictable behavior of users and networks.
- They require very hard training to characterize patterns of normal behavior.

### 2.7 Functionalities of IDS[30]

1. Observing the network and examining both client and system activities.
2. Conduct analysis on the system and identify vulnerabilities.
3. Surveying system and data integrity.
4. Capacity to distinguish patterns typical of threat.
5. Inspection of irregular activity patterns.
6. Following client policy breach.

### 2.8 Intrusion Detection System in Cloud Computing

The Intrusion Detection Service (IDS) benefit expands a Cloud's security level by giving two strategies for intrusion detection. Detection of signatures and detection of threats however some machine learning methodologies are incorporated here. The signature detection is the technique utilized by most business systems. The anomalies detection, in which the analysis searches for strange pattern of action, has been and stays under inspection. The discovery of threats is utilized by few IDSs[29], [31].

Numerous great causes prompt the exploitation of cloud based IDS. Since smart devices behave suchlike to desktop personal computer by undergoing corresponding functions and capacities in terms of utilization, the threads presented by a desktop personal computer are practical to smart devices too. Presently, smart devices reflect conventional protective mechanisms of desktop personal computer in which detection and security countermeasures are installed on the devices. However, in a smartphone, high computational security IDRS-based algorithms are hard to apply

and accomplish due to constricted storage and processing power. Consequently, the IDS must be switched to cloud resource, which has high computational power and storage capacity. The cloud-based IDS play's a robust purpose in effectively finding malevolent actions happing inside the smart device. Smart device and then employs the cloud-based IDRS is equal to a cloud application for securing smart device[32].

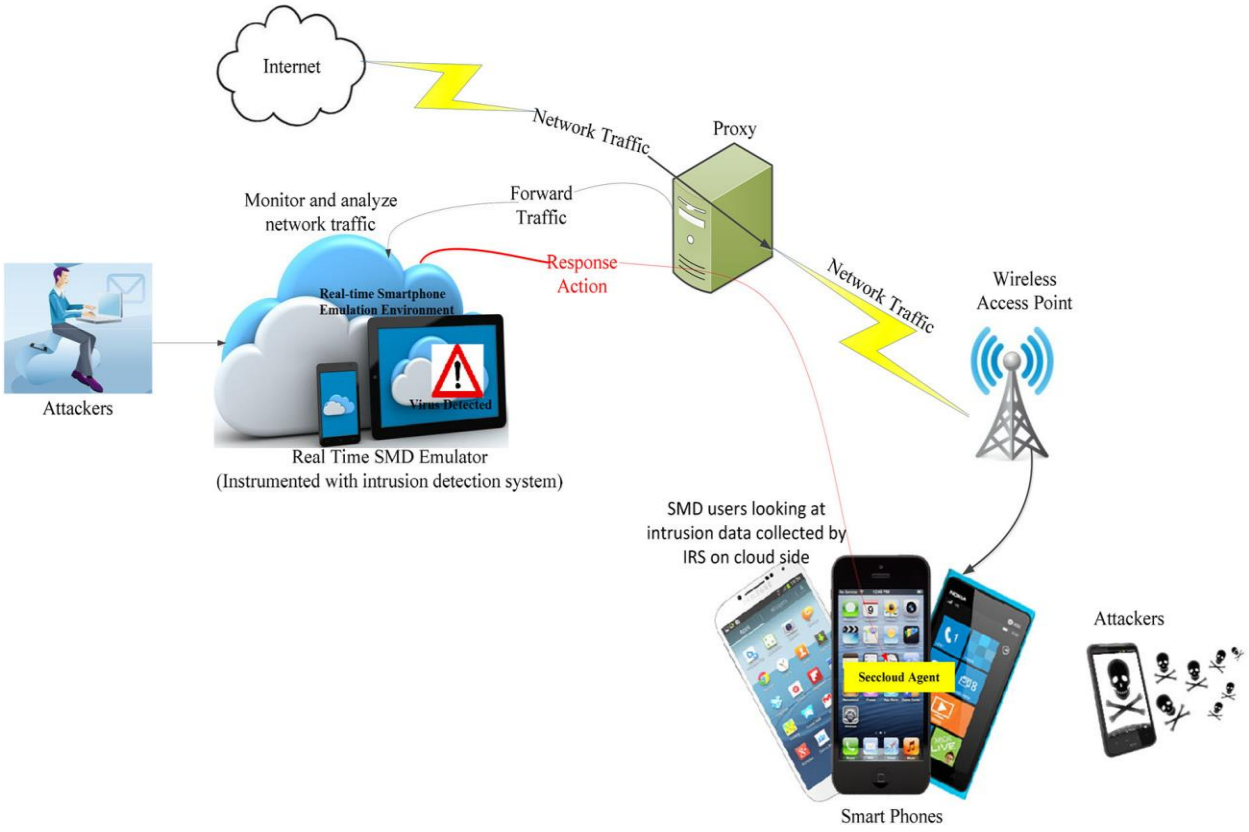


*Figure 2. 4 Overview of cloud-based IDS*[33]

## 2.9 Summary

The rapid progress of mobile computing has transformed mobile applications into more complicated and computation intensive. However, mobile devices have limited resources (e.g., battery life, storage, and bandwidth etc.) and those limitations handicap the functionality of the mobile applications. As a solution to this problem, mobile cloud computing was introduced which is the integration of the resourceful cloud computing environment and the mobile computing technology. This brought new types of services and facilities for mobile users. The service includes the migration of computation intensive mobile application parts into the cloud environment for execution, which is known as computation offloading. Computation offloading is a promising technique to overcome the limitations of mobile devices. In spite of every one of these features, mobile cloud computing is susceptible to wide range of attacks because of its distributed infrastructure and its accessibility by everyone. Security measures like firewall, cryptography, and access control have limitations to protect mobile cloud computing. To alleviate the limitations surrounding computation offloading, integration of network based intrusion detection system with computation offloading is a vital means of protection against network based attacks.

## CHAPTER THREE: RELATED WORK

### 3.1 Overview

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. The most prominent studies that are more related with the objectives of this thesis are discussed as follows.

### 3.2 Related Works by Using Intrusion Detection System in Cloud

Portokalidis et al[5] 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.

A fundamentally the same approach as Portokalidis et al. Response System for Mobile Phones exists in existing Android architecture[4]. In this architecture, a cloud based intrusion detection and response engine which creates image of the device on a remote cloud server to give intrusion detection through cloud computing where a lightweight mobile host agent was utilized to recover the device and acquire the information or files from the device and send it to the system. There are a wide range of intrusion detection procedures that are utilized as a part of this system. However, when an attack detected by the intrusion detection system the intrusion engine at the emulator environment sends the selected optimal actions to the agent which is running on the phone but smart attackers can manipulate this communication and make the system to act like normal time.

Manish Kumar and Dr. Hanumanthappa[3] 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[7] 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, Yael Weiss[34] 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.

In 2012 Huaibin Wang, Haiyun Zhou and Chundog Wang[35] proposes a virtual machine based intrusion detection system framework in cloud computing environment. The researchers isolate the properties of virtualization and then the virtual machine based multiple intrusion detection systems are put-on on each layer of virtual machine to monitor each virtual component. They also propose a mechanism to repel denial of service attack (DOS) and distributed denial of service attack (DDOS) by using cloud alliance concept on communicating agents. Isolating virtualization concept leads to the reduction of performance for the cloud servers since virtualization is the big concept in mobile cloud computing.

Numerous endeavors have been taken in the area of Cloud computing and intrusion detection system, yet at the same time there are more attacks that have not been identified. In[36], the specialists worked in this field to beat the present security dangers in the cloud computing through executing IDS in cloud environment which is responsible for observing the use of resources for the virtual machine utilizing information obtained from virtual machine monitors. More particularly, all inspection operations are done outside the virtual machines so the attacker can't alter the system on account of occupant's instance is breached. Although, there are many sorts of intrusions that this strategy can't distinguish, for example, getting to the account of permitted clients with no consent. Also, if strange movement happens, it will be distinguished as intrusion regardless of the possibility that it is approved action. In this way, every one of these holes have to be considered amid implementing IDS inside cloud environment.

In[37], the writers proposed an effective model that utilized multithreading procedure for enhancing IDS execution inside cloud computing environment to deal with expansive number of information packet streams. The proposed multi-threaded NIDS depends on three modules named: catch module, analysis module and reporting module. The first is capable of catching data packet and sending them to analysis part which investigates them effectively through matching against pre-defined set of rules and identify the terrible bundles to produce alarms. At last, the reporting module can read alarms and quickly get ready the report. The writers conducted experiment to demonstrate the viability of their proposed technique and contrasted it with single threaded which displayed elite as far as processing and execution time. Nonetheless, the issue of detecting new sorts of attacks still needs many effort to be done.

Miettinen[38] introduced a hybrid network based and host based IDS, on account of the claim that by itself, these system are not adequate. The engine on the server filters the acquired alerts as indicated by correlation rules deposited in the knowledge base and sends the outcomes to the security monitoring GUI for the purpose of analysis by the administrator. During deploying host based IDS privacy is the main challenge. When the host based IDS trigger intrusion alarm the IDS module sends the report to backend intrusion detection server. Doing this, the communication channel may be compromised by the attacker and the report may be hijacked. This makes the system vulnerable to attacks.

### 3.3 Related Works by Using Other Security Mechanisms in Cloud

In 2014 Bhavya Sareen, Sugandha Sharma and Mayank Arora[8] proposes an architecture which protects text and image data by encryption and decryption process. The study uses Advanced Encryption Standard (AES) algorithm to encrypt the data in the mobile device before offloading and decrypt by the server to conduct computation on the data. After computation is over at the server side the data will be encrypted again by the server and send back to the mobile device. The encryption and decryption process needs more computation on the mobile device. This usage of resource decrease the performance of the mobile device against the benefits of computation offloading.

Another approach, particularly focused on the Symbian platform, is depicted by Bose et al[6]. They are depending on behavioral investigation for distinguishing malware on smartphones. Their thought depends on the supposition that a single activity performed by an application can be delegated as safe, yet in connection to different activities, which are performed in a similar setting, malware conduct can be uncovered. In view of this presumption Bose et al. built up a database of behavioral signatures for malware. Via training a support vector machine with ordinary conduct of applications, anomalies such as malware can be identified.

Patrick P. Tsang, Apu Kapadia[39] proposed an approach called Nymble .Users are permitted to get to Internet secretly by utilizing anonymizing network, for example, Tor: through a series of routers masking the customer's IP address from the server. However, a few clients mishandle of such system secrecy for destroying sites. Site administrators uses IP-address blocking to limit such acting clients. Conversely IP-address blocking is worthless when clients can utilize anonymizing network. Nymble is a system that can blacklist bad clients; in this manner blocking clients without

their anonymity. Hiding the client's IP address from the server makes the computation offloading process impossible because the server and the device needs each other's IP address to identify each other for communication. This makes the solution incompatible with application offloading.

### 3.4 Summary

The following table summarizes the previous work on network intrusion detection systems in mobile cloud computing environment and other security mechanisms and then compares these systems with the proposed system.

*Table 3. 1 Summary of related works*

| No | Papers | Year | Contribution | Merits | Demerits |
|---|---|---|---|---|---|
| 1 | Portokalidis et al.[5] | 2010 | Paranoid Android: Versatile Protection for Smartphones | Protect smart devices by creating the exact replica of the device on virtual environment where the security checks are implemented. | The attacks can only be detected after synchronization with the server. |
| 2 | Sareen et al.[8] | 2014 | Securing Smartphone Data by Offloading Computation on Cloud | Protects text and image data by encryption and decryption process. | Decrease the performance of the mobile device against the benefits of computation offloading. |
| 3 | Houmansadr et al.[4] | 2012 | A Cloud-based Intrusion Detection and Response System for Mobile Phones | Provides IDR engine which create image of the smart device on a remote | Attackers can stop the communication between the server and the |

| | | | | | |
|---|---|---|---|---|---|
| | | | | cloud server for the detection of intrusion. | device and make the system to act like normal time. |
| 4 | Kumar et al.[3] | 2015 | Cloud Based Intrusion Detection Architecture for Smartphones | Clients install a lightweight agent on their smartphone and the smartphone is copied in a virtual machine where intrusion detection is executed. | Did not specify how the intrusion detection engine communicates with the client host agent. |
| 5 | Yuan et al.[7] | 2013 | Research of Intrusion Detection System on Android | It extracts different features acquired from the Android framework, for example, the network traffic of smartphones, battery utilization, CPU use, the measure of running processes and decide whether there is intrusion or not. | The performance of the smartphone decreased since it deployed on the device. |

| | | | | | |
|---|---|---|---|---|---|
| 6 | Shabtai et al.[34] | 2011 | "Andromaly": a behavioral malware detection framework for android devices | 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. | Did not demonstrate the proposed solution on the data which is obtained from the real world. |
| 7 | Bose et al.[6] | 2008 | Behavioral Detection of Malware on Mobile Handsets | Via training a support vector machine with ordinary behavior of applications, anomalies such as malware can be identified. | Performance of the device reduced because of the whole work is done on the device. |
| 8 | Wang et al.[35] | 2012 | Virtual Machine-based Intrusion Detection System Framework in | By isolating the properties of virtualization the virtual machine based multiple | Isolating virtualization concept leads to the reduction of |

| | | | | | |
|---|---|---|---|---|---|
| | | | Cloud Computing Environment | intrusion detection systems are put-on on each layer of virtual machine to monitor each virtual component. | performance for the cloud servers. |
| 9 | Nikolai et al.[36] | 2010 | Detecting Unauthorized Usage in a Cloud using Tenant | All inspection operations are done outside the virtual machines so the attacker can't alter the system on account of occupant's instance is breached. | Accessing the account of authorized users without any permission cannot be detected. |
| 10 | Gul et al.[37] | 2011 | Distributed Cloud Intrusion Detection Model | Demonstrate the viability of the proposed technique and compared it with single threaded which displayed elite as far as processing and execution time. | Detecting new sorts of attacks still needs many effort to be done. |

| | | | | | |
|---|---|---|---|---|---|
| 11 | Miettinen et al.[38] | 2006 | Host-based Intrusion Detection for Advanced Mobile Devices | The engine on the server filters the acquired alerts as indicated by correlation rules deposited in the knowledge base and sends the outcomes to the security monitoring GUI for the purpose of analysis by the administrator. | The communication channel which is used by the device and the server can be compromised by the attacker and the report can be also hijacked. |
| 12 | Tsang et al.[39] | 2011 | Nymble: Blocking Misbehaving Users in Anonymizing Networks | Blacklists bad users which are permitted to get to Internet secretly by utilizing anonymizing networks, for example, Tor. | Hiding the client's IP address from the server makes the computation offloading process impossible because the server and the device needs each other's IP address to identify each other for communication. |

| | | | | | |
|---|---|---|---|---|---|
| 13 | Proposed System | 2017 | Integration of Network based Intrusion Detection System with Mobile Cloud Offloading to Securely Offload Mobile Applications to the Cloudlet | -Protect the computation offloading process.<br>-Stop the attackers from capturing the client's data at the time of attacks.<br>-Improve performance.<br>-Reduce state transfer between the mobile devices and the cloud.<br>-Fully support computation offloading to cloudlet servers. | The offloading algorithm is application specific. |

# CHAPTER FOUR: DESIGN OF THE PROPOSED INTEGRATION OF NIDS WITH MOBILE COMPUTATION OFFLOADING

## 4.1 Overview

The existing intrusion detection systems in mobile cloud computing environment are sophisticated, vulnerable and can bring an additional overhead to the resource constrained mobile devices. To alleviate this problem, integration of network intrusion detection system with mobile cloud offloading has been proposed in this thesis. 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 proposed system.

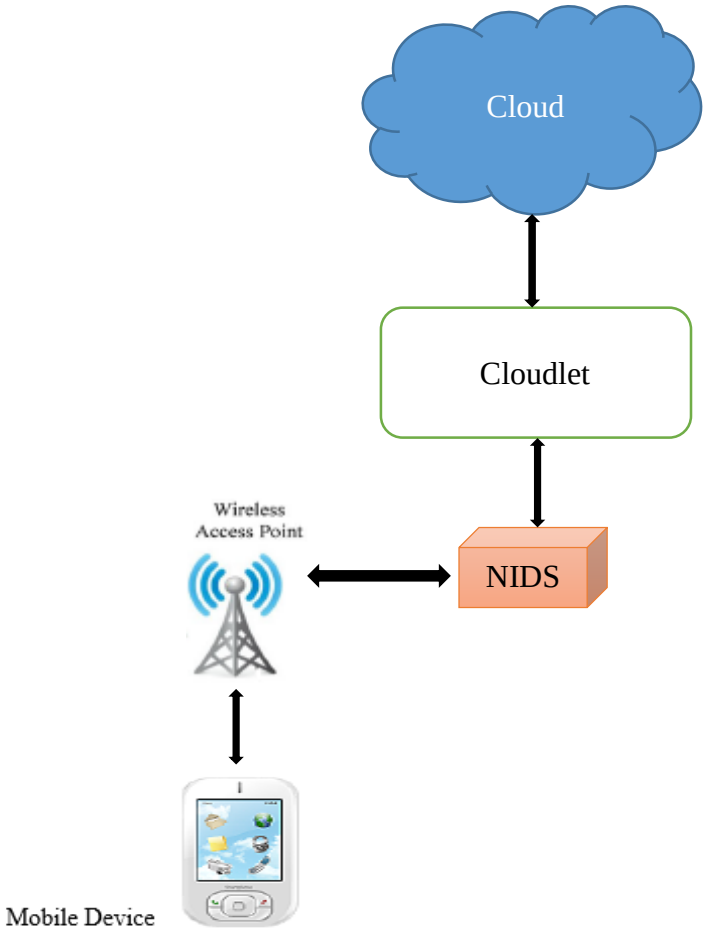


*Figure 4. 1 High level architecture of the proposed integrated system*

### 4.2 Detailed Architecture of the Proposed 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 proposed system is depicted in the following figure.

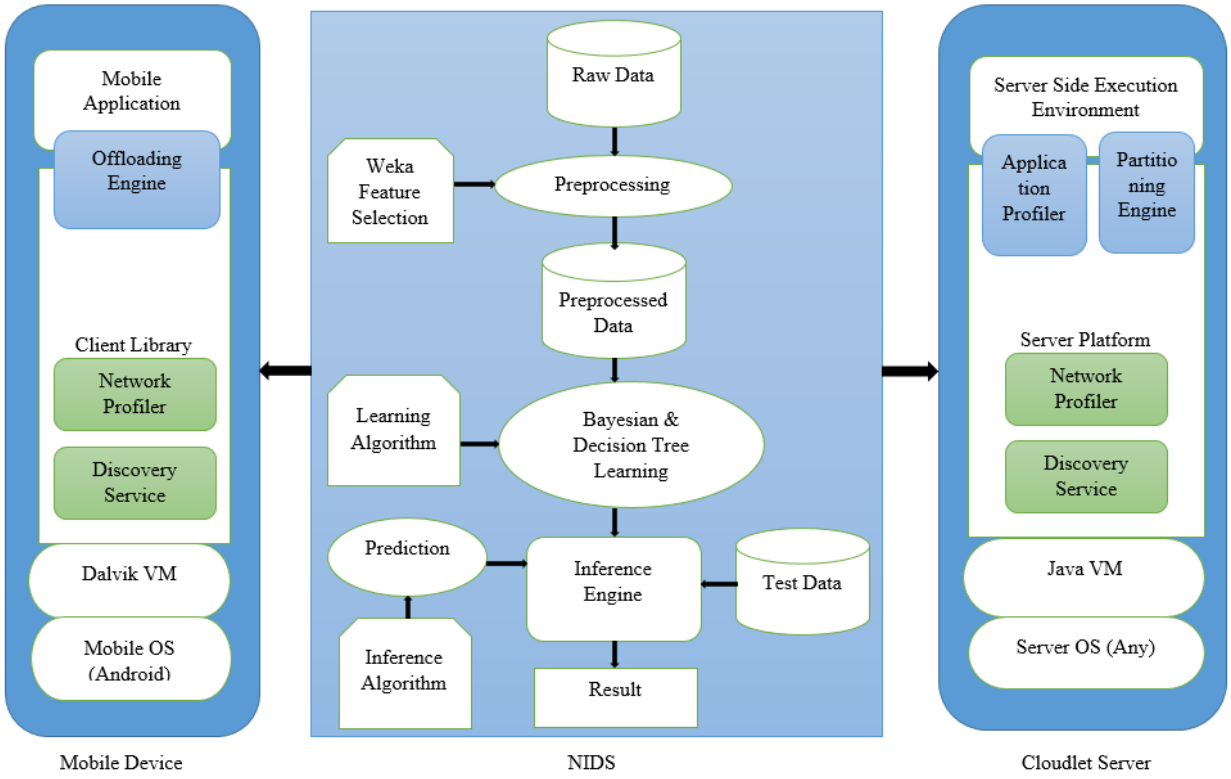


*Figure 4. 2 Architecture of the proposed integrated system*

#### 4.2.1 Mobile Module Components of the Proposed System

The first layer of the system architecture has the operating system of the mobile device (Android). The second layer include the Dalvik Virtual Machine (DVM) for Android operating system. The third layer consists of the Client Library. The Client Library contains the Network Profiler,

Discovery Service and the Offloading Engine components. The fourth top layer is represented by Mobile Application. Network Profiler and Discovery Service are the components that works in both the client and the server side.

### 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 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[40] with some minor adaptation for the purpose of integration with the proposed system.

### II. Network Profiler

This component of the proposed 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.

### III. 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[41]. The proposed 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[40] offloading is executed when RTT is lower than 40ms (40 millisecond). However, Tseghai et al [42] 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 proposed system uses it for offloading decisions. Specifically the proposed 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 4. 1 Pseudo code for offloading engine*

### 4.2.2 Cloudlet Server Module Components of the Proposed System

The server is responsible for checking the status of the NIDS before receiving the offloaded methods. If the NIDS report is normal, the server will receive the offloaded methods, execute the methods and send back the result to the mobile device. The architecture contains four layers. The first layer has the operating system. The second layer includes the Java Virtual Machine (JVM). The third layer consists of the Server Platform. The server platform contains different components such as Discovery Service, Network Profiler, Application Profiler and Partitioning Engine. The fourth top layer contains Server Side Execution Environment (SSEE).

```
Input: NIDS Report
Process:
BEGIN
Read Report generated by NIDS
If status is Anomaly
Deny service requests and close the stream
End If
END
Output: Alarm to the cloudlet administrator
```

*Pseudo code 4. 2 Pseudo code for cloudlet server response during attack.*

#### 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 proposed 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.

#### 4.2.2.1 Call Graph Model

The call graph is a focused weight graph which demonstrate the vertex as a method and the edge as the calling relationship which exist between the methods[43]. In this system the vertex is represented by the execution time of the methods and the edge represent the number of calls between the methods. The figure below shows the example of call graph. The numbers inside the circles represents the execution time of the methods in milliseconds and the number at the edges represents the number of calls between the methods. The letters are the name of methods.

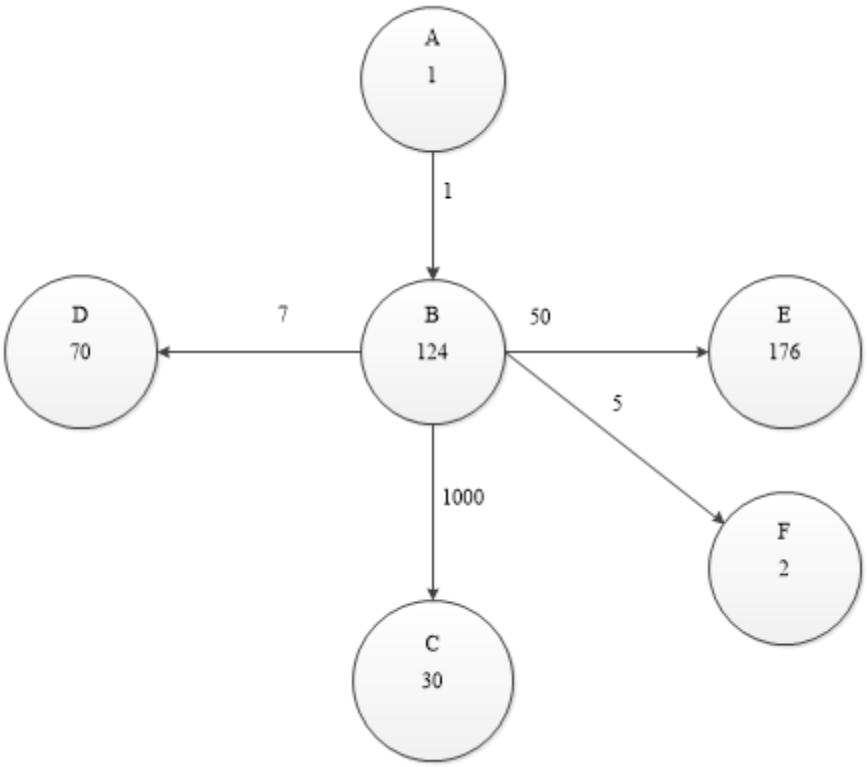


*Figure 4. 3 Call graph*

## 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[42]. 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 above call graph.

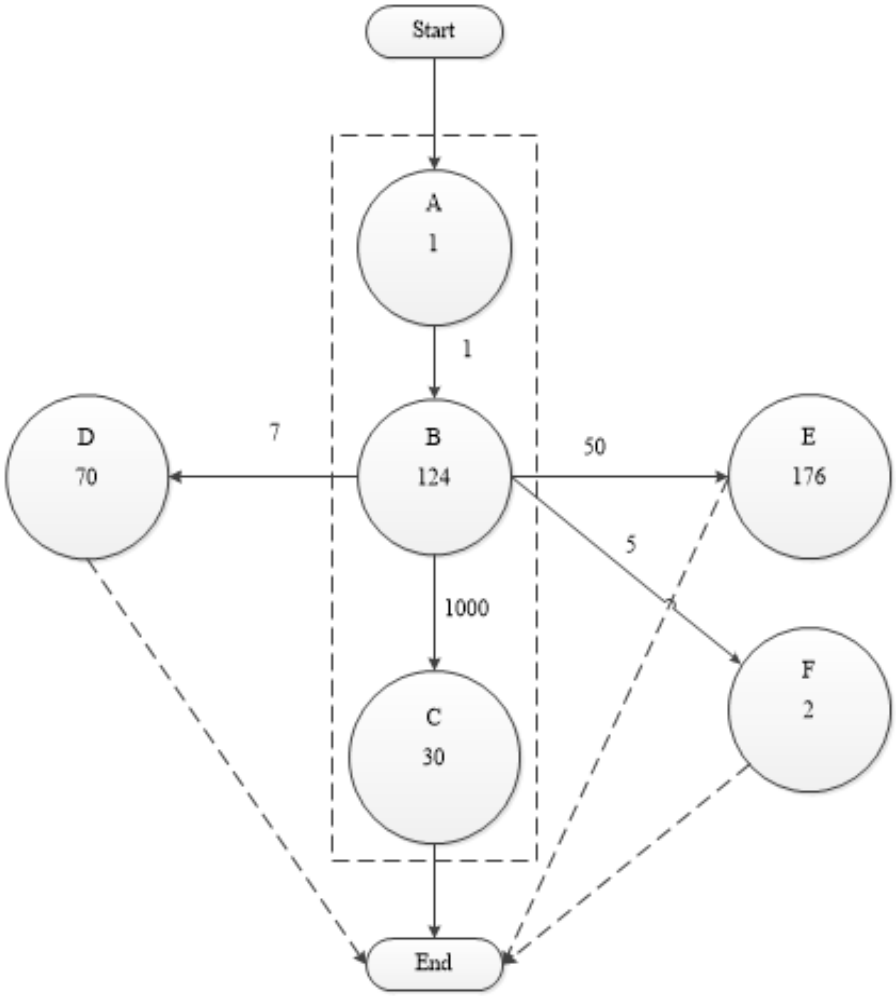


*Figure 4. 4 The first critical path in the call graph*

The first critical path in the call graph is A, B and C. After finding the first critical path the algorithm prune it and go to the process of finding the next critical path. The next figure shows the pruned call graph.

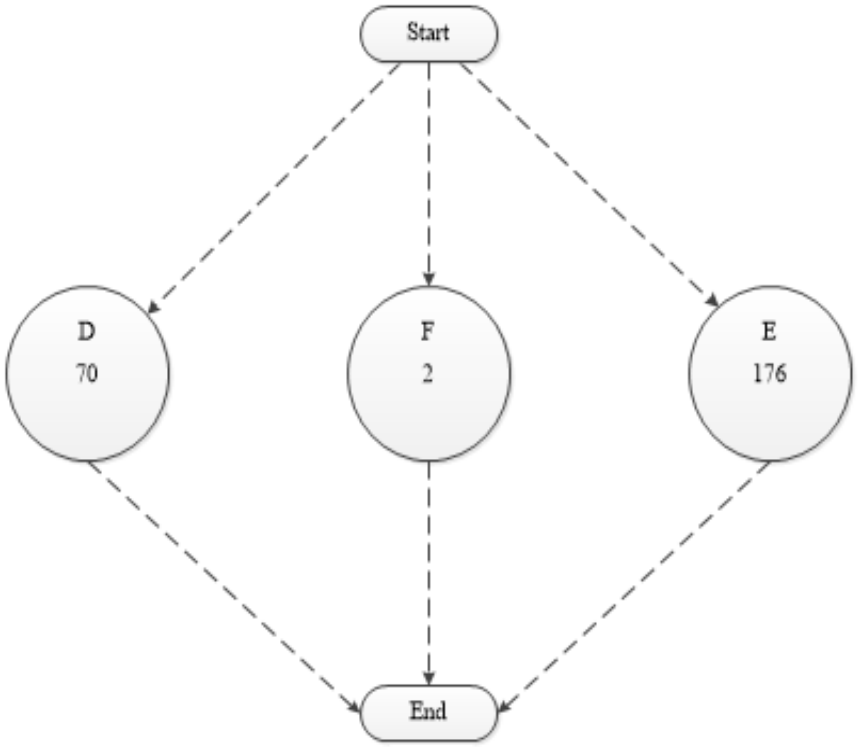


*Figure 4. 5 The call graph after pruning the first critical path*

#### 4.2.2.2 Application Partitioning Algorithm

This thesis assumes that executing application on the remote server is much better than executing application on the phone. Because of this assumption application profiling and application partitioning engine is moved by this research to the cloudlet server in order to alleviate the problem around the resource limitations of mobile device unlike most computation offloading frameworks. The only cause that makes the execution of application on remote server more time consuming is the network quality problem. To deter the problem, this research considers the quality of the network during offloading decisions.

```
Input: G = (VT, E)          // The call graph of the application

Process:

BEGIN

count = 1
While there are method nodes in the call graph Do
Initialize Length (vi) = 0, for each vi ϵ VT    // Find the length of the critical path
For each vi ϵ VT Do
    For each edge vi → wj Do
        If Length(wj) ≤ Length(vi) + Execution Cost(wj) + Edge Weight(vi → wj) Then
            Length(wj) = Length(vi) + Execution Cost(wj) + Edge Weight(vi → wj)
            PredCost(wj) = vi
        End If
    End For
End For
loc = 1, maxL = 0
For k = 1 to |VT| Do                    // To find the last node of the critical path
        If Length(vk) ≥ maxL Then
            maxL = L(vk)
            loc = k
        End If
End For
node = Vloc                             // Let node be the last method of the critical path
CPcount = {Vloc}
While node ≠ start Do                   // Add nodes to the current critical path CPcount
        node = PredCost(node)
        CPcount = CPcount U node
End While
G'= (CPcount, ECP)                      // ECP are the edges of the nodes in CPcount
G = G - G'                              // Prune the nodes in the critical path from the call graph
For each uk ϵ CPcount Do
    Annotation(uk) = count              // Change the annotation of the methods
    If |Pred(uk)| == 0 Then             // Add pseudo-edges to the call graph
        G = G U (null, start → uk)
        Else If |Succ(uk)| == 0 then
                G = G U (null, uk → exit)
    End If
End For
count+ = 1
End While
Return CP
END

Output: Optimal partitions of the call graph
```

*Pseudo code 4. 3 Pseudo code for partitioning engine*[42]

*Table 4. 1 The terms that are used in the application partitioning algorithm*

| Terms | Definition |
|---|---|
| CP | Critical path |
| Pred(uk) | The method that calls uk method (i.e. predecessor of node uk) |
| Succ(uk) | The method that is called by the uk method (i.e. successors of node uk) |
| ECP | The edges of the nodes in the critical path |
| G | Call graph |
| VT | Set vertices (nodes) in the call graph |
| E | Edges of the call graph |

### 4.2.3 Network based Intrusion Detection System Module Components

#### I. 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[44].

#### II. 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.

## III. 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 the best way to develop the system. Naïve Bayes is acknowledged as graphical modeling tool and used to model decision problems having uncertainty.

Naïve Bayes offer better detection rate on three main attacks such as probing, user to root and remote to user. Decision Tree have better accuracy[45]. By combining the two algorithms this thesis can achieve the best performance in detecting intrusions. The table below summarizes the advantages and disadvantages of each algorithm.

*Table 4. 2 Advantages and disadvantages of detection algorithms*[46]

| Techniques | Advantages | Disadvantages |
|---|---|---|
| Fuzzy Logic | -Reasoning is Approximate rather than precise.<br>-Effective, especially against port scans and probes. | -High resource consumption.<br>-Hard to develop a model from fuzzy system. |
| Genetic Algorithm | -Biologically inspired and employs evolutionary algorithm.<br>-Uses the properties like Selection, Crossover, and Mutation.<br>-Capable of deriving classification rules and selecting optimal parameters. | -No constant optimization response time. |
| Neural Network | - Ability to generalize from limited, noisy and incomplete data. | - Need long training time.<br>- Greater computational burden. |

| | | |
|---|---|---|
| | - Has potential to recognize future unseen patterns. | |
| Naïve Bayes | -Encodes probabilistic relationships among the variables of interest.<br>-Ability to incorporate both prior knowledge and data.<br>- Used to model decision problems containing uncertainty.<br>-Better detection rate | - Lack of available probability data. |
| Decision Tree | -Make decision according to particular attribute.<br>-More accurate. | -It is unstable. |

The following pseudo code demonstrate how Decision Tree algorithm builds the decision Tree.

1. Check for base classes
2. For each attribute A find the information gain from splitting A
3. Let $A^{att}$ is a best attribute with the highest information gain
4. Create a decision node that splits the $A^{att}$
5. Recurs on the sub lists obtained by splitting a best and add those nodes as children's of nodes.

*Pseudo code 4. 4 Pseudo code for building Decision Tree*

Naïve Bayes is the simplest form of Bayesian Network classifier. This method assumes that all the features are independent which is called class conditional independence. The proposed system uses

the Naïve Bayes algorithm. Bayes rule calculate the posterior probability P(C|x) using likelihood P(x|C) and prior P(C) with evidence P(x) as the formula on next page.

$$P(ci \mid x) = \frac{P(x \mid ci) \times P(ci)}{P(x)}$$

$$P(ci \mid x) = \frac{P(a1 \mid ci) \times P(a2 \mid ci) \times \ldots \times P(an \mid ci) \times P(ci)}{P(x)}$$

Where Ci is a possible value in the session class and X is the total evidence on attributes nodes.

***Input:*** New Dataset with reduced dimension

***Process:***

***BEGIN***

***Try***

Read features from dataset with reduced dimension

Train Decision tree classifier using training dataset

Train Naïve Bayes classifier using training dataset

***End Try***

***END***

***Output:*** The trained model

*Pseudo code 4. 5 Pseudo code for training phase*

## IV. 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 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.

```
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 4. 6 Pseudo code for detection stage*

### 4.3 Communication Model of the Proposed System

The communication module is responsible for transferring the data back and forth between the cloud and the mobile device. To allow communication between the mobile device and the cloudlet server this research uses the conventional client-server communication model. This model have two technological options which is Remote Procedure Calls (RPC) or Remote Method Invocation (RMI). In order to choose the best communication technology that is suitable for computation offloading, this research compares the two technology.

#### A. Remote Procedure Calls (RPC)

RPC is a technology standard which allows inter process communication between computer device and mobile devices in distributed computing environment. The main advantage of RPC is its independence from network detail and its language neutral. This allows programming easier and lets RPC to work on any network despite the protocol and physical variances[47].

### B. Remote Method Invocation (RMI)

To support object oriented programming, RMI is an API (Application Programming Interface) that implements RPC in java. This allows Java to make method calls from another Java Virtual machine residing on the same computer or a remote one. It's programmer friendly characteristics and its easiness to use is among the advantages of RMI. The table below summarizes the comparison between RPC and RMI[47].

*Table 4. 3 The comparison of RMI and RPC*

| Criteria | RMI | RPC |
|---|---|---|
| Programming paradigm | Object-oriented | Not object-oriented |
| Programming Language | It's a pure java implementation | It's programming language neutral |
| Invocation mechanism | Remote methods are invoked using proxy object | Remote functions are invoked using proxy function |
| Performance | Slower than RPC since RMI involves execution of java bytecode. | Since less computation is required to invoke remote functions, it is slightly better than RMI |
| Android support | Not supported by Dalvik VM[48]. | Since RPC is programming language neutral, it is interoperable with different programming languages and platforms. |

To summarize, because of its support to Dalvik virtual machine, language independence and best performance this research chooses RPC over RMI. In addition to that when a remote method is called using RMI, the state of the object is also transmitted within the method. This way of transmission increase the overhead of the mobile device. This means the idea behind computation

offloading is worthless. Due to this benefits RPC based technology is appropriate for the proposed system.

### 4.4 Execution Flow of the Proposed 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 proposed system offloads computations at method level granularity.

The proposed 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. The figure 4.6 shows the flow of execution in the proposed integrated system.

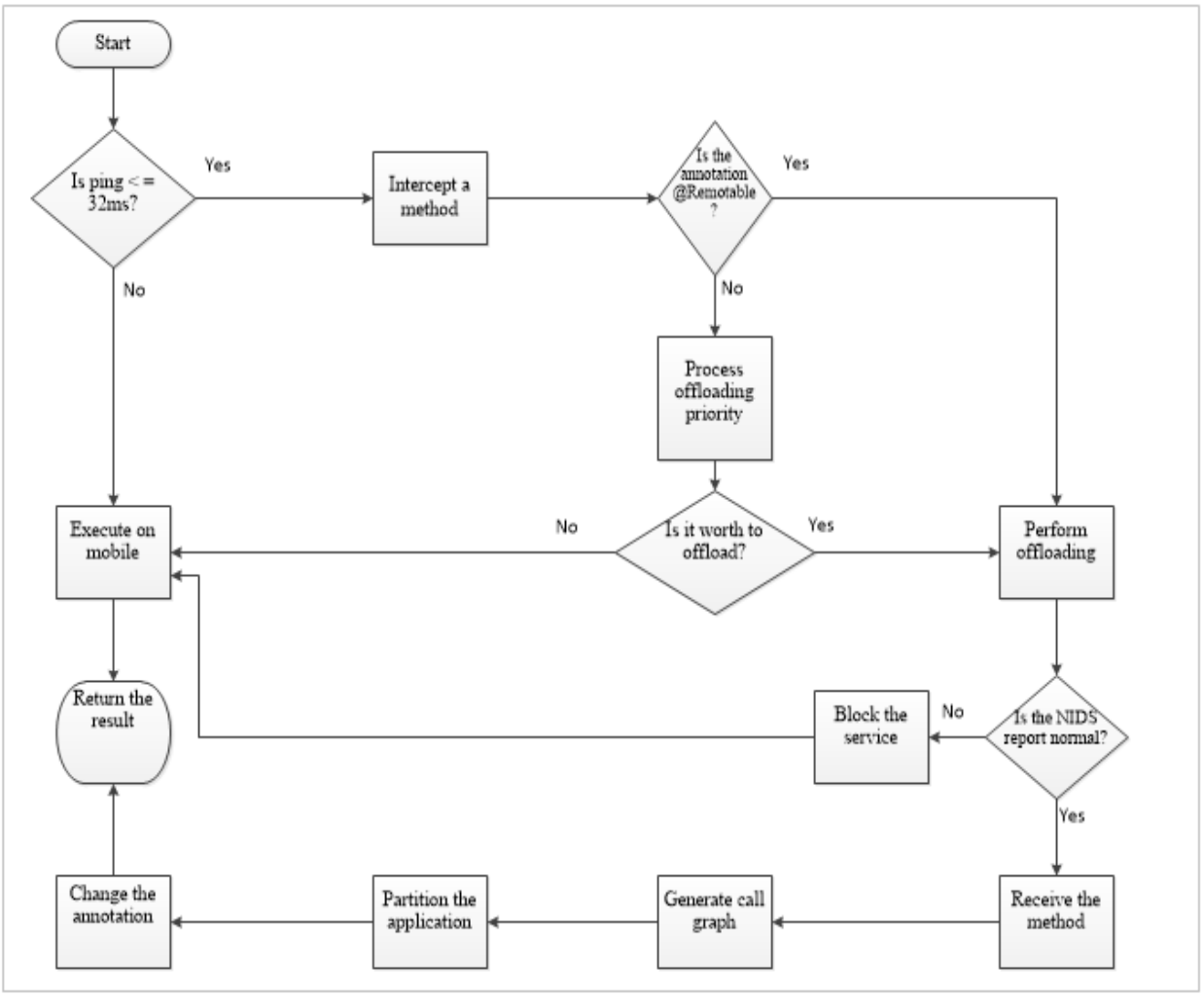


*Figure 4. 6 The execution flow of the proposed system*

### 4.5 Summary

The proposed system views the security of mobile device from the perspective of computation offloading. This mechanism alleviate the problem of existing systems by stopping the offloading process immediately at the time of attack and deter information theft. The network based intrusion detection module uses the combination of Naïve Bayes and Decision Tree algorithms to detect attacks and to generate report to cloudlet server. Then the report will be used by the cloudlet server to allow the service or to deny service requested by the mobile device. During attacks, denying service requests by the cloudlet server allows the proposed system to alleviate the problem which is happened by compromising the communication between the mobile device and cloudlet server and making the system to act like normal operation time and capture the offloaded data without being noticed by the mobile device and cloud administrator.

The computation offloading process is implemented at method level granularity. This helped the system to use RPC technology for communication purpose since RPC worked only at method level. The proposed system also uses critical path method which is applied on the call graph to partition the application. Additionally the system deploys application partitioning and profiler at the server side to reduce the overhead encountered during application partitioning and profiling at the mobile device side.

# CHAPTER FIVE: IMPLEMENTATION AND EVALUATION

## 5.1 Implementation of the Proposed System

The proposed 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.

### 5.1.1 Tools Used

Numerous tools are used in the development of the proposed system. Laptop computer has been configured and used as a cloudlet server and android based mobile device was used to install the application. For communication purpose the two devices are connected to WLAN (Wireless local area network). The laptop computer is also used for deploying the network based intrusion detection system.

The hardware and software specifications that are used to test the proposed system is discussed as follows:

**Hardware Specifications**

1. Mobile device
   - Brand: Samsung
   - Android version: 4.1.2
   - Processor: Dual Core
   - Main memory: 1GB
2. Computer
   - Processor: Intel (R) Pentium (R) Dual CPU T3400 @ 2.16GHz
   - Main memory: 2GB
   - System type: 32 bit Operating system
   - Java version: 1.8
3. Wireless device
   - Vendor: D-Link
   - Link speed: 200Mbps

**Software Specifications**

The following list of software tools have a great contribution for the accomplishment of the proposed system.

- Eclipse Neon
- Android SDK (Software Development Kit)
- ADT plugin for Eclipse (Android Development Tool)
- WEKA API
- Connectify Hotspot
- Maven

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.

### 5.1.2 Configuration of the Proposed 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[40].

- 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.

config.properties

```
1  cloudlet.interface.ip=192.168.81.1
2
3  #portable (any platform can to use) default services, can't to be deployed
4  service.portable.pingtcp.name=ping_tcp
5  service.portable.pingtcp.port=40000
6  service.portable.pingudp.name=ping_udp
7  service.portable.pingudp.port=40001
8  service.portable.saveprofile.name=save_profile_results
9  service.portable.saveprofile.port=40011
10
11 #turn on or off, the multicast discovery response
12 service.portable.multicastdiscovery=on
```

*Figure 5. 1 Config file of the cloudlet*

### 5.2 Evaluation of the Proposed System

The evaluation of the proposed 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 proposed 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.

#### 5.2.1 NSL-KDD Dataset

The NSL-KDD dataset is proposed to fix the inherent issues of the KDDCUP'99 dataset. Numerous researchers uses KDDCUP'99 dataset to develop intrusion detection system based on anomaly detection.

However, Tavallaee et al directed a quantifiable dissection on this dataset and found two key issues that essentially impacted the execution of evaluated frameworks, and realizes a poor review of anomaly detection approaches. To fathom these issues, they proposed new dataset, NSL-KDD, which involves chose records of the entire KDD dataset[49].

The following are the benefits of the NSL-KDD over the KDDCUP'99 dataset:

1. It eliminates the extra records in the training set, therefore the classifiers won't be biased towards frequent records.

2. The size of picked records from each difficulty level gathering is inversely proportional to the percentage of records in the formerly KDD dataset. So, the classification rates of unique machine learning frameworks vary in a broader reach, which makes it more capable to have an exact evaluation of different learning strategies.

3. The extent of records in the training and test sets is realistic, which makes it practical to run the examination on the entire set without the need to arbitrarily select a little partition. Hence, eventual outcomes of various investigation will be reliable and adequate.

The NSL-KDD data include 4 sorts of attacks such as: DOS, Probe, R2L, and U2R.

**Denial of Service Attack (DOS)** is an attack where an attacker makes some computing or memory resource too busy or too full to handle justifiable requests, denying legal users access to a device.

**Probing Attack** is an attempt to collect information around a system of machines so as to seek out identified vulnerabilities. This is not an attack but it is an attempt to learn information to facilitate an attack.

**User to Root Attack (U2R)** is an attack in which the user with user permission tries to access or gain root permission.

**Remote to User Attack (R2L)** happens when an attacker sends packets to a machine over a network that exploits the machine’s vulnerability to grasp native access as a user illegally.

### 5.2.2 BenchImage Application

BenchImage is image processing android application which is developed to test computation offloading algorithms in MPOS framework. The algorithms applies some effects (i.e., filters) such as cartooning, sharpening and Red Ton to the images which are available in the application. The images have different resolutions ranging from 0.3MP to 8MP. The user can choose one of these stored images and apply the desired filter on them[40].

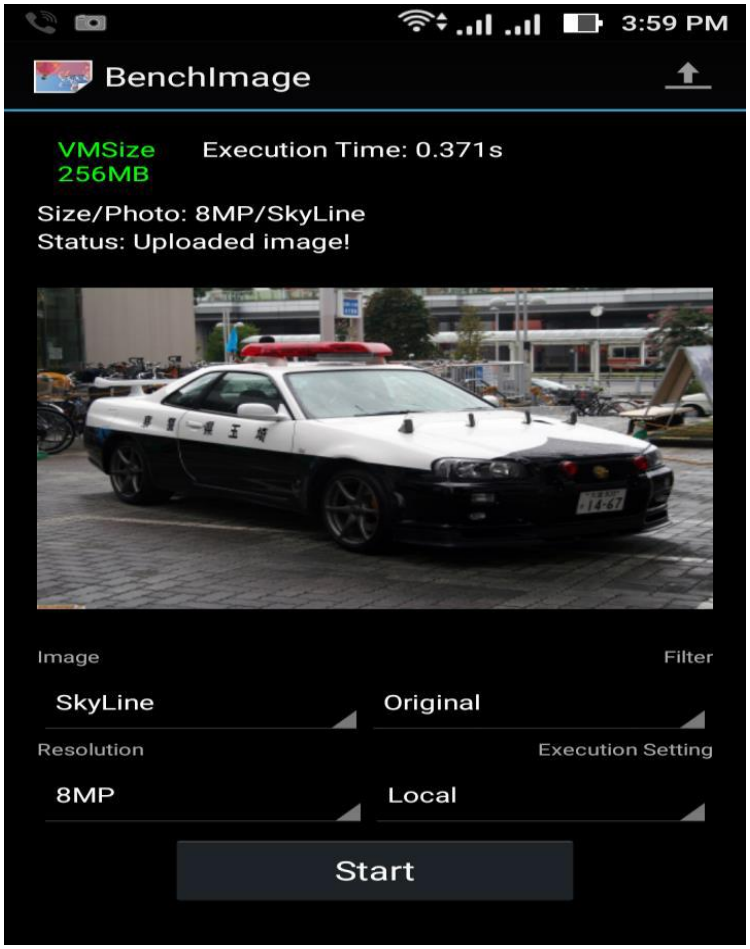


*Figure 5. 2 Screenshot of the BenchImage application*

The user can choose one of the BenchImage menu to accomplish the desired task which is:

- Image
- Resolution
- The execution environment (local or cloudlet)
- The filter to be applied (cartooning, sharpening and Red Ton)

In order to find the filtered image for another time, the application stores the image in BenchImageOutput folder on the mobile device.

### 5.3 Results and Discussion

This Section covers the experiments performed during the entire thesis. 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.

#### 5.3.1 Experiment on Network based Intrusion Detection System

The intrusion detection system uses combined algorithms such as Naïve Bayes and Decision tree. Since the main aim is to reduce the false alarm rate, the results in contrast with other algorithms will be discussed on next pages. The figure 5.3 shows how the proposed combined system trained with the training dataset with reduced dimension and then detect intrusive actions from the new dataset.

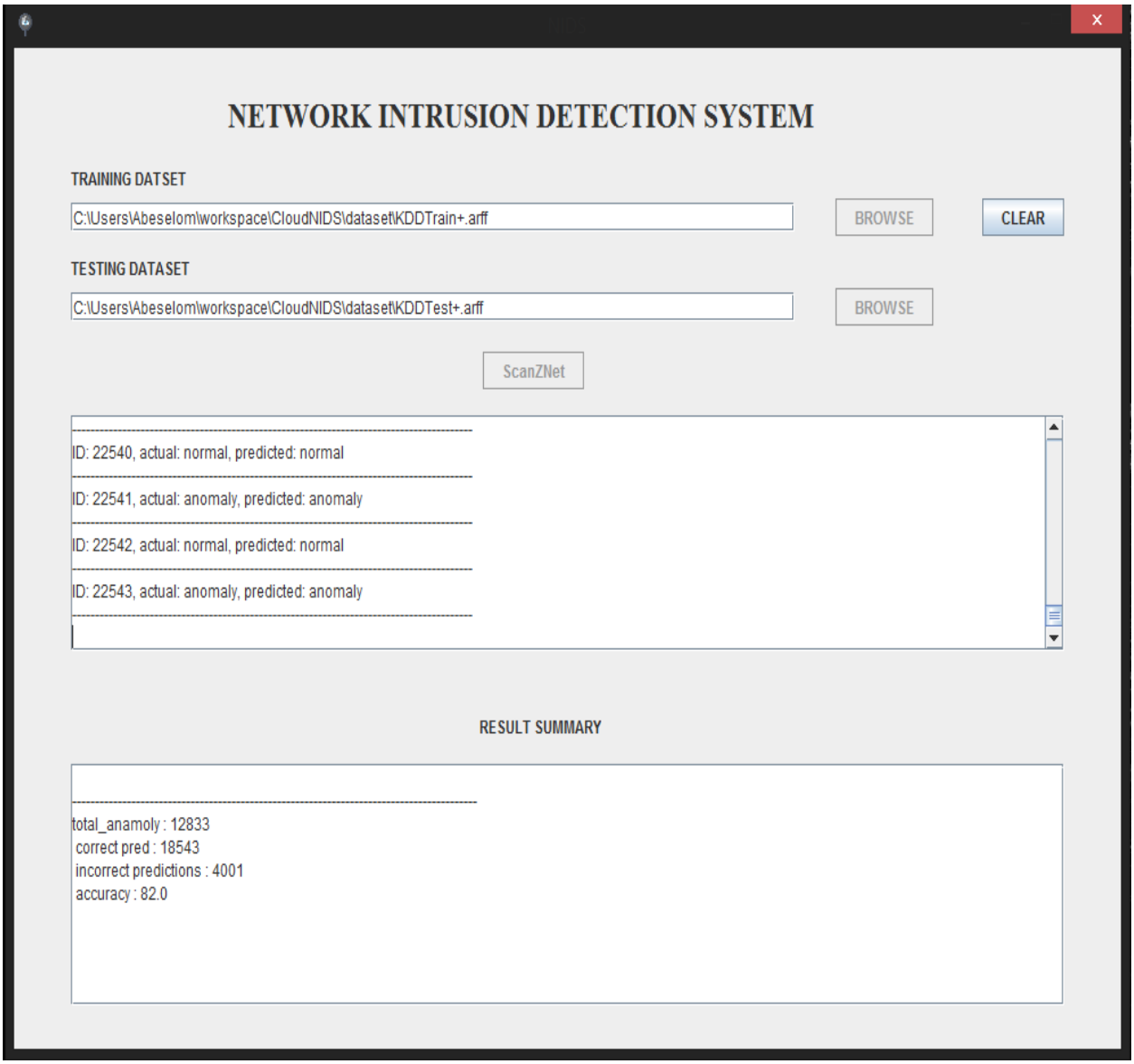


*Figure 5. 3 Performance of the combined intrusion detection system*

### 5.3.1.1 Performance Evaluation

The performance of the proposed 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 table 5.1 is a standard confusion matrix.

*Table 5. 1 Confusion Matrix*

| | | **Actual class** | |
|---|---|---|---|
| | | **Negative Class (Normal)** | **Positive Class (Attack)** |
| **Predicted Class** | **Negative Class (Normal)** | True Negative (TN) | False Negative FN) |
| | **Positive Class (Attack)** | False Positive (FP) | True Positive (TP) |

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 proposed 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. 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 proposed 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.

The following figures shows the performance of both the Naïve Bayes and Decision tree algorithms giving the same datasets.

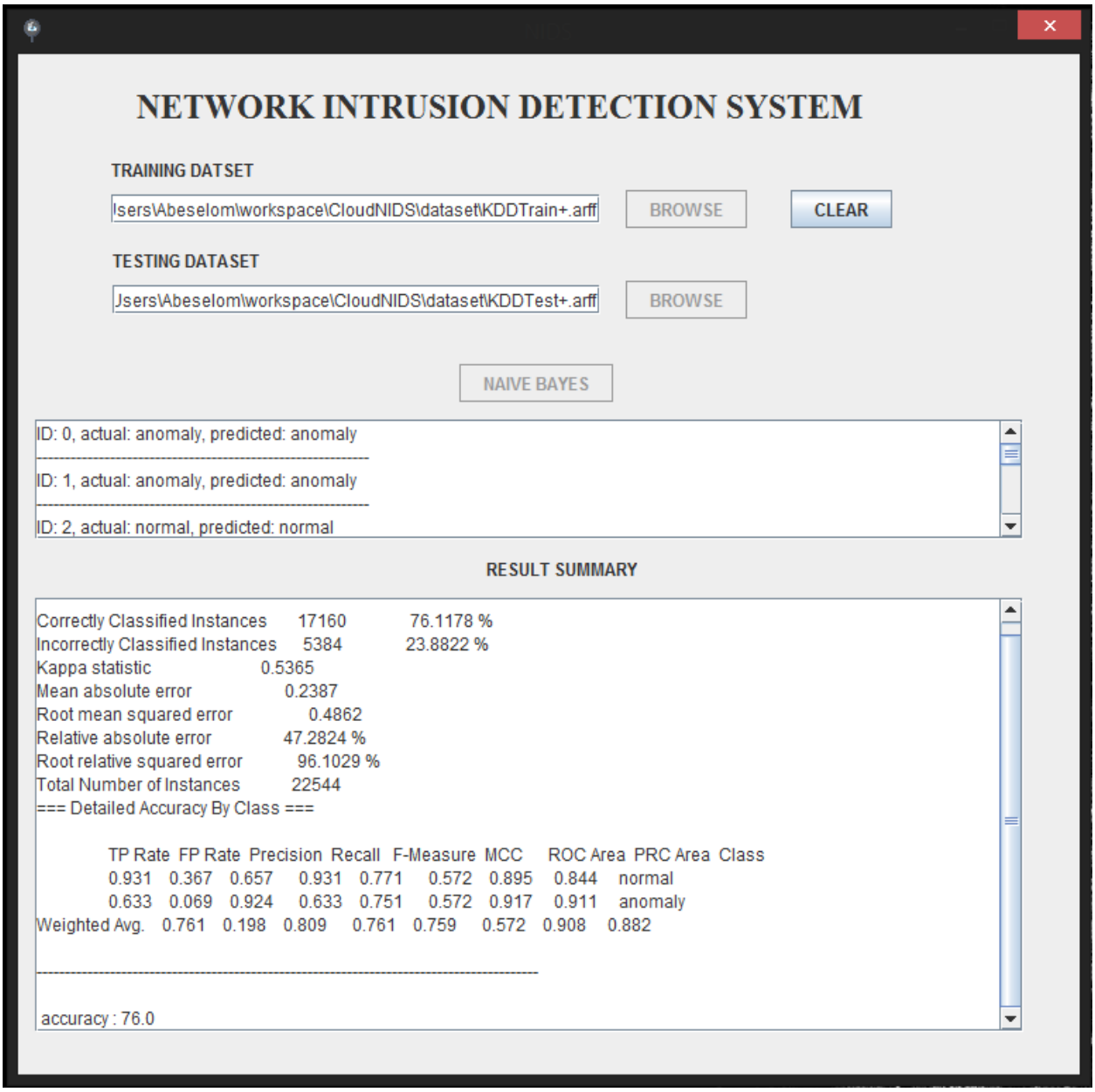


*Figure 5. 4 Performance of Naïve Bayes algorithm*

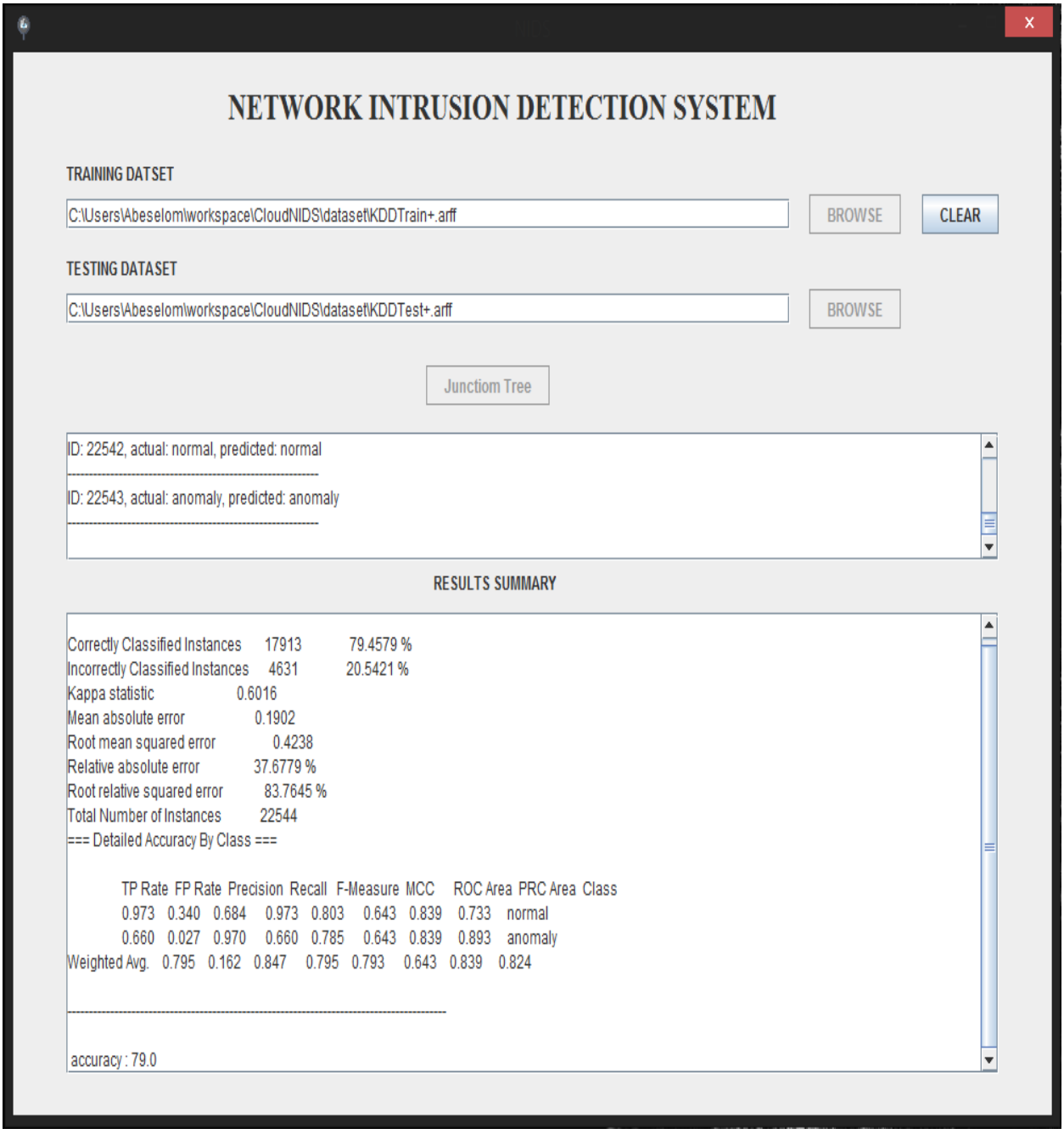


*Figure 5. 5 Performance of Decision Tree algorithm*

### 5.3.2 Experiment on Integration of NIDS with Mobile Application Offloading

The main purpose of integrating network based intrusion detection system with mobile application offloading is to reduce the security risks encountered during offloading computation intensive part of mobile application to the cloudlet. As the thesis mentioned in previous chapter the offloading

process is implemented at method level granularity. This allows the research to use RPC communication technology. The figure below shows the initialization of services at the cloudlet server.

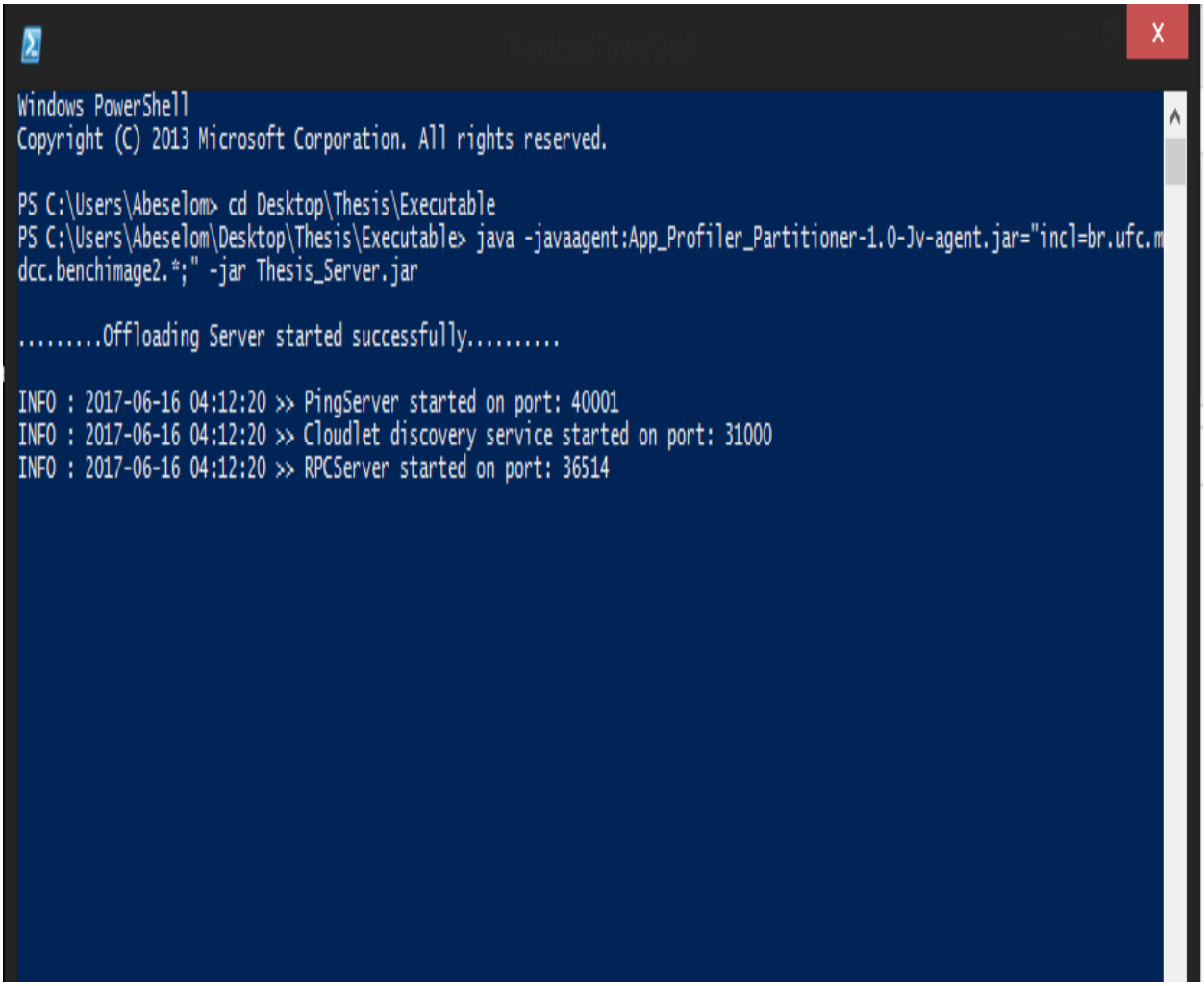


*Figure 5. 6 Service initialization of cloudlet server*

The next figure demonstrate how the proposed system handles the offloaded methods during the normal time of operations.

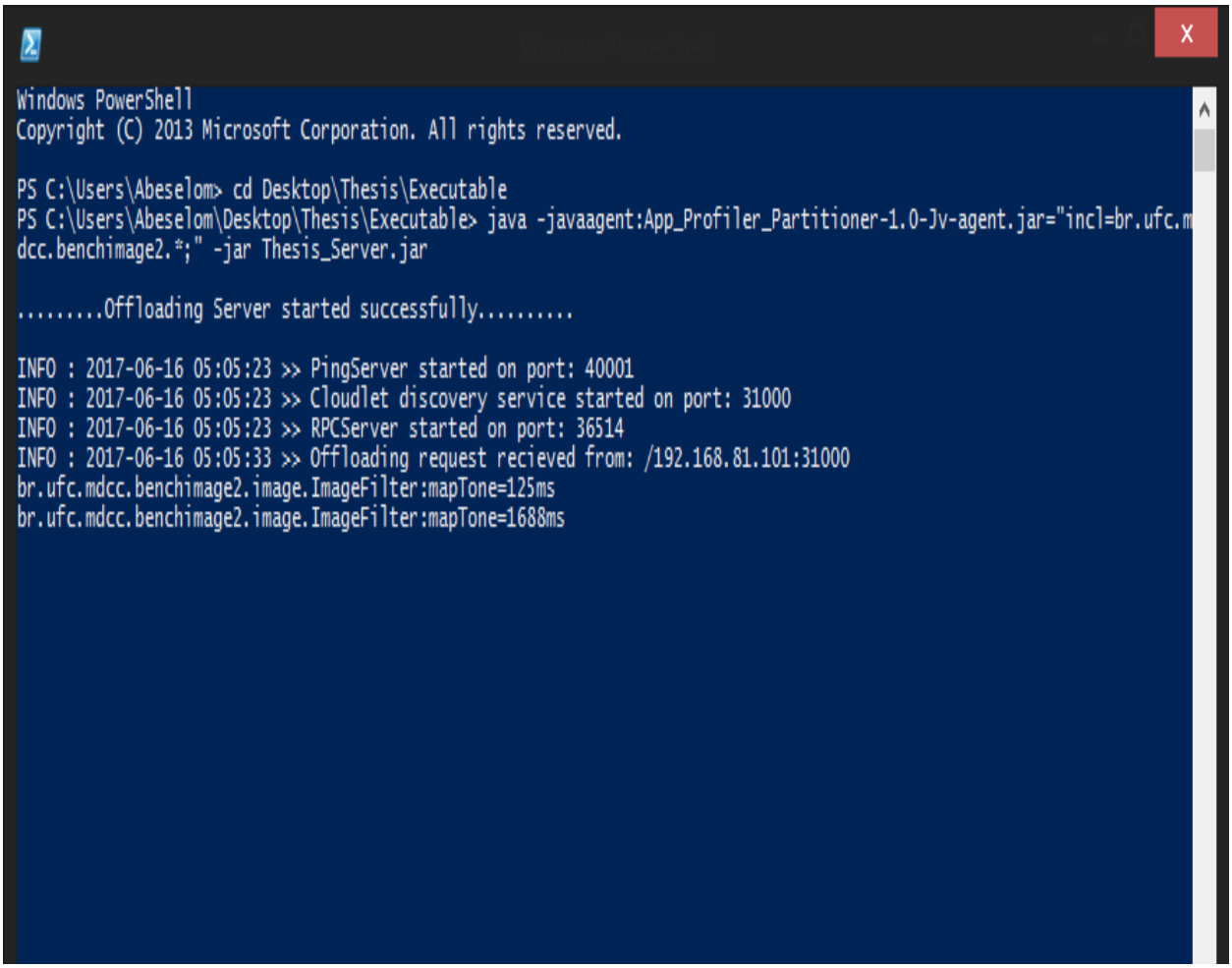


*Figure 5. 7 Handling of service request by the cloudlet server*

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 figure 5.8.

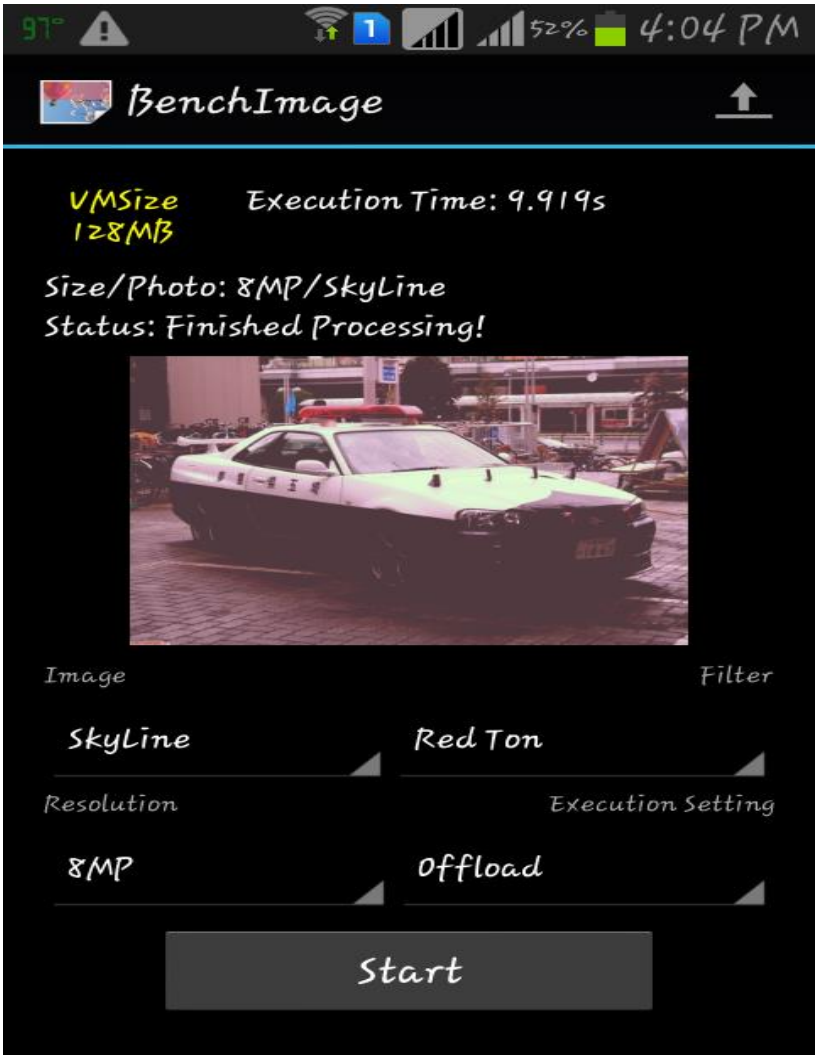


*Figure 5. 8 The result of applying Red Ton filter on the image*

When cyberattack happens the proposed 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 cloud 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. The figure 5.9 demonstrate how the proposed integrated system responds during attacks.

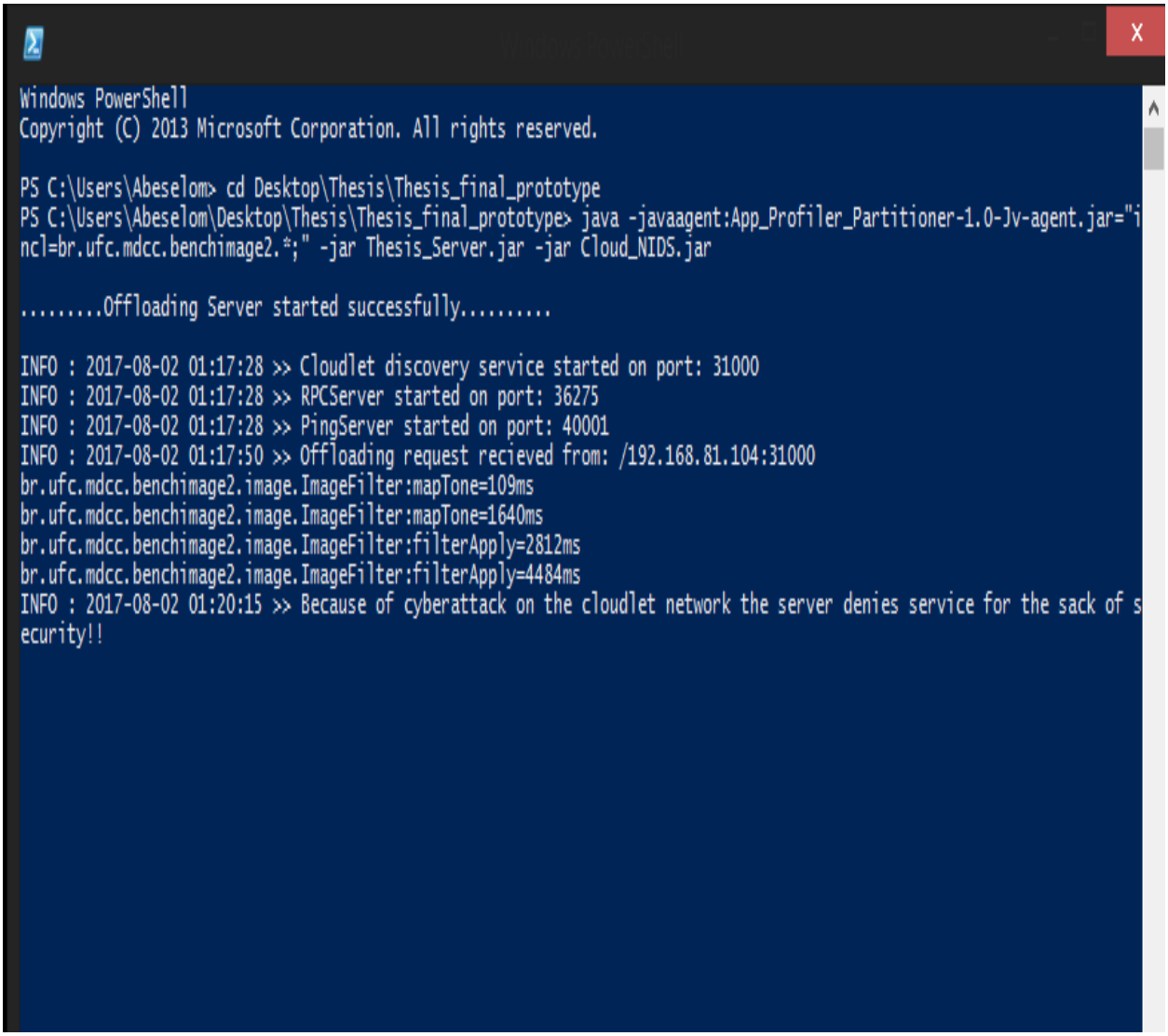


*Figure 5. 9 The proposed integrated system response during attacks*

As the figure 5.9 shows the reaction by the system during attack works well. This indicates that the proposed system is acting as expected at the time of attacks. Finally, this research achieved the main objective of the thesis which is integrating intrusion detection system with mobile cloud offloading to securely offload mobile applications to the cloud.

# CHAPTER SIX: CONCLUSION AND FUTURE WORK

## 6.1 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 thesis 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 proposed 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. Moreover, the security of mobile cloud computing have not been seen from the perspective of computation offloading by any of the scholars who develop the intrusion detection system for the mobile cloud computing.

This thesis tries to fill the gap around the shortcomings of the current intrusion detection system by proposing and implementing the integration of network based intrusion detection system with mobile computation offloading. The proposed system is intended to deter the data theft by the intruders or to reduce the vulnerability of the offloaded computation from being accessed by attackers. The proposed system also fills the time gap between the detection of intrusion and taking action by the cloud administrator which means there is a time gap between intrusion identification and taking care of the intruders. During this time gap the intruders may gain access of millions of user data.

The proposed system used the combined Naïve Bayes and Decision Tree algorithms for classifying and learning process of intrusion detection module. This anomaly based component is trained using a simulated dataset NSL-KDD. The offloading engine at the mobile side considers the quality of the wireless network during offloading decision. The intercommunication between each

method and their execution time is used for partitioning the application at the cloudlet server side. The system also used client/server communication model and Asynchronous Remote Procedure Calls (RPC) to enable communication between the mobile devices and the remote servers. The research implements the proposed system by using configured laptop as cloudlet server, Samsung mobile device for deploying the android application and Connectify software as wireless hotspot. Finally, the proposed 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 proposed system can be effective means of security for mobile cloud computing.

### 6.2 Future Work

Beside the promising result of the proposed system, there is a need to improve some aspects of the system. This thesis believes that more research is needed to make the system effective and accurate. There is also a need to test the system using current wireless attack datasets.

Some potential future works that could be a continuation of this work is as follows:

- Developing a more accurate model that can be used in real-time for detecting and classifying anomaly with minimum false alarms and less time.
- Implementation of the proposed system in a public cloud environment.
- Reducing the burdens on the developers which is leaving the process of identification of computation intensive methods to the system.
- The proposed system supports only Android mobile applications and enabling the system to support multiple mobile platforms can widened the delivery of computation offloading services.

## 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] G. Orsini, D. Bade, and W. Lamersdorf, "Context-Aware Computation Offloading for Mobile Cloud Computing : Requirements Analysis , Survey and Design Guideline," *Procedia - Procedia Comput. Sci.*, vol. 56, no. MobiSPC, pp. 10–17, 2015.

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

[4] A. Houmansadr, S. A. Zonouz, and R. Berthier, "A Cloud-based Intrusion Detection and Response System for Mobile Phones," 2012.

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

[6] A. Bose, X. Hu, K. G. Shin, and T. Park, "Behavioral Detection of Malware on Mobile Handsets," pp. 225–238, 2008.

[7] 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.

[8] B. Sareen, S. Sharma, and M. Arora, "Securing Smartphone Data by Offloading Computation on Cloud," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 4, no. 7, 2014.

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

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

[11] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing : A survey," *Futur. Gener. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[12] N. Khan, A. Noraziah, T. Herawan, and K.- Cloud, "Cloud Computing : Architecture for efficient provision of services," *15th Int. Conf. Network-Based Inf. Syst. Cloud*, pp. 18–23, 2012.

[13] B. K. Sy, “Integrating intrusion alert information to aid forensic explanation : An analytical intrusion detection framework for distributive IDS,” *Inf. Fusion*, vol. 10, no. 4, pp. 325–341, 2009.

[14] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, “A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing,” *IEEE Commun. Surv. TUTORIALS*, vol. 15, no. 3, pp. 1294–1313, 2013.

[15] E. E. Marinelli, “Hyrax : Cloud Computing on Mobile Devices using MapReduce,” vol. 389, no. September, 2009.

[16] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A survey of mobile cloud computing : architecture , applications , and approaches,” *Wirel. Commun. Mob. Comput.*, 2011.

[17] W. Zhang, S. Tan, F. Xia, X. Chen, and Z. Li, “A survey on decision making for task migration in mobile cloud environments,” *Pers. Ubiquitous Comput.*, 2016.

[18] S. A. Jalan and V. B. Bhagat, “Mobile Cloud Computing an Efficient Technique for Mobile Users,” vol. 3, no. 3, pp. 145–154, 2014.

[19] M. R. Momeni, “A Survey of Mobile Cloud Computing : Advantages , Challenges and Approaches,” *Int. J. Comput. Sci. Bus. Informatics IJCSBI.ORG*, no. July, 2015.

[20] H. Wu, “Analysis of Offloading Decision Making in Mobile Cloud Computing,” 2015.

[21] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, “COSMOS : Computation Offloading as a Service for Mobile Devices,” 2014.

[22] H. FLORES, “Service-situated and Evidence-mindful Mobile Cloud Computing,” *Univ. Tartu Press*, 2015.

[23] K. Kumar and J. Liu, “A Survey of Computation Offloading for Mobile Systems,” *Mob. Netw. Appl.*, 2012.

[24] B. Chun and A. Patti, “CloneCloud : Elastic Execution between Mobile Device and Cloud,” pp. 181–194, 2011.

[25] E. Cuervo, A. Balasubramanian, and D. Cho, “MAUI : Making Smartphones Last Longer with Code Offload,” pp. 49–62, 2010.

[26] S. A.Jalan and V. B.Bhagat, “mobile cloud computing an efficient technique for mobile users,” 2014.

[27] R. Chow *et al.*, “Controlling Data in the Cloud : Outsourcing Computation without Outsourcing Control,” pp. 85–90, 2009.

[28] “Introduction-to-intrusion-detection-systems,” *www.lifewire.com*. [Online]. Available: https://www.lifewire.com/introduction-to-intrusion-detection-systems-ids-2486799.html. [Accessed: 17-May-2017].

[29] S. J, *CISSP Professional: Certified Information Systems Security Professional Study Guide*. Sybex, Incorporated, 2005.

[30] “What is Intrusion Detection?,” *Midmarket IT Security*. [Online]. Available: http://searchmidmarketsecurity.techtarget.com/sDefinition/0,,sid198_gci295031,00.html . [Accessed: 17-May-2017].

[31] Jeams D. and S. D., *Cisco Security Professional’s Guide to Secure Intrusion Detection Systems (IDS)*. Syngress, 2003.

[32] Z. Inayat, A. Gani, N. Badrul, A. Shahid, and M. K. Khan, “Cloud-Based Intrusion Detection and Response System : Open Research Issues , and Solutions,” *Arab J Sci Eng*, pp. 399–423, 2017.

[33] H. Jhala, “Cloud-based Intrusion Detection and Response Mechanism for Smartphones,” *Int. J. Adv. Res. Eng. Sci. Technol.*, vol. 3, no. 11, pp. 102–107, 2016.

[34] 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.

[35] H. Wang, H. Zhou, and C. Wang, “Virtual Machine-based Intrusion Detection System Framework in Cloud Computing Environment,” *J. Comput.*, vol. 7, no. 10, pp. 2397–2403, 2012.

[36] J. Nikolai, “Detecting Unauthorized Usage in a Cloud using Tenant,” 2010.

[37] I. Gul and M. Hussain, “Distributed Cloud Intrusion Detection Model,” *Int. J. Adv. Sci. Technol.*, vol. 34, pp. 71–82, 2011.

[38] M. Miettinen, “Host-Based Intrusion Detection for Advanced Mobile Devices,” *20th Int. Conf. Adv. Inf. Netw. Appl.*, 2006.

[39] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith, “Nymble : Blocking Misbehaving Users in Anonymizing Networks,” *IEEE Trans. DEPENDABLE Secur. Comput.*, vol. 8, no. 2, pp. 256–269, 2011.

[40] 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.

[41] Q. Wang, “Restart in Mobile Offloading,” 2015.

[42] A. Tseghai, “A Lightweight Computation Offloading System for Mobile Cloud Computing,” 2017.

[43] Y. Zhang, H. Liu, L. Jiao, and X. Fu, “To offload or not to offload : an efficient code partition algorithm for mobile cloud computing,” *IEEE 1st Int. Conf. Cloud Netw.*, pp. 80–86, 2012.

[44] D. Tigabu, “Constructing Predictive Model for Network Intrusion Detection,” 2012.

[45] N. Ben Amor, S. Benferhat, and Z. Elouedi, “Naive Bayesian Networks in Intrusion Detection Systems,” 2000.

[46] M. Zewdu, “Intrusion Detection System using hybrid detection approach,” 2015.

[47] WWW.Differencebetween.com, “Difference Between RPC and RMI.” [Online]. Available: http://www.differencebetween.com/Home / Technology / IT / Applications / Difference Between RPC and RMI. [Accessed: 24-May-2017].

[48] N. S. Hauser and R. Friedman, “COARA : Code Offloading on Android with RMI and AspectJ,” 2014.

[49] M. Friedman, E. Bagheri, and W. Lu, “A Detailed Analysis of the KDD CUP 99 Data Set,” *IEEE Int. Conf. Comput. Intell. Secur. Def. Appl*, pp. 53–58, 2009.

# APPENDICES

## Appendix 1: Attack Types of NSL- KDD Dataset

| | |
|---|---|
| DOS | Back, Land, Neptune, Pod, Smurf, Teardrop |
| PROBE | Ipsweep, Nmap, PortSweep, Satan |
| R2L | Ftp_write, Guess_passwd, Imap, Multihop, Phf, Spy, Warezclient, Warezmaster |
| U2R | Buffer_overflow, Loadmodule, Perl, Rootkit |

## Appendix 2: Feature Description for NSL-KDD Dataset

| **Feature Name** | **Description** | **Type** |
|---|---|---|
| duration | Length (number of seconds ) of the connection | continuous |
| protocol_type | Type of the protocol, e.g. tcp, udp, etc. | discrete |
| service | Network service on the destination, e.g., http, telnet, etc. | discrete |
| src_bytes | Number of data bytes from source to destination | continuous |
| dst_bytes | Number of data bytes from destination to source | continuous |
| flag | Normal or error status of the connection | discrete |
| land | 1 if connection is from /to the same host/port; 0 otherwise | discrete |
| wrong_fragment | Number of "wrong" fragments | continuous |
| urgent | Number of urgent packets | continuous |
| hot | Number of "hot" indicators | continuous |
| num_failed_logins | Number of failed login attempts | continuous |
| logged_in | 1 if successfully logged in; 0 otherwise | discrete |
| num_compromised | Number of "compromised" conditions | continuous |
| root_shell | 1 if root shell is obtained; 0otherwise | discrete |
| su_attempted | 1 if "su root" command attempted ; 0 otherwise | discrete |
| num_root | Number of "root" accesses | continuous |

| | | |
|---|---|---|
| num_file_creations | Number of file creation operations | continuous |
| num_shells | Number of shell prompts | continuous |
| num_access_files | Number of operations on access control files | continuous |
| num_outbound_cmds | Number of outbound commands in an ftp session | continuous |
| is_hot_login | 1 if the login belongs to the "hot" list; 0 otherwise | discrete |
| is_guest_login | 1 if the login is a "guest" login; 0 otherwise | discrete |
| Count | Number of connections to the same-host as the current connection in the past two seconds | continuous |
| | *Note: The following features refer to the same-host connection.* | |
| Serror_rate | % of connections that have "SYN" errors | continuous |
| Rerror_rate | % of connections that have "REJ" errors | continuous |
| Same_srv_rate | % of connections to the same service | continuous |
| Diff_srv_rate | % of connections to the different services | continuous |
| Srv_count | Number of connections to the same services as the current connection in the past two seconds | continuous |
| | *Note: The following features refer to the same service connections.* | |
| Srv_serror_rate | % of connections that have "SYN" errors | continuous |
| Srv_rerror_rate | % of connections that have "REJ" errors | continuous |
| Srv_diff_host_rate | % of connections to different hosts | continuous |

**Appendix 3: Sample Code**

// TCP Server Class of the Proposed Integrated System

```
package yb.Thesis.Offload.net;


import java.io.BufferedReader;

import java.io.FileInputStream;
```

```
import java.io.IOException;

import java.io.InputStreamReader;

import java.net.InetAddress;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.SocketException;

import java.net.SocketTimeoutException;

import yb.Thesis.Offload.net.AbstractServer;

import yb.Thesis.Offload.net.util.Service;


public abstract class TcpServer extends AbstractServer {

private ServerSocket serverSocket;

public TcpServer(String cloudletIp, Service service, Class<?> cls) {

super(cloudletIp, service, cls); }

@Override

public void run() {

try {

serverSocket = new ServerSocket(getService().getPort(), 50, InetAddress.getByName(ip));

while (true) {

final Socket connection = serverSocket.accept();

FileInputStream input = new FileInputStream
("C:/Users/Abeselom/Desktop/Thesis/Thesis_final_ prototype/REPORT.txt");

BufferedReader br = new BufferedReader(new InputStreamReader(input));
```

```
for(int i=0;i<input.available();i++)
{
String pdtNo = br.readLine();
String no = pdtNo.trim();
if(no.equals("anomaly")&& !connection.isInputShutdown()){
connection.shutdownInput();
}}
if (connection != null) {
new Thread() {
@Override
public void run() {
try {
clientRequest(connection);
} catch (SocketTimeoutException e) {
logger.info("Socket timeout!", e);
} catch (SocketException e) {
logger.info("Because of cyberattack on the cloudlet network the server denies service for the sack of security!!");
} catch (IOException e) {
logger.info("The generic I/O problem", e);
} finally {
close(connection);
}}
```

```
}.start();

}}

} catch (IOException e) {

logger.info("Any I/O Exception on server socket level!", e);

} finally {

try {

serverSocket.close();

} catch (IOException e) {

}}}

public abstract void clientRequest(Socket connection) throws IOException;

}
```