Mitigation of DDoS Attack on Cloud Computing due to IoT Devices

By
Amna Riaz
NUST201464162MSEECS63114F

Supervisor
Dr. Rizwan Ahmad
Department of Electrical Engineering

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Security (MSIS)

In
Department of Computing (Doc)
School of Electrical Engineering and Computer Science
National University of Sciences and Technology (NUST)
Islamabad, Pakistan.

(July 2018)

Approval

It is certified that the contents and form of the thesis entitled "**Mitigation of DDoS Attack on Cloud Computing due to IoT Devices**" submitted by **Amna Riaz** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Rizwan Ahmad**

Signature: __________________

Date: __________________

Committee Member 1: **Dr. Adnan Khalid**

Signature: __________________

Date: __________________

Committee Member 2: **Dr. Shehzad Saleem**

Signature: __________________

Date: __________________

Committee Member 3: **Dr. Salman Abdul Ghafoor**

Signature: __________________

Date: __________________

Thesis Acceptance Certificate

Certified that final copy of MS/MPhil thesis written by Ms. Amna Riaz, (Registration No NUST201464162MSEECS63114F), of SEECS (School/College/Institute) has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Name of Supervisor: **Dr. Rizwan Ahmad**

Signature: ________________
Date: ________________

Signature (HOD): ________________
Date: ________________

Signature (Dean/Principal): ________________
Date: ________________

Dedication

This work is dedicated
to
my Parents, my Siblings, my Husband, my Teachers and my Friends.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Amna Riaz**

Signature: ________________

Date: ________________

Acknowledgment

I am very thankful to my Allah Almighty for all His help and blessings in every stage of my life.

I am also thankful to my parents, my siblings and to my husband for supporting and encouraging me throughout my research work.

I would like to express my gratitude to my supervisor Dr. Rizwan Ahmad for his help throughout my thesis. Besides my supervisor, I am also very thankful to GEC Committee members namely Dr. Adnan Khalid, Dr. Shehzad Saleem and Dr. Salman Abdul Ghafoor for their efforts, encouragement, and support to overcome numerous obstacles I had faced throughout my research.

In addition, I would also like to thank my fellow coursemates and all other friends for their virtuous support and cooperation.

Finally, I am grateful to all the individuals who have aided in this endeavour.

Table of Contents

List of Figures

List of Tables

List of Acronyms

IoT	Internet of Things
DoS	Denial of Service
DDoS	Distributed Denial of Service
SLA	Service level agreement
NIST	National Institute of Standards and Technology
RSA	Rivest, Shamir and Adleman
AES	Advanced Encryption Standard
DES	Data Encryption Standard
MAC	Message Authentication Code
SHA	Secure Hash Algorithm
ECC	Elliptic-curve cryptography
PIP	Policy information point
PDP	Policy decision point
PEP	Policy enforcement point
ABE	Attribute based encryption
MAC	Mandatory Access Control
EDoS	Economic Denial of Sustainability
HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
DNS	Domain Name System
SSL	Secure Sockets Layer
SMTPS	Simple Mail Transfer Protocol
DPI	Deep Packet Inspection
IDS	Intrusion Detection System
SYN	Synchronize
ACK	Acknowledge

Abstract

Over the years cloud computing users have increased drastically. In this escalated growth, smart devices have played a vital role. To coup up with this growth, researchers have introduced edge/fog computing at the edge of the network. Fog computing is not an alternative for cloud computing but an effective complement. Fog computing provides a lot of benefits for smart devices, however, it leads to a lot of security challenges as well. Access Control is one of the most fundamental and crucial security feature in deploying a secure system and is primary focus of this research. The dynamic nature of IoT devices combined with lack of proper security hardening leads to catastrophic results having huge impact. The DDoS attack launched on IoT devices by using Mirai malware lead to unavailability of services by major service providers namely Dyn, Twitter, the Guardian, Netflix, Reddit, CNN and many others in Europe and the US. In this research, we are proposing to use fog computing resources to provide authentication and access control. The devices will be authenticated by fog node and then connected to cloud. The fog node will monitor the behavior of connected devices and in case it observes malicious activity the connection between the device and node will be terminated. The experiments carried out to evaluate the proposed scheme shows promising results. We validated the proposed scheme based on NIST security criteria and by launching critical attacks from IoT devices. Additionally, in this research we have identified the security threats faced by a user in fog/ cloud enabled IoT devices environment. Next, we have proposed an access control model for this environment utilizing all the resources available. We have implemented this access control scheme where cloud resources are minimally utilized for the process of authentication and access control. The deployed Fog layer is used to provide these crucial security controls. From the results gathered in the performed experiments we have reached the conclusion that fog layer can be used to offload security solutions.

Key Words: *Cloud Computing, Fog Computing, Internet of Things, Authentication, Access Control, DDoS, Security Challenges, Open Issues.*

Chapter 1

Prologue

In this chapter, introduction to access control, internet of things (IoT) and fog computing is provided. Furthermore, security issues faced in these technologies and open issues/ limitation while implementing access control in IoT enabled cloud environment is discussed. The aim and scope of the research and methodology used to conduct the research is also highlighted. In the end the complete outline of the research is stated.

1.1 Introduction and Motivation

In a world filled with digital innovation billions of apps and devices are coming online. The aim of these apps/ devices is to measure, monitor, process, analyze and react to seemingly endless storm of data. Initially all this data was handled by cloud, however, now this data storm is also too much for the cloud to handle alone, as the distance between cloud and the end devices creates an issue for latency sensitive applications i.e. applications like disaster management and content delivery. Furthermore, in some cases the Service level agreements (SLAs) between the cloud service provider and customer might impose the location for data processing. However, the problem arises when the cloud service provider does not have data centers in the area. Fog computing is a breakthrough innovation to address such challenges. It allows the provisioning of resources and services at the edge of the network near the end user, outside the cloud eventually at a location imposed by the SLAs. It is a distributed architecture built to resolve latency, network inaccessibility, rising cost and sensitive data security concerns faced in cloud. It allows the distribution of core functions; compute, communication, control, storage and decision making closer to where data is originated.

1.1.1 Security of IoTs

The dynamic and scalable nature of IoT devices raise a lot of security challenges for the end user and cloud service provider. The IoT devices consist of smart devices, smart vehicles, wearable devices and different sensors over the internet. Cisco prediction of escalated increase in number of IoT devices is becoming a reality. Moreover, the number of attacks on these devices or attacks generated using these attacks have also increased dramatically. For example, in 2016 Mirai malware identified by white hat hacker group MalwareMustDie was one of the deadliest DDoS attack [31]. Mirai malware has been used to compromise multiple systems at different times e.g. security consultant Brian Krebs' website was hit by a DDoS attack of 620 Gbps traffic in September 2016, in parallel a French cloud service provider and webhost OVH was also targeted and DDoS attack was stronger peaking at 1.1 Tbps. The real chaos started after the public release of Mirai source code by its creator. Hackers started to sell/ rent out botnet of 400,000 interconnected smart devices. One of the major attacks by using Mirai malware happened on October 2016 when service provider Dyn was compromised for several hours taking down hundreds of websites with it namely Twitter, GitHub, Reddit and Netflix.

After the Mirai malwares source code was made public one expects that security research will develop efficient and effective detection and defense mechanisms. However, after 2 years multiple variants of the malware have emerged, the number of compromised devices has doubled but no effective solution has been deployed. Hackers are still using the same weakness to inject the malware into the IoT devices [31].

1.1.2 Security of Fog Computing

Fog computing is a brain child of security researchers at Cisco [10]. It is a paradigm that brings computing closer to the end user and offer services on behalf of the service provider at the edge of the network saving computational, network and storage resources. In figure 1.1 a three-layer cloud-fog-user hierarchy is shown. IoT devices connect with fog device which are interconnected and connected with cloud. Hence the data processing can occur at fog layer and send to the cloud which will reduce the computation burden on the cloud and reduce bandwidth latency, improving the overall user experience.

Fog computing will handle IoT device data which may include personal information. Hence, it is important to deploy effective privacy preserving and security procedures else without them fog computing will be useless regardless of its benefits. Fog devices can be manipulated by the attacker to launch man in the middle attack, also it can be target of classical security and privacy issues adopted from cloud.

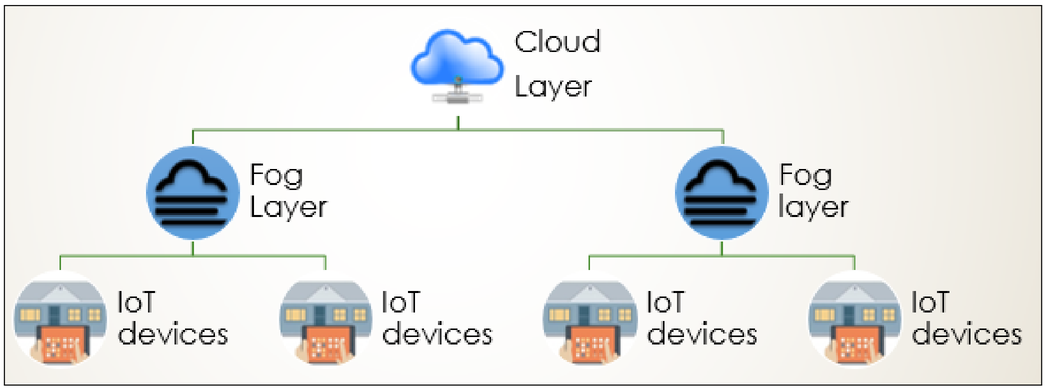


Figure 1.1: Three-layer cloud-fog-user hierarchy

1.1.3 Access Control

Access control is one of the important controls deployed to secure a system. It stops unauthorized users from getting inside the network and monitor the behavior of authorized users for malicious intent. Various regulations require privileged access and activity to be controlled and monitored. This includes implementing a system to monitor all use of privileged accounts, as well as monitor the activities performed by these accounts. Monitoring cloud resources for user access helps to:

1. Identify and prevent access violations
2. Quantify risk exposure and residual risk
3. Enforce segregation of duties
4. Have a role-based access control mechanism

While implementing access control for cloud computing the following features must be considered:

1. Control over elevated privileges
2. Reporting on access success and failures

3. Least privilege
4. Segregation of administrative vs. end user interface/access
5. Removal and archiving of identity information at the end of the lifecycle
6. Disabling of access for the identity at the end of the lifecycle
7. Real-time provisioning and de-provisioning
8. Dynamic trust propagation
9. Development of trusted relationships among service providers
10. User centricity
11. Violation reporting during provisioning
12. Enforcement of segregation of duties
13. Integration with Single Sign-On, dual mode authentication features
14. Provisioning to add multi-authentication layers for specific roles

While implementation of access control the security concerns include the points, where data could be decrypted, security of access to the data, separation of duties, and separation of logs when in multi-tenancy environments. Following concerns should be kept in mind while deploying an access control scheme.

1. Standards/Vendor Lock-in
2. Identity Theft
3. Unauthorized/Fraud & Accidental Access
4. Elevated Privilege Control
5. Non-Repudiation
6. Least Privilege & Excess Privileges
7. Performance/Availability
8. Features/Functionality Gaps
9. Disaster Recovery & Business Continuity Management

1.2 Aim of the work

This research addressed the security concerns of IoT devices connected with cloud. The first aspect of this research was to study existing access control schemes available for IoT devices connected with cloud thoroughly and identify their issues. A comparison of the existing access control scheme security and performance was done, for this purpose a list of ideal characteristics for this environment were identified and all the techniques were evaluated accordingly.

The second aspect of this research was to propose a valid access control mechanism for IoT devices connected with the cloud. This research did not only focus on how a device is authenticated but also provided a solution for the devices which are compromised after being authenticated, especially in the case when the hacker compromises the device to create a botnet.

1.3 Research Challenges/ Limitations

In this section, the security challenges faced while designing and implementing an access control framework is discussed.

1.3.1 Identifying the Compromised IoTs

In Cloud environment, the resources are shared which leads to an escalated risk of data theft by an insider. In this environment, the end user device must blindly trust the cloud service provider and the authentication mechanism in place to keep malicious entities out. Kolias et al. [31] has discussed Mirai and other botnet attacks by compromising the IoT devices at length. The author has elaborated how to create a botnet of IoT devices and execute a DDoS attack on a massive scale. Compromised IoT devices have been used in the past to hack different corporate organization and steal their user data (e.g. Twitter, U.S. President Barack Obamas account, Netflix, Reddit, and GitHub).

1.3.2 Dynamic Join and Leave of Fog Node

The fog layer itself is dynamic and fog node leaves and joins rapidly. Due to this reason below security challenges are faced [38];

- Whenever a fog node joins or leaves fog layer how the security and privacy issue will be tackled? Example of such a scenario is how a IoT device will be authenticated when it joins the fog node and how privacy of data will be preserved after the IoT device leaves.

- How to ensure that the authentication scheme implemented will allow scalability in the fog network?
- Furthermore, how the identity of the user is protected and how will it be traced in case of a malicious activity detected?

1.3.3 Context Aware Access Control between IoTs and Fog Nodes

In IoT devices the data is updated and send for processing very rapidly. The access control model should be efficient and effective enough to manage the access control request and capture, transfer, process and store the contextual data to make necessary decision [38].

1.3.4 Resources Restriction

The access control process must be power and resource aware since IoT devices have limited of both [38].

1.3.5 Network Availability

While designing an access control scheme keep in mind that IoT devices also function in area with network unavailability. So the authentication and authorization process should also work offline [38].

1.3.6 Privacy Violation

A data privacy clause between the IoT devices and fog nodes should be established since fog computing is a decentralized technology and data is shared among different nodes [38].

1.3.7 Decision Latency

The communication lag between the IoT devices and fog/ cloud should be minimum since the delay can cause serious issues to infrastructure or even human safety [38].

1.3.8 Accountability

Administration accounts activity should be monitored for any malicious activity [38].

1.3.9 Multi-Tenancy and Virtualization

In fog computing as well as in cloud computing, virtualization plays a key role in creating an isolated environment. By implementing a shared resource environment among untrusted nodes, the risk of side channel attack is escalated. Furthermore, the information flow in a multi-tenancy environment may lead to unauthorized access.

A hypervisor can be used to discourage unwanted traffic before it enters the network as it is scalable and programmable. Furthermore, it is network-independent. Popa et al. proposes an access control scheme with multi-tenancy which claims to be more scalable and robust than network based approaches [41]. Hypervisor can be used to implement solutions based on access control policies, like all policies and mapping is deployed at the hypervisor so every flow that enters the network is routed accordingly. However, this approach is not scalable. A better approach is to create a central repository of the policies and group membership. Hypervisor will cache and access this repository for each new flow. The drawback of this scheme is that the repository should be responsive and available at all time. Furthermore, it can be an easy target for denial of service attack.

From all the security challenges enumerated above we have reached the conclusion that a well-developed access control mechanism can cater for most of the issues.

1.4 Research Contribution

1. We have conducted a detailed survey on existing access control technique in IoT devices connected to cloud/fog computing. This research also included identifying ideal characteristics of an access control scheme in this environment and performing a detailed analysis of state of the art access control techniques on this criteria. In the end, challenges and limitations faced by security researchers in developing an ideal access control mechanism are discussed.

2. We have proposed and implemented an access control scheme using

Raspberry Pi to conduct the desired experimentation. The aim was to identify compromised devices before they interact with cloud and offload the computation overhead of access control from cloud. This included implementation of different security mechanisms within this environment including secure authentication, intrusion detection system and deep packet inspection. The security mechanism implemented were tested against denial of service attacks. We have offloaded the access control mechanism computational and security concerns from the cloud and handed it over to the fog layer.

1.5 Research Methodology

1.5.1 Research Approach

This research work is divided into two main parts. First part is based on survey of current state of the art access control techniques proposed or implemented for IoT devices connected to cloud/fog. In this research, we have tried to understand the compatibility of a conventional access control scheme in this dynamic environment. We highlighted the challenges and limitation faced by security researchers. Also a detailed performance analysis of the existing access control technique is done.

In the second part of this research we proposed an access control scheme using fog computing and offloading the computation overload from the cloud. Furthermore, our research focuses on how the environment will handle a compromised entity. To prove the stability of the proposed scheme we launched denial of service attacks on it.

Design Science and Our Research

Our research falls in problem solving domain because in our case we have to suggest an access control scheme that can provide a solution against DDoS attacks. The research aiming to solve problem that can be studied through by Design science research approach [40], March and Smith [34] says that natural science and design science are different as natural science is about understanding the reality and the latter is about creating it. Also according to Simon [51], it is described that the main foundation of design science lies on engineering science and sciences of artificial. Design science products usually serve human purposes. The products are evaluated for their performance, value, utility and improvement [34]. Innovation is the main objective to design science that gives birth to new products, ideas and practices. Moreover, it serves human in a more efficient ways [13] . Also, according to Aken's design science, the main objective of some innovation is to solve construction and improvement problems [3].

Activities and Output

Designing an access control scheme and evaluation of the existing techniques were the main activities of this research and the outputs were the scheme and method. It is important to explore this area in the domain of security as access control is one of the crucial defense mechanism implemented. To develop these schemes and methods, we explored the area to obtain knowledge about the traditional access control schemes and how they are obsolete in this dynamic and scalable environment.

1.5.2 Research Process

This section maps our research process to the process of Design science research. Design science research contain five sub processes [55]. These are:

1. Awareness of problem
2. Suggestion
3. Development
4. Evaluation
5. Conclusion

Vaishnavi and Kuechler [32] also differentiate five different phases based on design science research approach, as shown in figures 1.2 and 1.3.

Awareness of the Problem

During this phase, familiarization with the problem as well as with associated domain is acquired. Design science research; demand the awareness of problem to suggest a solution to it [32]. In this phase, we have explored and evaluated the existing access control techniques and schemes in the paradigm of traditional environment and virtualized environment to become familiar

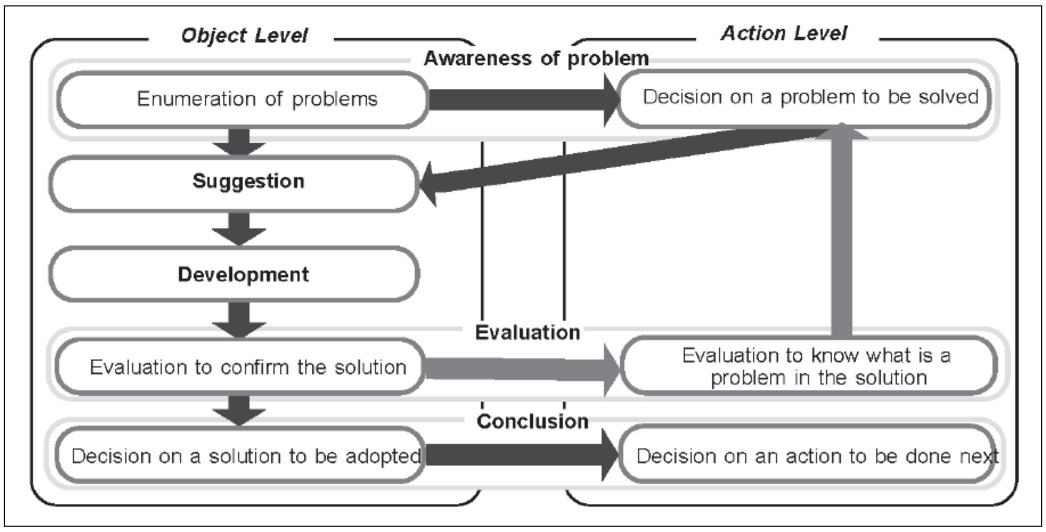

Object Level
Action Level
Awareness of problem
Enumeration of problems
Decision on a problem to be solved
Suggestion
Development
Evaluation
Evaluation to confirm the solution
Evaluation to know what is a problem in the solution
Conclusion
Decision on a solution to be adopted
Decision on an action to be done next

Figure 1.2: Design Cycle [55]

as well as to recognise the problem associated with them. In this phase, we have:

1. Explore and evaluate access control techniques for cloud/fog enabled IoT devices.
2. Explore access control techniques threats to identify the problem.
3. Explore threats which can exploit the vulnerabilities of this environment.

Suggestion

After the awareness of problem, we proposed the solution for the problem that was identified in first phase. Awareness of problem enabled us to be familiar with the domain, theoretical basis, and also with the techniques used for access control. In this phase, we suggested an access control scheme for the research community.

Development

In the light of suggestion phase, an access control is implemented [32]. In this phase, we developed specifications to design the scheme by adopting the existing tools and techniques available as open source and modified them according to our requirements.

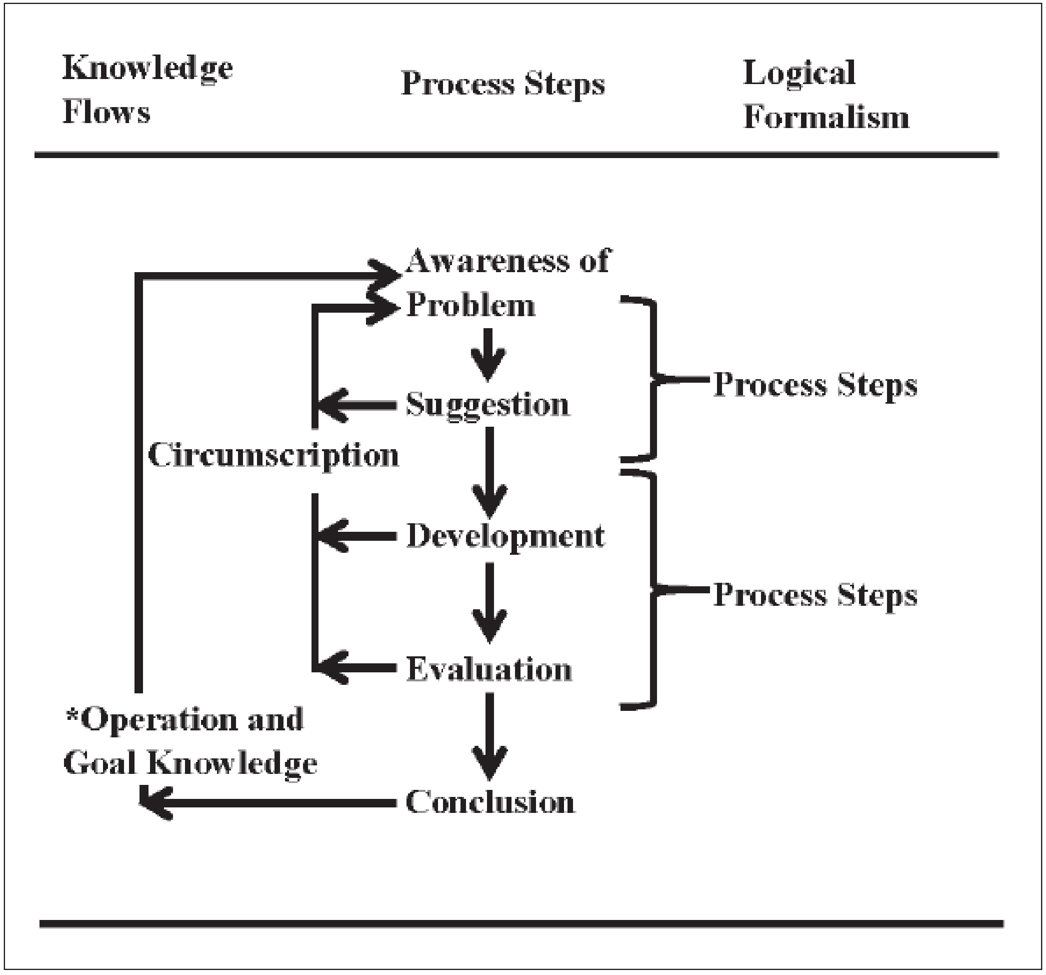


Figure 1.3: Reasoning in the Design Cycle [32]

Evaluation

After developing the access control scheme, it is evaluated for different criteria of performance and improvement. Any deviations from expectations, both quantitative and qualitative are carefully noted. A new cycle is vital, if a problem is catered during the phase of evaluation. Development, evaluation and suggestion phases are frequently performed over and over again in a classic design research approach [32].

The proposed solution was evaluated and it was established that, an intrusion detection system needs to be introduced in the existing system. On the positive side, the solution provides strong authentication and prevention from malwares, insider and DDoS attacks.

Conclusion

It helps us to offer a solution or to modify the description and mark the end of this project. A specific solution was suggested in the form of a scheme. Additionally, our scheme provides a new avenue for the research community to control the attacks that are being launched at a massive level as the number of IoT devices is increasing dynamically.

1.6 Outline of the Work

The thesis document is organized into seven different chapters. Each chapter explains a particular aspect of research.

Chapter 1, titled **Prologue**, includes the introduction of internet of things, fog computing and access control from security perspective. It explains the problem statement and the motivation behind the research. The basic objective of the research is defined followed by details about the research contribution and the methodology used to carry out this research. This chapter describes, briefly, the aim and scope of the thesis with its objectives and limitations of the research.

Chapter 2, titled **Threats in Fog/Cloud enabled IoT Devices**, highlights the security threats to cloud/fog enabled IoT devices. The vulnerabilities existing in each layer of this environment are also identified.

Chapter 3, titled **Literature Review**, provides a detailed survey of the existing schemes which are used for access control in Cloud/fog enabled IoT environment, followed by a detail discussion on different authentication schemes that can be used for authentication in this environment. At the end of this chapter a detailed performance analysis of existing access control techniques has been done.

Chapter 4, titled **Proposed Solution**, describes in detail, the solution proposed by the author, in order to propose a more secure access control mechanism. This chapter also explains the architecture of the proposed solution.

Chapter 5, titled **Implementation**, provides the technical details about the implementation of the proposed scheme, minimum requirements of hardware and the software required to recreate this scheme. Furthermore, installation details of deployed cloud, deployed fog nodes, its environment, and all the needed components to execute the framework are provided. The next section explains the setup of the whole environment and how all the devices are connected.

Chapter 6, with the title of **Validity of Proposed Research**, discusses the results from the proposed access control scheme in detail. We validated the proposed scheme based on NIST security criteria and by launching critical attacks from IoT devices.

Chapter 7, with the title of **Conclusions**, details the conclusion and highlights the future research directions.

Chapter 2

Threats in Fog/ Cloud enabled IoT Devices

In this chapter, security threats to cloud/fog enabled IoT devices are highlighted and then it is identified/ mapped how each layer is vulnerable to which threat.

The Cisco Fog paradigm can include, thrive and influence several enhanced features such as rapid analysis, interoperability of devices, increase in response time, centralized or machine-to-machine management, low bandwidth utilization, efficient power utilization, device abstraction and many others. It can be an enabler of many advanced technologies. Different approaches similar to fog computing are being used to utilize the full potential of cloud computing [49]. As these technologies i.e. fog computing, edge computing, micro data centers and cloudlets are being widely deployed they are also becoming a target to different cyber-attacks [62]. Cloud Security Alliance [1] and other researchers [52] [53] [59] have identified twelve critical issues which directly effect on-demand, shared and distributed nature of cloud. Fog computing is a virtualized environment like cloud computing, it is affected by the same threats. The twelve security threat categories can be classified into three domains i.e. on users end, virtualized environment and database. In figure 2.1, threats have been identified and classified based on the deployment architecture [30]. The highlighted threats have been discussed individually on the basis of their deployment.

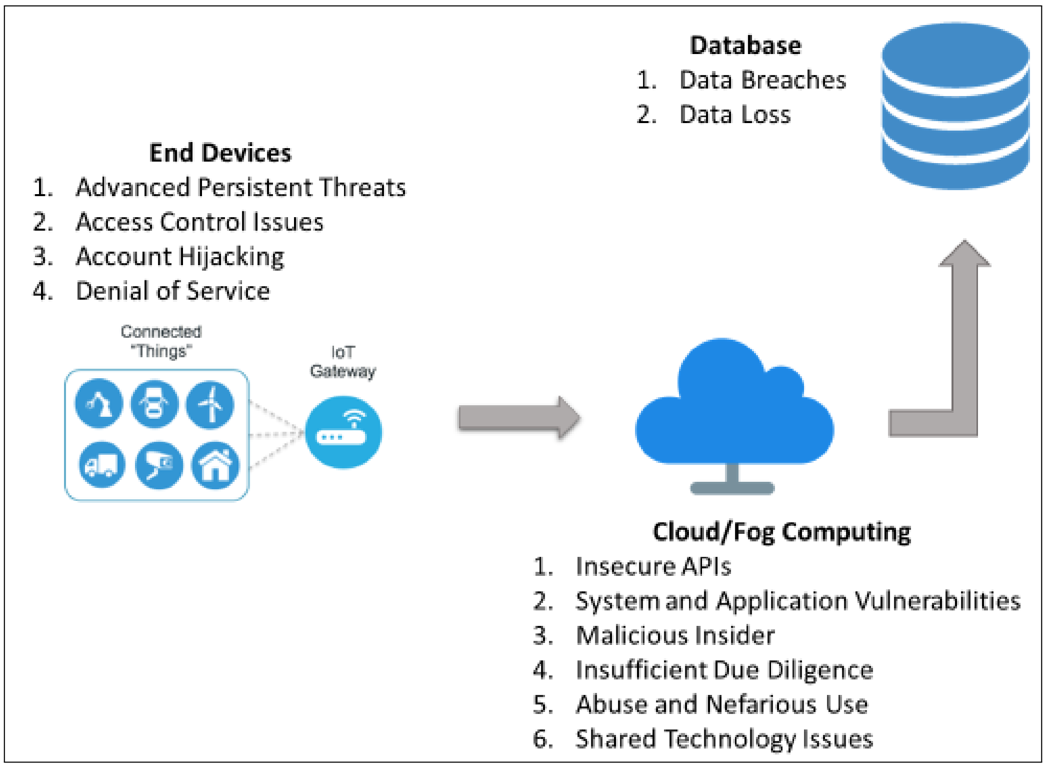


Figure 2.1: Potential Security issues in Fog computing

2.1 On User's End

During deployment end user devices and network security controls are generally not considered which afterward leads to security issues. The attacker can exploit network and end device vulnerabilities. The most common and hazardous threats in cloud/ fog computing is account/ session hijacking and denial of service attack.

2.1.1 Advanced Persistent Threats

Advanced persistent threats are cyber-attacks which are targeted attacks planned and carried out by skilled individuals to compromise or take down an organization. The attacker motivation can be to steal data, intellectual property or defacement of the company [31].

2.1.2 Access Control Issues

Access control issues are the result of poor management or bugs/ vulnerabilities in the software. Access control issues can lead to unauthorized user getting access to sensitive data or permission to administrative rights [61].

2.1.3 Account Hijacking

Account/ Session hijacking is a network level issue in cloud/fog computing. In account hijacking the attacker targets a specific account and steals the identity and personal information of the user. In session hijacking the attacker tries to take control of a specific session. Multifactor authentication can be used as a precautionary measure [43]. A strong and sophisticated identity management mechanism should be deployed. Over the years researchers have proposed different preventative and detective techniques i.e. Network monitoring [45], Data Leakage Prevention Technology [12], Vulnerability Detection Technology [12]. In fog computing the approach proposed by researchers is to combine both data recovery and decoy technique to overcome this threat [29] [6].

2.1.4 Denial of Service

During denial of service attack legitimate users cannot access the required services. To prevent and detect this attack Intrusion detection system (IDS) is the most deployed control [12]. In Cloud computing to prevent this attack IDS is installed in the virtual machine (VM). Whenever the IDS detects an unusual high inbound traffic it migrates the service to an alternate data server [31] [14].

2.2 Cloud/ Fog Computing Layer

The primary focus of attackers is the virtualized environment (cloud/fog). Maintenance of cloud environments security is the sole responsibility of the cloud service provider. Similarly, fog nodes security will be the responsibility of the service provider.

2.2.1 Insecure APIs

Cloud service providers give access of their APIs to different customers. The security of these APIs is pivotal as they are used in the implementation of cloud. If the third party/ customer finds a vulnerability and exploit it, they can get access to sensitive data and privileged accounts. While giving APIs to customer the cloud service provider needs to ensure a solid authentication and access control mechanism to avoid exploitation [27]. In fog computing for further ensuring the security cryptographic hash functions can be used.

2.2.2 System and Application Vulnerabilities

System and application vulnerabilities are exploitable bugs due to misconfiguration or lack of security aware development. Furthermore, an attacker can also use configuration errors in software ad to penetrate and exploit a system [62].

2.2.3 Malicious Insider

Malicious insider is also known as insider threat. The user has authorized access to the system but abuses its privilege. The motivation of the employee is to get consumers sensitive information. The prevention of this threat is strong access control mechanism and continuous logging [27].

2.2.4 Insufficient Due Diligence

When shifting to cloud services the user/ customer lack of knowledge related to security i.e. logging, auditing, data access, data custodian can lead to multiple threats in cloud. In certain cases, the designers and developers might not be aware of the issue relating to application deployment on cloud which can lead to operational, architectural and security issues [53].

2.2.5 Abuse and Nefarious Use

Many of the cloud users dont have the required knowledge to use the cloud services. In such cases there is a strong chance that the user might exploit the cloud services and violate their contract with the cloud service provider. To avoid such scenarios the cloud user needs to be educated about the basics of cloud service usage. In case of an organization the employees should be explicitly explained the important policies in the service level agreement (SLA). The users can utilize the resources for malicious activity [54].

2.2.6 Shared Technology Issues

Cloud/ fog services are provided by hosting the services on a single environment and perfect isolation. Sharing of platforms, infrastructure or applications without proper isolation may lead to computation issues and provide an opportunity for the malicious user [12].

2.3 Database

Data security is an issue in cloud computing from the start. Data threats exploits the vulnerabilities in cloud and target the sensitive information present on cloud. Data loss or leakage can gravely affect the cloud service provider as well as the organizations hosting the information.

2.3.1 Data Breaches

In data breaches confidential, sensitive hosted data on cloud is stolen or tampered by the attacker. This threat affects the consumer as well the cloud service provider. One of the proposed techniques to prevent data breaches is encryption of the data [47] [21]. Multifactor authentication is one of the more effective solutions proposed [43]. In Fog computing to prevent data breaches use of decoy techniques is proposed [29] [57].

2.3.2 Data Loss

Data loss is unintentional or malicious deletion, corruption of information. It is not necessarily caused by an attacker; a technical issues or human error can also result in data loss or leakage [62].

2.4 Conclusion

Access control is an issue identified at the user's end. However, if no proper access control mechanism is in place it can lead to other attacks as well. If authentication protocol is compromised the attacker gets access to the users information. It can exploit the vulnerabilities and misconfigurations of the underlying technology to escalate his/her access. Moreover, it can use cloud resources to launch advance persistent threats. Additionally, it can get access to database and delete or modify the existing information. Access control plays a key role in stopping different threats faced in a virtualized environment, including but not limited to malicious insider, denial of service, abusive user, botnet and zero-day vulnerability exploitation. Access control access mechanism helps in upholding the confidentiality, integrity and availability security triad. In figure 2.2 potential security threats related to implementation of fog application have been highlighted. The threats have been divided into six categories namely; virtualization, web security, internal/ external communication, data security, wireless security and malware protection.

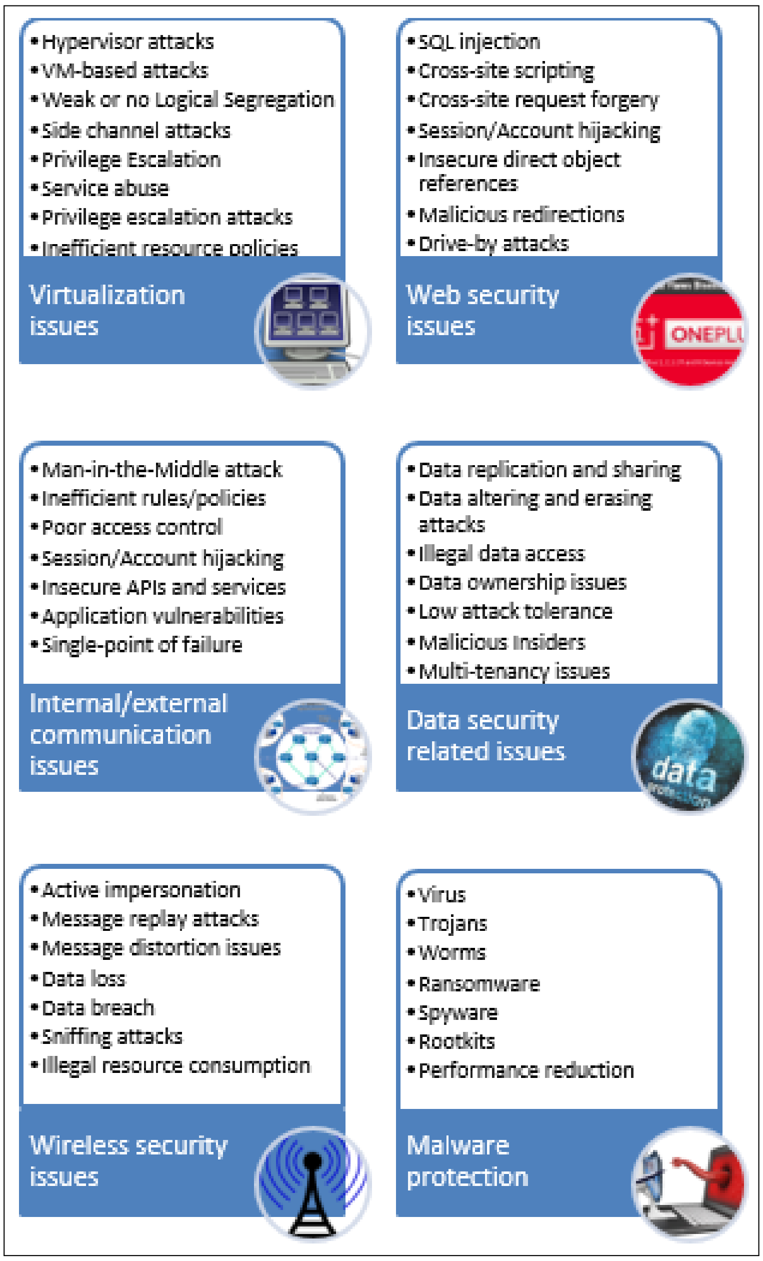


Figure 2.2: List of security threats found in Fog nodes

Chapter 3

Literature Review

In this chapter, the state of the art solutions provided by different researchers are highlighted and discussed. Additionally, a theoretical comparison analysis has been performed of the techniques.

3.1 State of Art Access Control Approaches

One of the main security concern in fog/cloud computing identified is end user (IoTs) authentication to fog node & cloud. Researchers have suggested different techniques to mitigate data tampering and spoofing attacks, namely Diffie-Hellman key exchange, Intrusion detection techniques, monitoring for modified input values and Public Key Infrastructure (PKI). In the authors have demonstrated the impact and high importance of man in the middle attack on fog computing. They compromised a video call between WLAN and 3G user in fog network and in the results, they observed minimal changed in memory and CPU consumption of fog node. Hence the detection and mitigation of man in the middle attack is difficult and the authors recommend for prevention of such attacks to devise a secure authentication protocol between the fog node and the user.

Over the years researchers have proposed different authentication schemes for Fog enabled IoTs. Hu et al. has proposed three solutions for authentication scheme, data integrity and data encryption by using face identification and resolution technology [23]. To provide confidentiality, integrity and availability to Fog enabled IoT devices the researchers have proposed three countermeasures namely; Secure Hash Algorithm-1 (SHA-512), Advanced Encryption Standard (AES) and authentication on session key agreement.

Different open source frameworks like fog computing are available in the market to make interoperability simpler between cloud and user. OPENi framework [42] is one such example of a personal cloudlet where the virtual space securely stores data and give user access to control data. To enforce access control in this framework an access control list is maintained by the cloudlet owner in NoSQL database and OAuth 2.0 is used to set permissions for every resource.

Similarly VORTEX is a data sharing platform developed by Prismtech for IoT devices [2]. This platform enables the IoT devices, fog nodes, web applications and cloud to share data efficiently and in a seamless manner. In VORTEX multiple privacy and security challenges have been addressed, both symmetric and asymmetric authentication are compatible and it supports fine grained access control.

Dsouza et al. [16] proposed a policy management framework to ensure secure sharing and communication with fog computing. The designed solution is a preliminary framework and vital information like how the policy repository will be build, user identification, decision making and how identity & data privacy will be maintained are not discussed.

Distributed Reference monitoring access control model for fog computing is proposed by Salonikias et al. [48]. In this model, the security policy and attributes are maintained in a distributes policy information point (PIP). Policy decision point (PDP) is carrying out access control evaluation on fog nodes and enforcement point of access control decision policy enforcement point (PEP) are deployed on the edge of the network. Furthermore, a policy propagation control method is also proposed to synchronize the policies in PDP when they are updated in PIP.

Stojmenovic et al. highlights the authentication and authorization issues among fog devices and between cloud and fog devices [53]. The authors suggest that by implementing cryptographic schemes like attribute based encryption the IoT devices can get access to fog devices irrespective how delicate the link between fog and cloud is.

Li et al. has proposed an attribute based encryption (ABE) scheme; using the location, Wi-Fi network, unlock failure and similar attributes generated from smart devices and referring to them as dynamic attributes [33]. Using these dynamic attributes in ABE scheme the authors have proposed a real-time access control system.

Zaghdoudi et al. has proposed a robust and scalable access control scheme specifically for fog computing and ad hoc mobile cloud computing [60]. The proposed scheme uses distributed hash tables for access control model.

Mollah et al. suggested to offload all security related operations to an edge network which is present one hop away from the end device [37]. At the edge device lightweight cryptographic scheme will be implemented for access control and data sharing.

Moghaddam et al. suggested an efficient and scalable authentication scheme for cloud computing. It proposes a client based user authentication scheme to confirm the identity of the user [36]. The service provider installs a plugin on the web browser of the user with a unique code, efficiently, every time identity of the user is confirmed from the plugin instead of the service providers server. To connect unregistered devices to cloud server modified Diffie Hellman is implemented [36] [11] [28]. For registration of new devices a cloud based application is used [44]. Authentication and cryptographic resources are stored on a separate server and the main server is relived from the computation burden. This process enhances the reliability and trust between cloud and user. It provides security to the cloud environment from Man in Middle attack, Brute Force Attack and Timing Attack.

Auxilia et al. has proposed a dynamic access control after discussing various access control schemes [8] . The dynamic access control model proposes to implement a link between data requestor, requested and action taken. Authors have performed a theoretical comparison between the various schemes discussed and proposed model.

Niu et al. proposed an effective and secure access control model [39]. The solution is based on attribute based encryption for lightweight devices like IoTs communicating with a cloud environment. In this access control scheme, the computation burden is shifted to cloud environment from end user.

Kanna et al. proposed a scheme for outsourced cloud data by hybrid cryptographic technique and keyword encryption [28]. The scheme uses an identity based hybrid encryption method. In this scheme, the data is encrypted before storing on cloud.

Rehman et al. proposed implementing a passphrase based authentication scheme over the cloud [44]. Authors have performed a comparative analysis of multifactor authentication frameworks in cloud environment. In this scheme, asymmetric cryptography is deployed, private key is generated from a random passphrase. Furthermore, a onetime access to cloud environment is pushed to users mobile device. The proposed scheme discusses a one-time based authentication system which will majorly drain the resources of cloud.

Atlam et al. has proposed an adaptive and dynamic risk based access control model [7]. The scheme is designed specifically for IoT devices keeping in mind the dynamic nature and low computation of the end user devices. They have evaluated the access control scheme in real time. Each access request is evaluated on device context, action severity, resource sensitivity and risk history. This scheme is in its preliminary stages and still requires an appropriate risk estimation technique.

Access control schemes discussed above are based on one of the three techniques i.e. access control list, cryptographic controls or policy-based framework. Each technique has its own pros and cons, however, discussing the cons from our specific identified issue point of view they individually fail to provide a complete coherent solution. In the case of a compromised IoT device environment due to default credentials used and other unpatched vulnerabilities these solutions must be molded accordingly. Cloud based techniques discussed above once authenticated and give access to IoT devices do not check for malicious behavior again. Now a days hackers are exploiting daily use smart devices to get access into the network and exploit it. These schemes fail to address this issue.

After reviewing multiple existing access control models and frameworks in cloud and fog environment one can conclude that there is a lot of effort still required to tackle the security challenges in this domain. Wadhwa et al. [58] has also concluded in their study that a user specific access control framework is required.

3.2 Different Authentication Schemes Used to Authenticate IoT Devices

In this subsection, we analyze and generalize different types of authentication. Authentication is one of the main security issues in IoT devices. IoT devices are interconnected to perform their specified tasks. If no security protocols are implemented to secure the network the data transmitted will be vulnerable to multiple attacks. Security credentials transmitted for authentication purposes over the network channel can be captured and used for launching man in the middle attack or impersonation attack. Each IoT device has a unique identifier and if compromised can be used by the attacker in malicious activities.

IoT devices network consist of billions of devices which needs to be authenticated simultaneously with the authentication server and in some cases with eachother. IoT devices have limited resources e.g. power, memory and computation so traditional cryptographic based methods can not be deployed. The world is becoming a global village and IoT device data is directly transmitted to cloud/ fog. From IoT device's perspective all the communication between device and cloud/fog/online server needs to be authenticated. Authentication is the backbone of a secure network and ensures safe data communication and browsing experience.

Cryptographic protocols are divided into two domains symmetric cryptography and asymmetric cryptography. Cryptographic protocols uses one or more of the below mentioned aspects to create a secure authentication protocol;

1. Key agreement or establishment
2. Entity authentication
3. Symmetric encryption
4. Message authentication
5. Non-repudiation methods
6. Secret sharing methods
7. Secure multi-party computation

3.2.1 Lightweight Symmetric Cryptographic Primitives

Implementation of a symmetric cryptographic based authentication protocol is preferred in a constraint network like IoT as it requires less computational power and time. Furthermore, no mathematical operations are required to be performed which reduces the computational overhead and allows the process to take place in few milliseconds on restricted processing power of less than 1 Kb RAM [20]. In symmetric cryptographic based authentication protocol a key is shared among the authentication parties.

Symmetric cryptographic algorithms

Symmetric cryptographic algorithms include Vigenere Cipher, Blowfish, Twofish, RC4, Advance Encryption Standard algorithm (AES) and Data Encryption Standard (DES). In symmetric cryptography authentication the two entities encrypt the authentication required information by the key shared in advance among them. Some authentication protocols implemented use Message Authentication Code (MAC) to provide authentication and integrity to a message. MAC is basically an encrypted checksum which is generated using the message sent to authenticate the message.

In alot of recent IoT authentication schemes being developed symmetric cryptography method is used [15] [4] [56] [17] [5]. It is more secure, fast and less energy consuming, however at the same time key management is big issue that needs to be resolved i.e. initially how the keys will be shared between the entities? Does the device has enough memory to store the key? In case of cloud/fog how will they share this information? Is creating a central repository to manage the keys secure?

3.2.2 Lightweight Asymmetric Cryptographic Primitives

In public key algorithms, traditional implementation scheme key management is flexible, however, a significant chunk of memory and power is consumed by modular multiplication and squaring of large integers. To reduce the constraint on network and device resources number of handshakes and size of the message sent during authentication should be minimum. When X.509 certificates are used in conventional form over a low rate communication channel, specifically for long messages greater latency and more airtime is utilized during the authentication process [22].

Asymmetric cryptographic based authentication can be further divided into three categories;

1. Public key algorithms
2. Digital signatures

3. Digital certificates

Public key algorithms

In literature multiple public key algorithm schemes to achieve secure authentication are discussed. The most commonly used public key algorithms used for authentication are Elliptic-curve cryptography (ECC) and RSA (Rivest, Shamir and Adleman) asymmetric algorithm. ECC is better than RSA as it offers faster computation, smaller key size, less memory, bandwidth and energy consumption [18].

RSA SecurID is a two-factor authentication protocol to protect the network. The two factors consist of something you know (password) and something you have (authenticator/token). The authenticator can be a hardware (smart card, USB, RFID etc.) or a software token (code received by email or SMS). The basic working of the protocol is when a requester attempts to access a protected resource, he is prompted for a unique passcode. The passcode is generated from the user's password and the authenticator token generated at the time of the request. In figure 3.1 the authentication mechanism is shown [19]. The requestor is sending an authentication request to the authentication manager with passcode and token code.

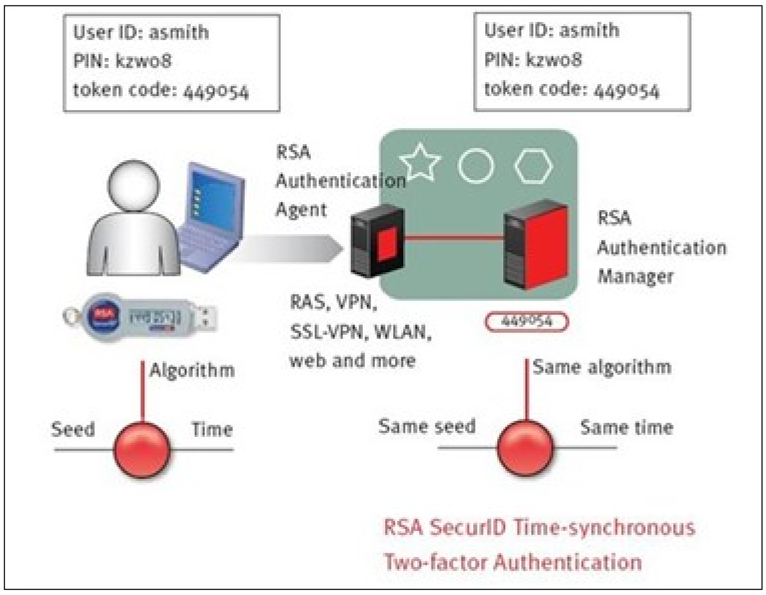


Figure 3.1: RSA Authentication [19]

Digital signatures

Digital signature is similar to MAC i.e. it also appends an authentication token with the message. However, the difference between the two is that digital signature uses a pair of public and private key. The keys are used to generate the signature which is used as authentication token and verify it. Mostly hash function is used to generate digital signature. To generate a digital signature two steps are followed. In the fist step authentication message is hashed and in the next step message digest (hashed value) is signed by the sender's private key. The signed output is attached to the message and forwarded for authentication shown in figure 3.2.

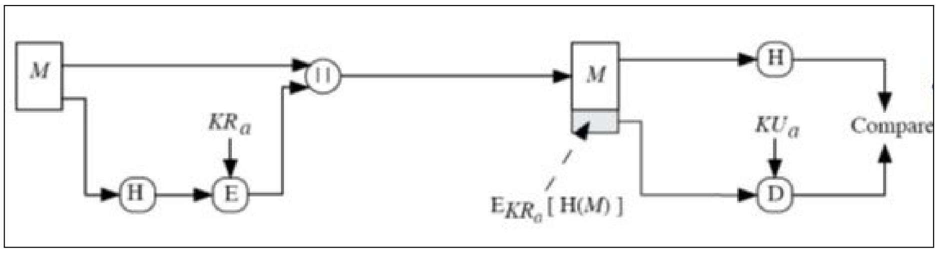


Figure 3.2: Digital signature

Digital certificates

Sciancalepore et. al. has proposed an authentication and key agreement protocol using PKI for IoT devices that doesn't use any explicit signature [50]. The researchers have used elliptic curve Diffie-Hellman and elliptic curve Qu-Vanstone for key agreement and implicit certificate respectively. Experiments conducted shows that the combination of these two protocol is far better than the conventional schemes relying on X.509 certificates.

3.2.3 Conclusion

From the literature review conducted, about different cryptographic authentication schemes being implemented to authenticate IoT devices we have reached the conclusion that RSA SecurID is the most well-suited option for our scenario. It is a lightweight algorithm which require less computational resources. Furthermore, no high-priced infrastructure is required for the deployment of this cryptographic scheme.

3.3 Performance Analysis of Existing Access Control Techniques

Different approaches for fog/ cloud enabled IoT have been discussed in the above section, each one addressing somewhat unique research gaps. However, each technique has its own strengths and limitations as well. In Table 3.1 we have identified nine requirements based on which the techniques surveyed in this paper are evaluated. The analysis of these techniques is conducted on the virtual environment the access model is deployed in i.e. cloud or fog, whether authentication mechanism is proposed or not, what kind of access control technique is used to devise the solution, can it detect insider attacks, is it scalable and resource utilization i.e. resources of which layer will be consumed (cloud, edge or end user).

While conducting the survey of different access control models it was noted that almost every scheme was based on one of the three techniques i.e. access control list, cryptographic controls or policy-based framework. Our proposed access control scheme is based on behavioral analysis i.e. once the device is authenticated by the fog/ edge layer its behavior will be monitored for anomaly. If irregular or suspicious behavior of a device is detected the fog layer will isolate it and run a full scan. This scheme will be able to detect insiders launched attack and stop the attacker from making a botnet.

Some of the access control schemes surveyed uses the fog layer resources for authenticating and access control to offload the cloud layer and end user devices of the computation overhead. However, in all the literature surveyed none of the solutions cater of devices which are compromised after that the device has been authenticated and authorized. Once the device has joined the network they are considered legitimate. This negligence by security researchers is the reason Mirai attack and its variant are so successful.

To perform the comparative analysis of the existing access control schemes we thoroughly reviewed and understood the techniques the authors have proposed and implemented. The criteria defined for the analysis is whether the solution:

1. caters for the fog/edge layer i.e. the implemented scheme uses fog layer or considers its implications in the network.
2. considers the role cloud computing will play in the scheme and how it will be affected due to the scheme.

3. proposes an authentication scheme and is using it solely to maintain access control.
4. proposes an access control mechanism.
5. handles the insider threat. Since one of the greatest threat in cloud computing is malicious insider and misuse/ escalation of privileges.
6. considers the scalability of the network. As the connected devices increase or decrease the implemented resources should scale up and down accordingly utilizing them optimally.
7. utilizes the resources of cloud computing for computational activities.
8. utilizes the resources of fog/ edge computing for computational activities.
9. utilizes the resources of end user devices for computational activities.

We have evaluated the techniques on the basis that, if the reviewed solution:

1. has discussed, implemented or given satisfactory reasoning about the criteria it is marked as ‘Y’.
2. provides no sufficient evidence related to the criteria we have marked it is as ‘N’.
3. is described such that the criteria is not applicable i.e. due to lack of information provided or no sufficient evidence given to prove the claim, the field is left unmarked.

After going over different techniques used to implement and monitor access control we have reached the conclusion that incase of IoT devices connected to cloud the best approach used is restrictive access control. The most restrictive access control technique is Mandatory Access Control (MAC) in which a system administrator gives access to the user and individual resource owner cannot give access to their objects or processes. In our case on the basis of behavioral analysis if the device is behaving oddly the fog node will terminate its connection to the cloud and run a full scan on the suspicious device. In a restrictive access environment, the service provider has control over the individuals.

Table 3.1: Performance analysis of existing technique

Research Work	Fog/ Edge Computing	Cloud Computing	Authentication	Access Control	Resist Insider Attack	Scalable	Restrain on Cloud Resources	Restrain on Fog/ Edge Resources	Restrain on End User Resources
[23]	Y		Y	N	Y	Y	N	Y	N
[42]	Y		Y	Access Control List	N	Y	N	Y	N
[2]	Y		Y	Fine grained	N	Y	N	Y	N
[16]	Y			Policy-based	N	N	N	Y	Y
[48]	Y		Y	Policy-based	N	N	N	Y	Y
[53]	Y		Y	N	Y	N	N	Y	N
[33]	Y	Y	Y	Attribute Based Encryption	Y	N	Y	Y	Y
[60]	Y		Y	Distributed hash tables	N	Y	N	Y	N
[37]	Y			Encryption	Y	N	N	Y	Y
[36]	N	Y	Y	N	Y	N	Y	N	N
[8]	N	Y		Dynamic Access	N	Y	Y	N	N
[44]	N	Y	Y	N	N	Y	Y	N	N
[39]	N	Y	N	Attribute Based Encryption	Y	N	N	N	Y
[7]	N	Y	Y	Risk based	Y	N	Y	N	N
[58]	N	Y	Y	Multi-Security Level	Y	Y	Y	N	N
Our Research	Y		Y	Behavioral based	Y	Y	N	Y	N

Chapter 4

Proposed Solution

In this chapter, an access control scheme is being proposed for Cloud based IoT devices.

4.1 Introduction

In the proposed scheme authentication and access control will be off loaded from Cloud environment as well as IoT devices. Fog computing will be used to authenticate and manage the trusted devices. For authentication asymmetric scheme is used keeping in mind the dynamic nature of IoT devices.

As explained in section 3.3 we are using restrictive access control scheme to enforce access control mechanism. Baig et al. has implemented restrictive access in cloud environment mitigating Economic Denial of Sustainability (EDoS) attacks [9]. Restrictive access techniques are basically admission control methods to take preventive action against the service capacity. This strategy implements the prevention by delaying responses/access to the suspected attackers or even additional clients. This delay is introduced by prioritizing the legitimate clients or selecting clients with good past behaviors. Similar techniques like Delayed access and Selective access exist, however, the method to provide access to end users differ.

4.2 Architecture Diagram

Figure 4.1 and 4.2 shows the architecture of the scheme proposed for secure authentication and access control respectively. In figure 4.1 IoT devices are being authenticated by Fog nodes and Fog nodes after establishing a connection with cloud, connects them with cloud. In figure 4.2 we have depicted the scenario in which an IoT device behaves suspiciously. The first step the fog node takes is to terminate the cloud and IoT device connection. After that it runs a scan on the untrustworthy device. If the device is malicious the fog node will send the traffic from the compromised device to the controller node (fog node with deep packet inspection tool) for deep packet inspection. Once the controller node identifies the malicious application compromising the device it will send the new ruleset/ policy to all the other fog nodes in the fog layer.

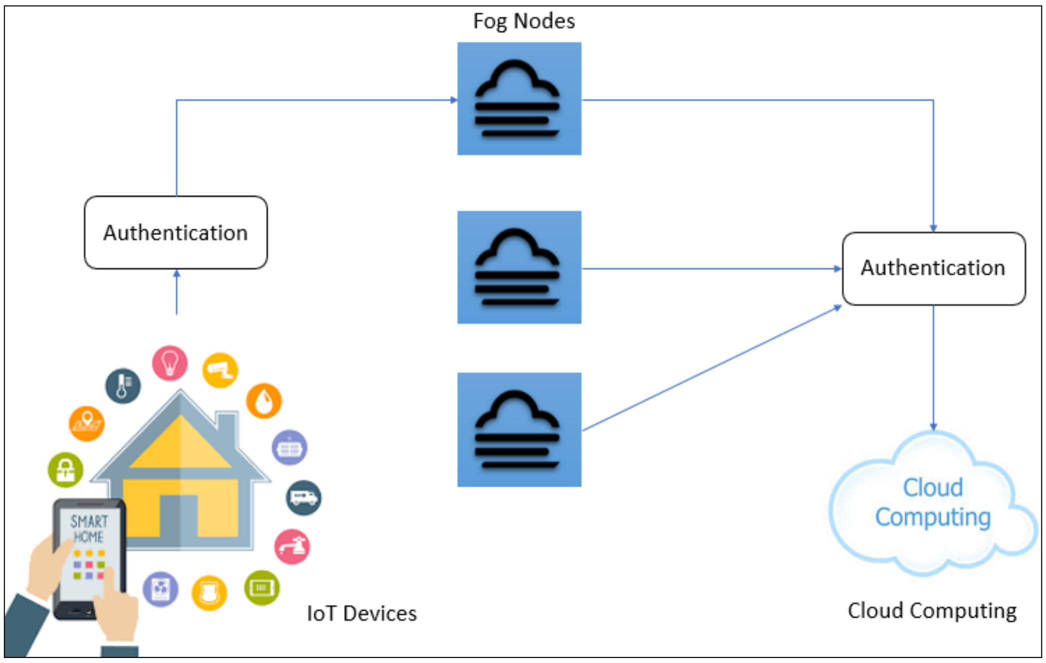


Figure 4.1: Authentication Process

Following are the core design components of the proposed solution:

1. Cloud Layer
2. Fog Layer
3. IoT devices
4. Intrusion detection system
5. Deep packet inspection software

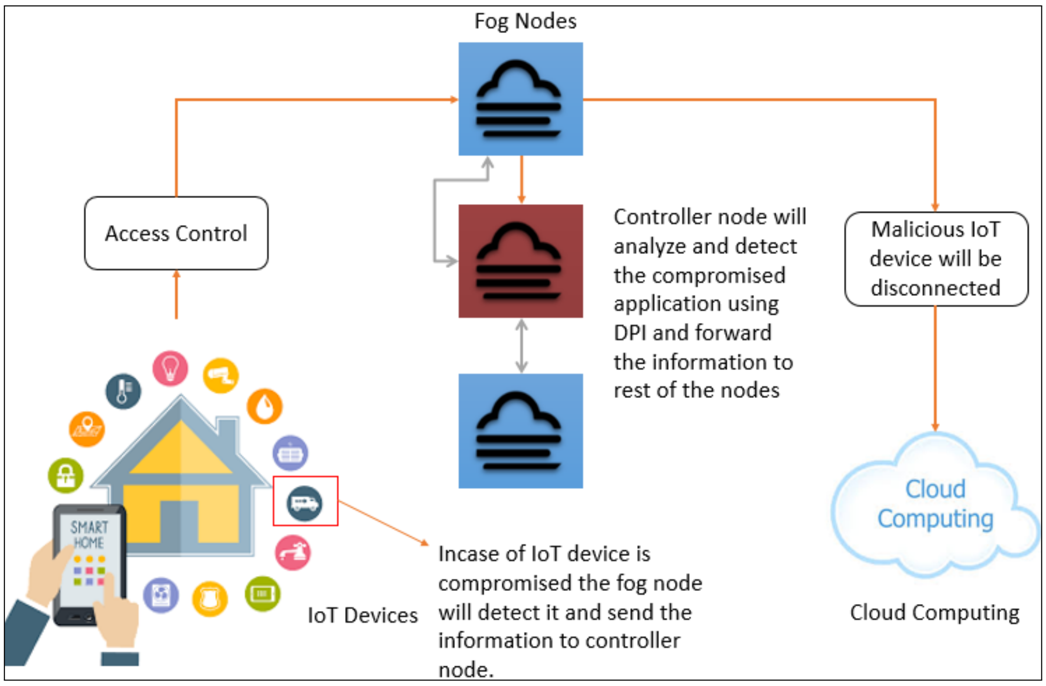

Fog Nodes
Access Control
Controller node will analyze and detect the compromised application using DPI and forward the information to rest of the nodes
Malicious IoT device will be disconnected
SMART HOME
Cloud Computing
Incase of IoT device is compromised the fog node will detect it and send the information to controller node.
IoT Devices
Cloud Computing

Figure 4.2: Access Control Scheme

4.3 Design of Proposed Scheme

In this section, the details about how the proposed scheme will work are provided.

The proposed scheme can be divided into two main parts. The first is authentication of the IoT devices. In this scheme, we have used RSA (Rivest,Shamir,Adleman) SecurID authentication mechanism. The reason for using RSA is to keep the environment scalable and dynamic. In symmetric cryptography authentication for IoT devices, key storage management is a big issue. Furthermore, keys between the participating entities needs to be shared before hand. We are using multiple fog nodes in our fog layer and each one will be required to have the same information. Hence, asymmetric authentication protocol is used for the authentication purpose. The literature review of authentication protocol conducted in subsection of **Literature Review** shows that RSA is a protocol which is suitable for a dynamic and scalable environment. In our implementation the device sends a packet encrypted with its private key and the fog node decrypts the packet with the public key of the IoT device, to verify its identity. The authentication packet contains IoT device identity, encrypted data and timestamp. The communication aspect is handled by standard event-based messaging process. For the generation of hash function we have used HMAC-SHA Signature512. The RSA scheme is implemented, we are using a hash-based message authentication scheme so that every message sent by the IoT device will be verified for integrity.

The second phase of the scheme is access control. For this purpose an intrusion detection system is installed on each fog node. Restrictive access technique is used to weed out the malicious/ compromised authenticated devices on the base of their behavior. The basic principle being followed is that the fog node set a specific acceptable criteria for the IoT devices. If any device behaves outside of this criteria the fog node will isolate the device and run a scan. In our implementation the criteria defined for the IoT device compliance is message integrity (by checking the hash value attached to each message) and threshold of traffic received from a single device defined. If the device is found compromised than it will be disconnected from the cloud and controller node will analyze its traffic. Controller node is a fog node which is dedicated to inspect the traffic from compromised device using deep packet inspection and analyze/ identify the malicious application. After identification of the application the controller node will forward this information to all the other fog nodes in the layer and they will update their signature repository.

In figure 4.3 the flowchart of the proposed scheme is provided. The flowchart describes how our proposed scheme is working and what are the different stages of the process. In the first step IoT devices needs to connect with the cloud to transmit information, so they send an authentication request. In our proposed scheme this authentication request will be received at the fog layer. The fog node will authenticate the IoT device and connect it to the cloud. Furthermore, it will continuously monitor the behavior of the devices on the parameters defined by the cloud service provider. If the IoT device is not authenticated the fog layer will drop the connection request. In case the IoT device starts to act suspiciously the fog node will terminate its connection with the cloud and run a full scan on the device. If the device is not compromised the fog layer will reconnect the device to cloud. However, if the device is compromised the fog layer will terminate the connection with the device completely. Additionally, it will send the traffic packets from the malicious device to the controller node (fog node with deep packet inspection tool) for deep packet inspection. Once the controller node identifies the malicious application compromising the device it will create a new ruleset/ policy. Next the controller will forward this ruleset/ policy to all the other fog nodes in the fog layer. Fog nodes will update their intrusion detection repository accordingly.

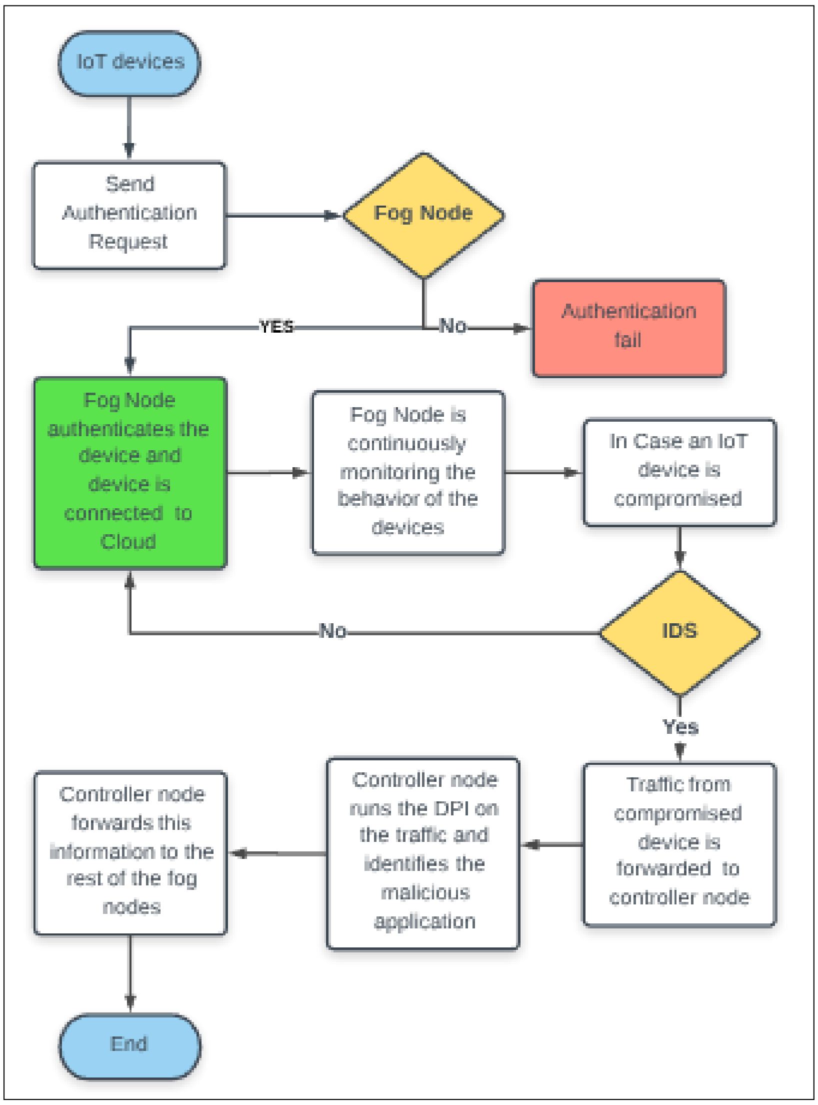

IoT devices
Send Authentication Request
Fog Node
YES
No
Authentication fail
Fog Node authenticates the device and device is connected to Cloud
Fog Node is continuously monitoring the behavior of the devices
In Case an IoT device is compromised
IDS
No
Yes
Traffic from compromised device is forwarded to controller node
Controller node runs the DPI on the traffic and identifies the malicious application
Controller node forwards this information to the rest of the fog nodes
End

Figure 4.3: Flowchart of the proposed scheme

Chapter 5

Implementation

This chapter serves the purpose of describing the implementation of access control for cloud based IoT devices using fog computing framework. The next pages explain the proposed solution, general overview of the deployed system, components/ softwares required to implement the solution and installation instructions followed by how to execute the scheme in an use case scenario.

5.1 Bird's Eye View

The first section of this chapter provides bird overview of the implemented access control scheme in fog computing framework.

In figure 5.1 the authentication process is explained. The device sends a packet encrypted with its private key and the fog node decrypts the packet with the public key of the IoT device, to verify its identity. The authentication packet contains IoT device identity, encrypted data and timestamp. The communication aspect is handled by standard event-based messaging process. For the generation of hash function, we have used HMAC-SHA Signature. For the authentication purpose we are using a hash-based message authentication scheme so that every message sent by the IoT device will be verified for integrity. After the device is authenticated and shows compromised behavior then the fog node will isolate it and check for malicious content. For this purpose, a fog node known as controller is dedicated to inspecting the traffic from the malicious IoT device. Deep Packet Inspection (DPI) is used in the controller to identify the application the attacker has compromised.

The Bro IDS implemented on fog node to check the suspicious activity and declare the device compromised or not. Bro IDS is selected for this implementation as it is an open source IDS with additional features of full programmability and flexible network security monitor with event correlation. Furthermore,we are implementing our IDS on Raspberry Pi which is running a Raspbian OS. The Raspberry Pi has 1 GB RAM and we needed to opt a light-weight open source IDS. Bro IDS is comparatively a light-weight IDS and does not require a lot of hardware resources. Bro IDS detection is based on the policy scripts which are user friendly and designed to describe what kind of traffic or activities that are looked as malicious. The Bro IDS is placed in fog node to keep the detection mechanism closer to the edge of the network and make the detection process prompt. So that traffic from IoT devices using the same application can be checked. In the next phase controller sends the details/ rule to rest of the fog nodes in the layer to block similar compromised devices.For Deep Packet Inspection nDPI tool is implemented, it is an opensource tool. DPI is used for application detection phase of the scheme. nDPI is used to detect the application protocol of communication flow. The results can be an output on different level i.e. the output is a mix of results on various levels:

1. IP protocols (i.e., TCP or UDP)
2. Application protocols (e.g., DNS, HTTP, SSL, BitTorrent, or SMTPS)
3. Types of the content (e.g., MPEG, or Flash)
4. Service providers (e.g., Facebook, or YouTube).

In our implementation to achieve these goals following process is followed:

1. IoT devices sends connection request to Fog layer.
2. Fog layer authenticates using RSA algorithm and responds back.
3. At Fog Layer BRO IDS will be deployed to check the behavior of the IoT devices.
4. Fog node will send connection request to Cloud.
5. Cloud will authenticate Fog node and the authenticated IoT devices by the Fog node will be by default authenticated.
6. All the devices authenticated by the Fog node can communicate with cloud.

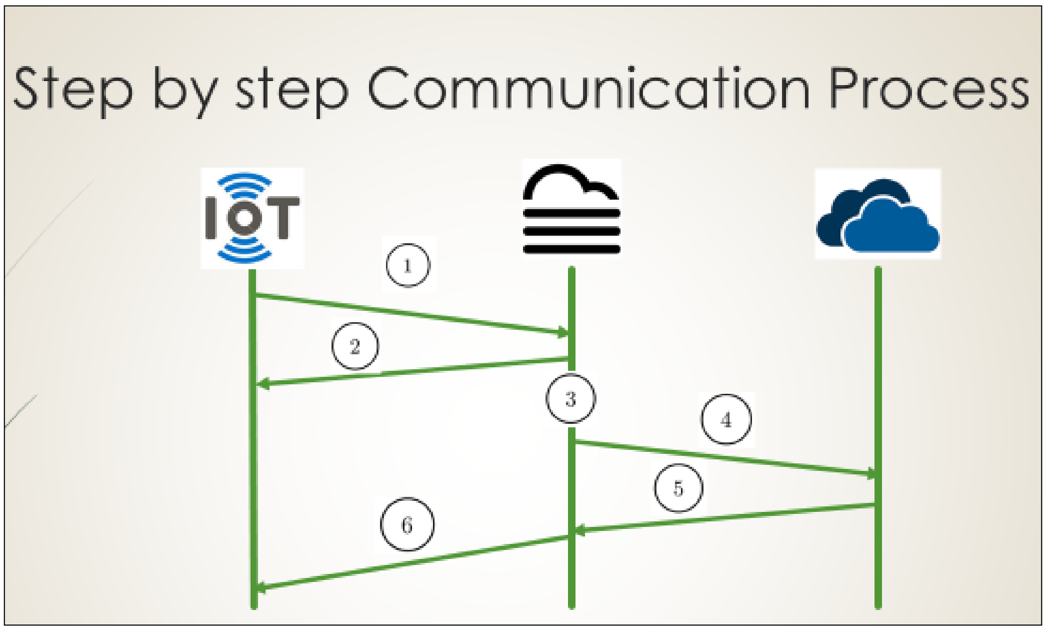


Figure 5.1: Authentication Process

5.2 Hardware Requirement

For the implementation of the access control framework we used the following hardware for deploying different layers:

1. For Cloud layer, Acer "Aspire E 15" with 12 GB RAM and Core i7.
2. For Fog layer, Raspberry Pi
3. For IoT devices, Raspberry Pi
4. USB WiFi Dongle
5. USB Keyboard
6. USB Mouse
7. SD Card
8. Micro SD Card Reader
9. HDMI Display
10. HDMI cable
11. 5V Power supply cable

5.3 Software Requirement

The softwares required for the deployment of this scheme are;

1. VMware
2. OpenStack
3. Etcher
4. Raspbian OS
5. Ubuntu OS
6. BroIDS
7. nDPI
8. Wireshark
9. VNS Server and Client

5.4 Implementation Plan

To achieve the proposed implementation the following steps were followed;

1. Create an IoT devices network consisting of multiple IoT surveillance devices on a Raspberry Pi.
2. Next configure OpenStack in Ubuntu based virtual machine to create cloud node.
3. To create fog layer Raspbian OS is installed on Raspberry Pi and two nodes out of the three nodes have IDS and one node has DPI i.e. in future will be referred as controller node.
4. Create communication links between all the entities.
5. IoT devices are authenticated by Fog node and Fog node connect them to Cloud after verification.
6. Incase any IoT device behaves abnormally the Fog node will disconnect it from the Cloud and run a scan on the device.

7. Once the scan confirms that IoT device is compromised the traffic from that device will be forwarded to the controller.

8. At controller DPI tool will be used to detect the type of application which is generating the malicious traffic.

9. After successful detection of the compromised application controller node will forward this information to the rest of the fog nodes in the network.

10. The efficiency and performance of the proposed scheme will be evaluated on the basis of;

    (a) Detection time
    (b) Packet Inspection
    (c) Application detection accuracy

5.5 Installation Instructions

This section compromises of details how we implemented the different components and created the required environment. Namely setup of the deployed cloud, deployed fog nodes, its environment, and all the needed components to execute the framework as intended. The next chapter explains the setup of the whole environment and how all the devices are connected.

5.5.1 OpenStack

This section is intended to be a guideline for beginners trying to launch OpenStack. We will describe what step-by-step instructions for setting up OpenStack on Ubuntu. Finally, we will describe the options the OpenStack dashboard contains and how to use them.

A joint effort of RackSpace and NASA, OpenStack is an open source collection of python-based softwares that is used to provide infrastructural and platform support for Cloud services. It is one of the most widely-used software for deploying Cloud for setting up on your personal computer for experimentation as well as on enterprise-level. It is easier for beginners to think of it as a Cloud operating system which has the capability to control exponential pools of computing, storage, and networking resources that can be dynamically provisioned on-demand.

At the core of the Cloud is a concept called virtualization, which is an important concept to grasp before moving forward with an explanation of OpenStack. Basically, virtualization abstracts hardware resources from systems running on top of them. This way, the physical boundaries of resources such as a server, storage, applications, operating systems, etc. can become virtualized, allowing the sharing of pools of resources among various tenants. It must be noted that virtualization is only a fraction of what OpenStack actually has to offer, but it is a start.

Installing on Ubuntu

OpenStack can be installed by three different methods it depends on your goals which method you select.

1. From an ISO image
2. Scripted Installation
3. Manual Installation

In this manual, we are doing scripted installation and step-by-step instructions are provided.

Scripted Development Installation

The simplest way to deploy Open Stack is by using a developer's tool Dev Stack. It is simple to install and provides a platform for the deployment of Open Stack.

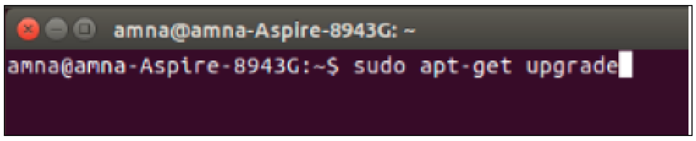
Dev Stack is a documented shell script to build complete Open Stack development environments and is located at http://devstack.org/.

The minimum requirement for installation of Dev Stack is Ubuntu 12.04 LTS, Fedora 20 or Cent OS/RHEL 7. We are using Ubuntu 14.04 (Trusty).

Step-by-Step Instructions

1. Install Ubuntu on your machine or VM. You can use an image, bootable hard drive, CD or DVD to install it.

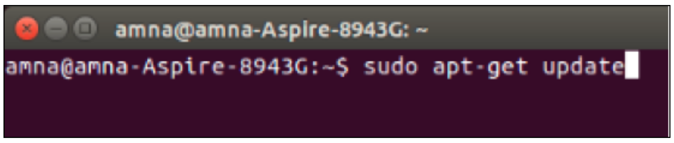
2. After successful installation of the operating system upgrade it.So that you have all the latest packages available to you.



```
amna@amna-Aspire-8943G: ~
amna@amna-Aspire-8943G:~$ sudo apt-get upgrade
```

Figure 5.2: Upgrade Ubuntu

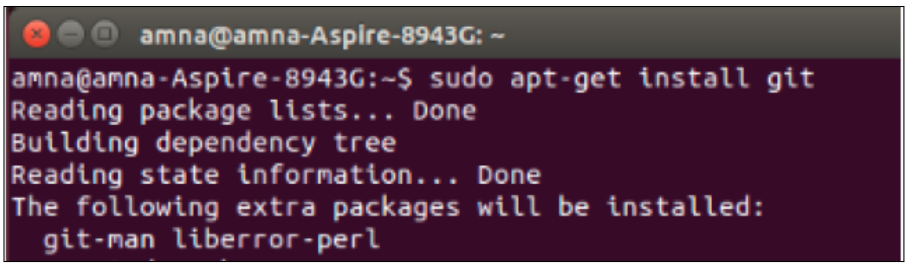
3. Now install all the updates of the operating system.



```
amna@amna-Aspire-8943G: ~
amna@amna-Aspire-8943G:~$ sudo apt-get update
```

Figure 5.3: Update Ubuntu

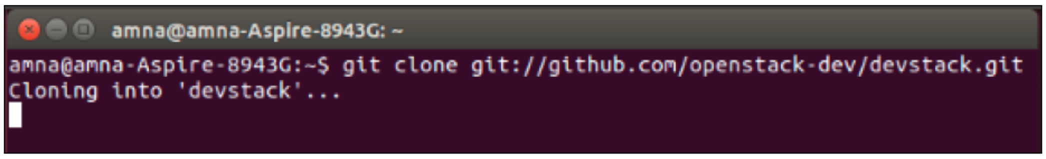
4. Now install git to Ubuntu.



```
amna@amna-Aspire-8943G: ~
amna@amna-Aspire-8943G:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  git-man liberror-perl
```

Figure 5.4: Install GitHub

5. The next step is to clone the DevStack repository to the operating system.

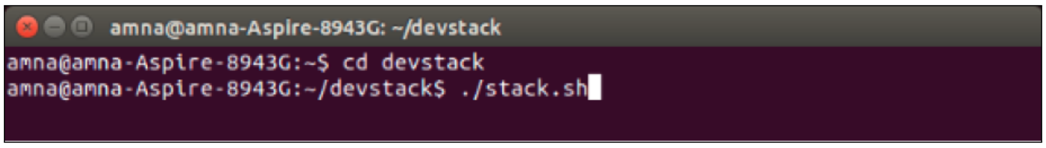


```
amna@amna-Aspire-8943G: ~
amna@amna-Aspire-8943G:~$ git clone git://github.com/openstack-dev/devstack.git
Cloning into 'devstack'...
```

Figure 5.5: Downloading DevStack

6. Now there are two methods to proceed forward:

(a) By directly running .stack.sh

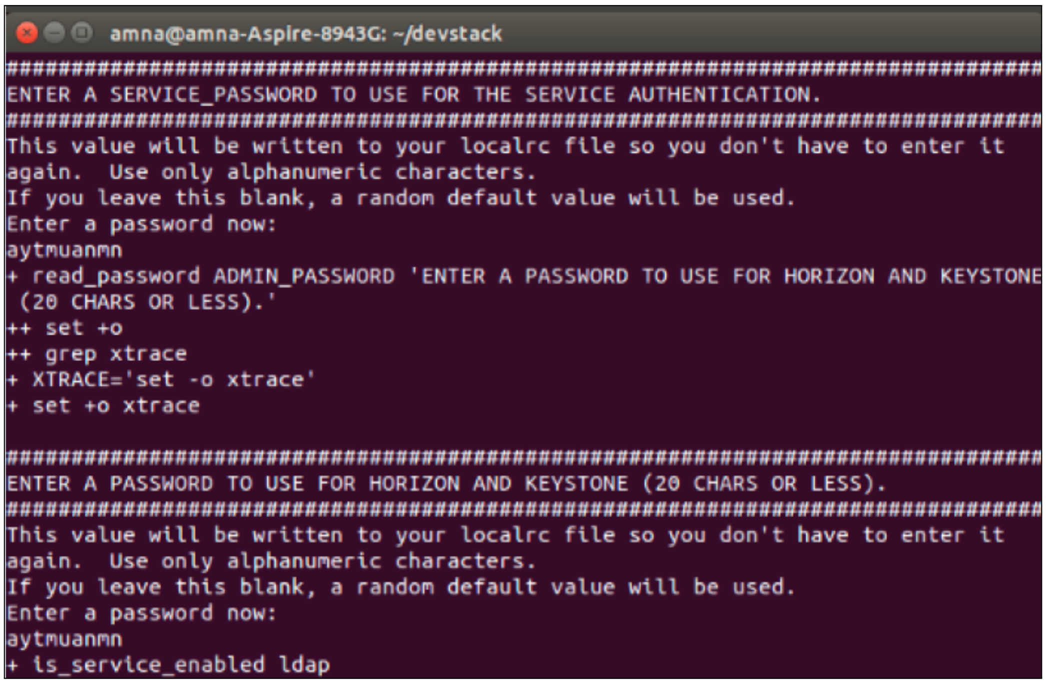


```
amna@amna-Aspire-8943G: ~/devstack
amna@amna-Aspire-8943G:~$ cd devstack
amna@amna-Aspire-8943G:~/devstack$ ./stack.sh
```

Figure 5.6: Compiling DevStack

Now it will ask for different passwords;

i. Password for Database.
ii. Password for RABBIT.
iii. Service_Token to use for the Service Admin Token
iv. Service_password to use for the service authentication, and
v. Password for Horizon and Keystone



```
amna@amna-Aspire-8943G: ~/devstack
################################################################################
ENTER A SERVICE_PASSWORD TO USE FOR THE SERVICE AUTHENTICATION.
################################################################################
This value will be written to your localrc file so you don't have to enter it
again.  Use only alphanumeric characters.
If you leave this blank, a random default value will be used.
Enter a password now:
aytmuanmn
+ read_password ADMIN_PASSWORD 'ENTER A PASSWORD TO USE FOR HORIZON AND KEYSTONE
 (20 CHARS OR LESS).'
++ set +o
++ grep xtrace
+ XTRACE='set -o xtrace'
+ set +o xtrace

################################################################################
ENTER A PASSWORD TO USE FOR HORIZON AND KEYSTONE (20 CHARS OR LESS).
################################################################################
This value will be written to your localrc file so you don't have to enter it
again.  Use only alphanumeric characters.
If you leave this blank, a random default value will be used.
Enter a password now:
aytmuanmn
+ is_service_enabled ldap
```

Figure 5.7: Creating passwords from terminal

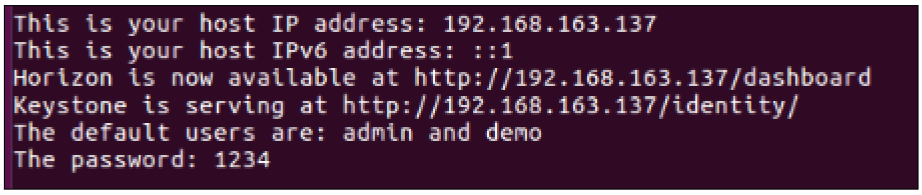
(b) The second simpler and faster method is to create a local.conf file The minimal configuration settings that are required in the file.

```
[[local|localrc]]
SERVICE_TOKEN=aaytmuanmn
ADMIN_PASSWORD=aytmuanmn
MYSQL_PASSWORD=aytmuanmn
RABBIT_PASSWORD=aytmuanmn
SERVICE_PASSWORD=aytmuanmn
```

These settings can be written in the local.conf to install neutron (network) tab in your OpenStack dashboard.

```
disable_service n-net
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
# Install the tempest test suite
enable_service tempest
```

7. Irrespective of the fact that you have followed a or b method in step 6 the output will be same. You will have successfully installed OpenStack and you will get login and password of your credentials as well as your host ip.



```
This is your host IP address: 192.168.163.137
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.163.137/dashboard
Keystone is serving at http://192.168.163.137/identity/
The default users are: admin and demo
The password: 1234
```

Figure 5.8: Openstack credentials

8. Open the horizon URL in any browser and you will see and authentication page. Enter your credentials in it. Use the admin or demo credentials the setup has created. Admin, obviously allows more privileges.

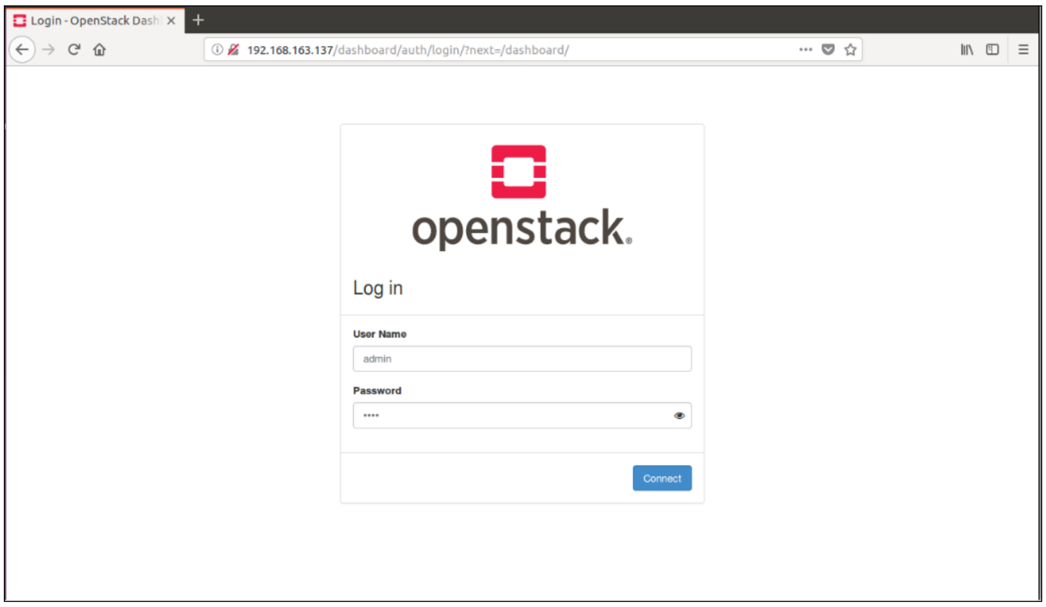
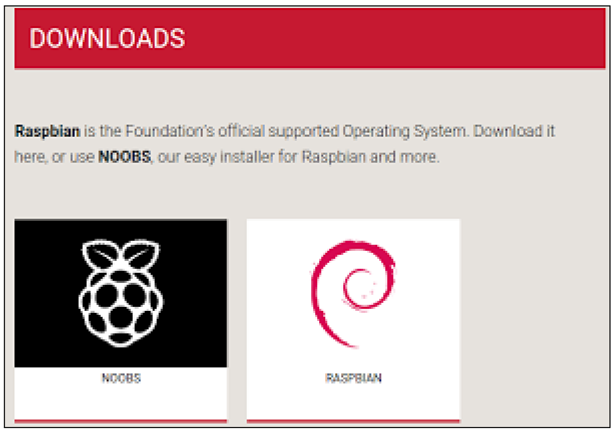


Figure 5.9: Openstack successfully installed

5.5.2 Raspbian OS

Aiming for a fast and reliable Fog environment, Raspbian the officially supported operating system for Raspberry Pi was selected. *Raspbian stretch with Desktop* is a desktop based Debian stretch image which requires around 4.7 Gb space. Following configuration steps are required to successfully install raspbian on Raspberry Pi.

1. Download *Raspbian stretch with Desktop* from "https://www.raspberrypi.org/downloads/raspbian/" and unzip the containing image.



DOWNLOADS

Raspbian is the Foundation's official supported Operating System. Download it here, or use **NOOBS**, our easy installer for Raspbian and more.

Figure 5.10: Download Raspbian stretch with Desktop

2. Use *Etcher* to Flash the .iso file to minimum 8 Gb SD card. In figure 5.11 Etcher is used to flash Raspbian OS on SD card.

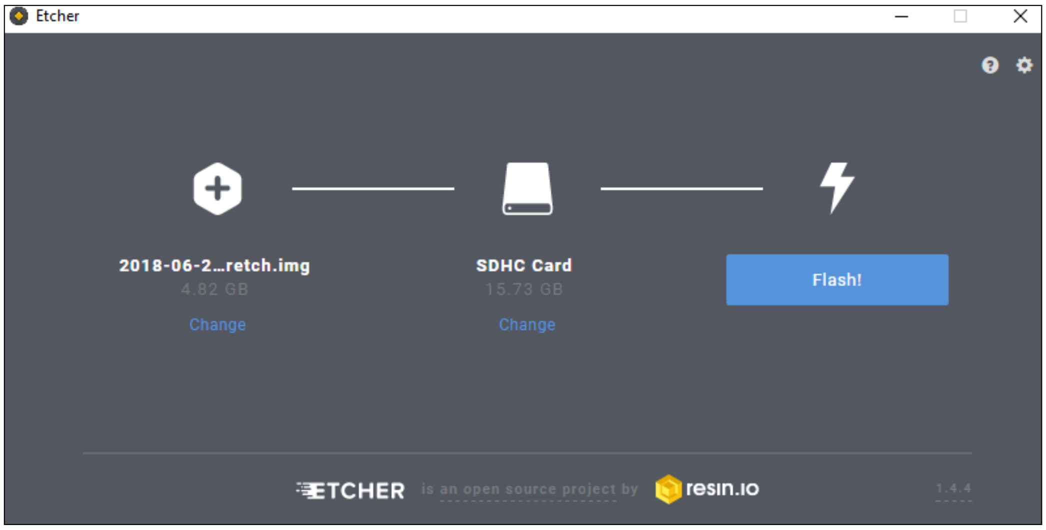


Figure 5.11: Flashing Raspbian OS using Etcher

3. After the software has validated the SD card plug it in the SD card slot in Raspberry Pi device. In figure 5.12 Etcher is used to validate Raspbian OS on SD card.

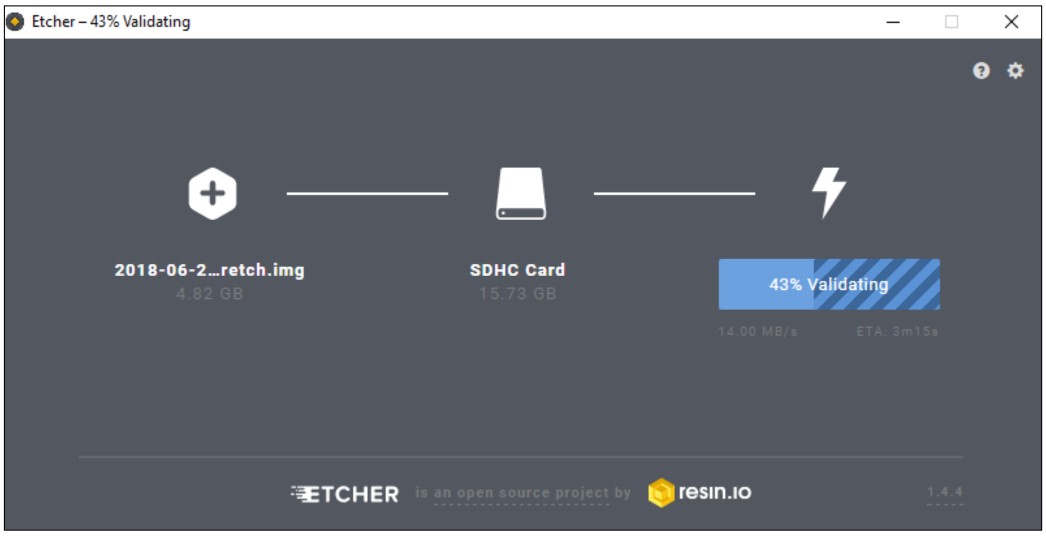


Figure 5.12: Validating Raspbian OS on SD card using Etcher

4. Connect the wifi dongle, mouse and keyboard to the USB slots in Raspberry Pi, connect it to a HDMI display and plug in the 5V adapter.

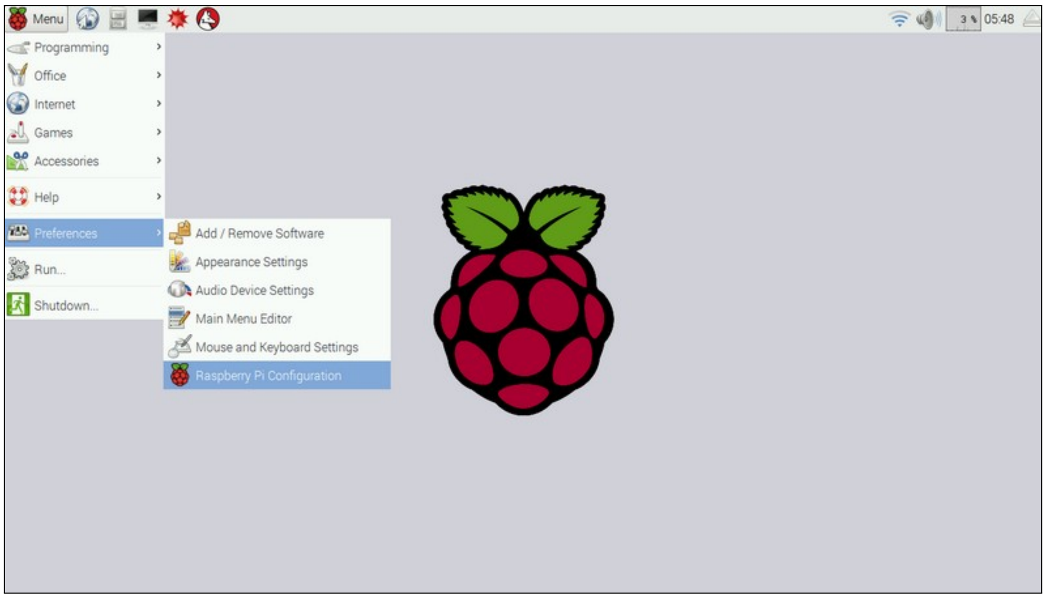
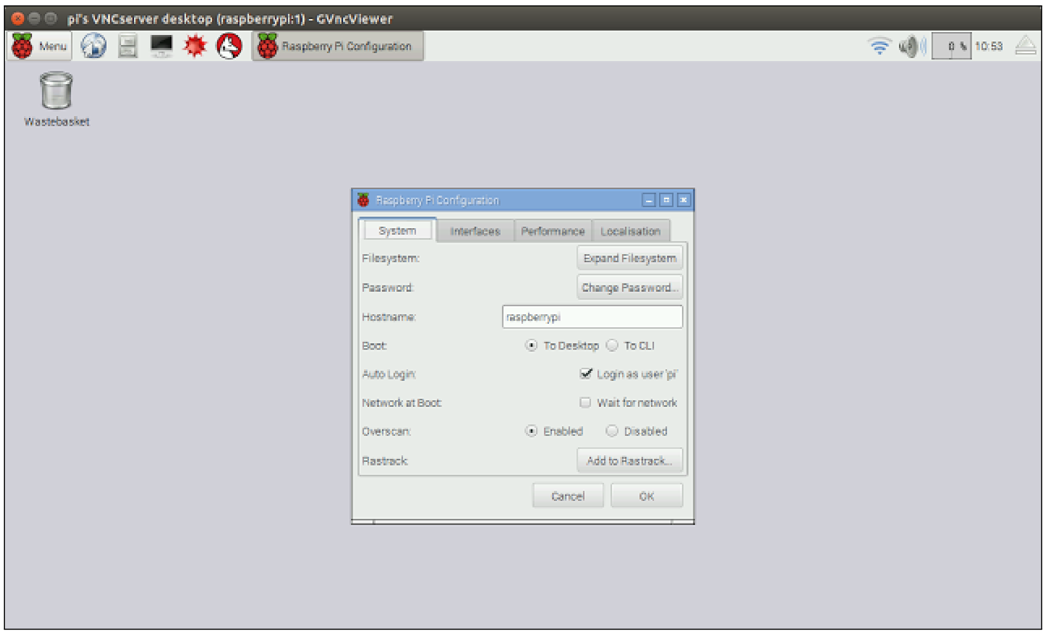


Figure 5.13: Start-up Screen

5. Once the Raspberry Pi is powered on it will prompt for configuration settings. After selecting the appropriate settings the system will reboot.



Filesystem:
Password:
Hostname: raspberrypi
Boot: [✓ To Desktop] To CLI
Auto Login: [✓ Login as user 'pi']
Network at Boot: Wait for network
Overscan: [✓ Enabled] Disabled
Rastrack:

Figure 5.14: Configuration of Raspberry Pi

6. Raspbian OS is successfully installed on the Raspberry Pi.

5.5.3 BroIDS

Intrusion detection techniques are used in any computing environment as a layer of defense. The basic aim is to detect any malicious activity well before any significant harm is possible. The general idea is to detect and identify attacks by either analyzing system artifacts (such as log files, process lists, etc.), or by keeping track of network traffic. Two main approaches used are signature based detection and anomaly-based detection. Signature based detection works by defining patterns of known attack signatures. If the system is found to be processing any code similar to those signatures, it is detected suspicious and marked as an intrusion. On the other hand, anomaly based detection works by analyzing activities performed on the system. Initially, a profile for a particular system is created by recording normal activities (e.g., by setting thresholds for normal bandwidth usage). If later on, the system's behavior is analyzed as anomalous to the profile defined, it is marked as an intrusion. Whereas signature-based detection techniques (also called misuse pattern matching) cannot detect unknown attacks, anomaly based techniques usually result in huge false positives or negatives.

Bro is a passive, open source network traffic analyzer. Primarily, a security monitoring tool that inspects deeply for suspected activities. However, can also perform wide range of network analysis tasks outside of the security domain. A platform for traffic analyses which is fully customizable and extensible. Like python, the system has a large number of pre-built functionality. Bro provides user with domain specific, Turing-Complete Scripting language but still we can use it in novel ways by writing our own code.

BroIDS is implemented using Sweet Security solution, it is a lightweight open-source software. We want the fog node to monitor all the traffic, we need to install software to inspect the traffic and analyze it. For this purpose we install the BroIDS and dependencies by following the process mentioned below. To make the Bro intelligent Sweet Security solution has integrated Critical Stack threat intelligence integration. Critical Stack is a free aggregator of threat intelligence feeds. Its a simple point-and-click integration to pull information, such as Tor Exit node IP addresses, known malicious IPs, or known phishing domains. BroIDS sends email notification of attacks, however, it is a better approach to integrate another open source Log Manager. For this purpose Logstash is used to receive alerts of different threats and attacks. To install Sweet Security solution in Raspberry Pi below mentioned steps were followed:

1. Download the Sweet security solution software to Raspberry pi using;

   ```
   git clone --depth=1
       https://github.com/travisfsmith/sweetsecurity
   ```

2. Prerequisites for installation of BroIDS are:

   (a) Libpcap
   (b) OpenSSL libraries
   (c) BIND8 library
   (d) Libz
   (e) Bash (for BroControl)
   (f) Python 2.6 or greater (for BroControl)

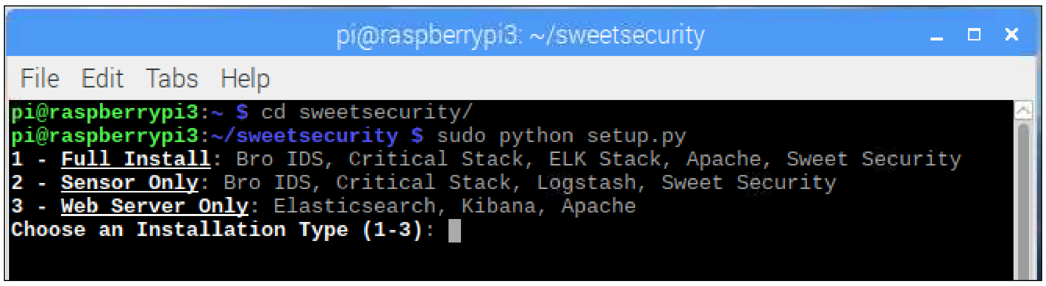
3. Prerequisites for this software are Python 2.7 and Java 1.8.

4. All the other packages will be installed during the installation process. Following are the list of these packages:

    (a) curl
    (b) cmake
    (c) g++
    (d) flex
    (e) bison
    (f) libpcap-dev
    (g) libssl-dev
    (h) python-dev
    (i) python-pip
    (j) python-flask
    (k) python-scapy
    (l) apache2
    (m) libapache2-mod-wsgi
    (n) swig
    (o) nmap
    (p) tcpdump
    (q) oracle-java8-jdk (Raspbian Only)
    (r) ant (Raspbian Only)
    (s) zip (Raspbian Only)

5. Run the following command to start the installation process. There are three packages to select from according to the requirement of the environment.

```
sudo python setup.py
```



```
pi@raspberrypi3:~ $ cd sweetsecurity/
pi@raspberrypi3:~/sweetsecurity $ sudo python setup.py
1 - Full Install: Bro IDS, Critical Stack, ELK Stack, Apache, Sweet Security
2 - Sensor Only: Bro IDS, Critical Stack, Logstash, Sweet Security
3 - Web Server Only: Elasticsearch, Kibana, Apache
Choose an Installation Type (1-3):
```

Figure 5.15: Sweet Security Solution Setup

5.5.4 nDPI

Deep packet inspection (DPI) include technique of packet filtering that tasks at the Application layer of the OSI reference model. The use of DPI is to find, classify, identify, reroute or block packets with definite data or program payloads that orthodox packet filtering, which inspects only packet headers, cannot detect.

Deep packet inspection (DPI) and its study can be used to observe application and its quality of service. Deep packet inspection is a sort of data processing that looks in detail insides the data actually sent, and routes their data accordingly. It can be able to use for effortlessly innocent explanations, like making sure that the data is being process is being viruses free and content in the right format.

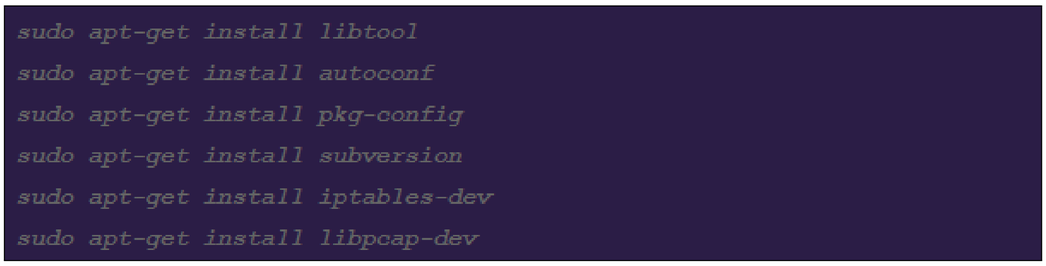
In many circumstances, insides of the packets go unchecked. But once a network supplier involves in deep packet inspection, it does the equal of opening up letters in a mail depot, and analysis the inside content. Software is used to examine the insides of content of each packet (and occasionally log it), and then a packet be able to be re-routed (or discarded entirely) if it permits certain criteria. Those criteria, might be the existence of a virus, or just arrangement of confident types of traffic that consuming most of the Bandwidth, like Skype, YouTube or Netflix above traffic that just requires to arrive in time, like web searching or application data.

Deep Packet Inspection is implementation of scanning content that permits ISPs to test packets wandering through their networks further carefully than ever previously. Usually, we only look over the packet which can only examine the header and collect information on the source and destination. Nonetheless Deep Packet Inspection can examine the code and look for marks of malware deprived of slowing down transmission (no latency). Upon finding of malware, the ISPs can essentially eliminate them from presence the risky packets, thus ending malware attacks.

Deep Packet Inspection is performed at controller node i.e. Raspberry Pi device. The tool selected to perform deep packet inspection is nDPI from ntop.

1. To install nDPI on Raspbian OS a debian based operating system following prerequisite are required;

```
sudo apt-get install build-essential
sudo apt-get install autoconf
sudo apt-get install pkg-config
sudo apt-get install subversion
sudo apt-get install iptables-dev
sudo apt-get install libpcap-dev
```

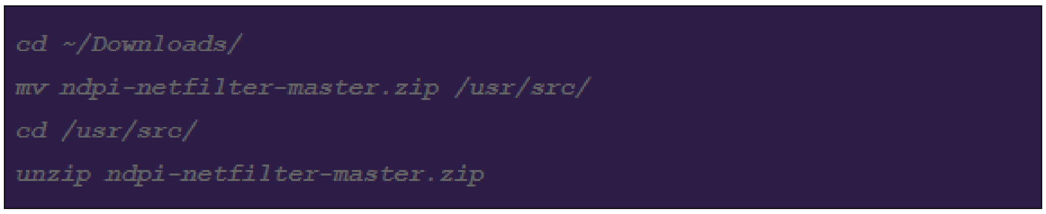


```
sudo apt-get install libtool
sudo apt-get install autoconf
sudo apt-get install pkg-config
sudo apt-get install subversion
sudo apt-get install iptables-dev
sudo apt-get install libpcap-dev
```

Figure 5.16: nDPI prerequisite

2. Next step is to fetch ndpi-netfilter source files.

```
cd /usr/src
sudo wget
    https://github.com/betolj/ndpi-netfilter/archive/master.zip
    -O ndpi-netfilter-master.zip
sudo unzip ndpi-netfilter-master.zip
```



```
cd ~/Downloads/
mv ndpi-netfilter-master.zip /usr/src/
cd /usr/src/
unzip ndpi-netfilter-master.zip
```

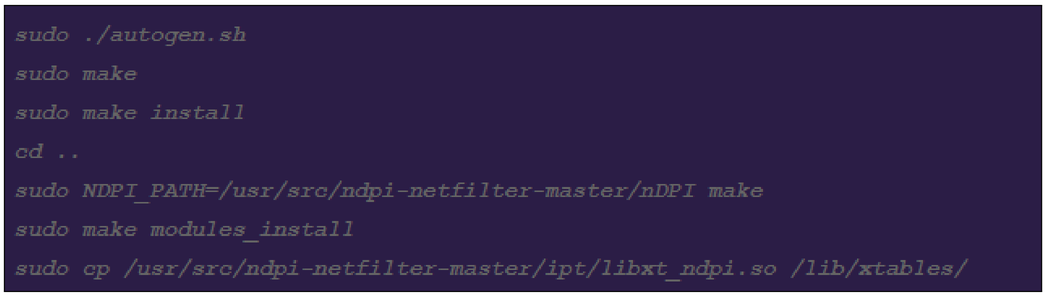
Figure 5.17: Fetch ndpi-netfilter source files

3. Once all the required packages are installed, prepare and compile the kernel and the kernel is compiled and properly installed in /boot/ go ahead and reboot your Raspberry Pi into the new kernel.

```
cd /usr/src/ndpi-netfilter-master/nDPI/
sudo ./autogen.sh
sudo make
sudo make install
```

4. Once the Raspberry Pi reboots compile the ndpi-netfilter.

```
cd ..
sudo NDPI_PATH=/usr/src/ndpi-netfilter-master/nDPI make
sudo make modules_install
sudo cp /usr/src/ndpi-netfilter-master/ipt/libxt_ndpi.so
   /lib/xtables/
```

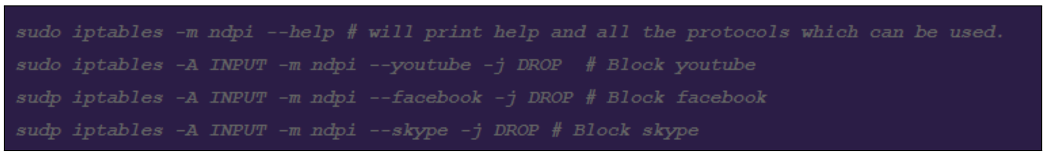


```
sudo ./autogen.sh
sudo make
sudo make install
cd ..
sudo NDPI_PATH=/usr/src/ndpi-netfilter-master/nDPI make
sudo make modules_install
sudo cp /usr/src/ndpi-netfilter-master/ipt/libxt_ndpi.so /lib/xtables/
```

Figure 5.18: Installing nDPI

5. If everything goes as planned, the following command can be used to confirm whether nDPI has been installed properly or not. This command will print out the basic usage details of nDPI module.

```
sudo iptables -m ndpi --help
```



```
sudo iptables -m ndpi --help # will print help and all the protocols which can be used.
sudo iptables -A INPUT -m ndpi --youtube -j DROP  # Block youtube
sudp iptables -A INPUT -m ndpi --facebook -j DROP # Block facebook
sudp iptables -A INPUT -m ndpi --skype -j DROP # Block skype
```

Figure 5.19: Confirming the installation of nDPI

Chapter 6

Validity of Proposed Research

In this chapter, we have explained the evaluation scenario implemented to assess our proposed access control scheme.

6.1 Evaluation Setup

For the sake of creating an evaluation scenario, we have used IoT devices generated data from a survilleaneous system. The data of the survilleaneous system gathered consists of data from number of security cameras and video survilleaneous. We are using Raspberry Pi device and generating the data from it and sending to Fog node as a live deployed IoT device. We connect the complete survilleaneous system to Fog layer. The complete scenario was deployed in KTH Lab in IAEC building.

Every camera is connected via internet to fog node and sending data as live feed to fog nodes. While fog node is continuously receiving data from IoT devices. Fog Node, is acting as a bridge between cloud and IoT devices. It also provides authentication and access control services to IoT devices. If an IoT device is authenticated by the fog node, only then it allows IoT device to connect itself with the cloud. The second and most important task of fog node is to hold data temporarily, which is received from IoT devices, the node then processes the data and sends it to cloud for persistent storage.

Any malicious traffic received from the IoT device is detected from the Fog node and Fog node disconnects the Cloud and IoT device connection. Malicious traffic is forwarded to the controller node and it performs deep packet inspection to evaluate the type of application generating malicious data. Once the controller identifies the malicious application it creates a ruleset and sends it to all the other Fog nodes in the layer to update their ruleset repository.

6.2 Evaluation Scenarios

For the evaluation of our proposed scheme we have used two different scenarios:

1. Validation based on NIST Security Criteria.
2. By launching critical attacks from IoT devices.

6.2.1 Evaluation Scenarios - I: Validation based on NIST Security Criteria

Directions in security metrics research carried out by NIST [26], is a document of practical guidelines which provides a security metrics to evaluate the security of a system. It provides various criteria and methods for the evaluation and testing of a security system. Additionally, access control systems evaluation metrics and assessment have been discussed in different NIST document [25] [46] [24]. The attributes used to evaluate the information security system contribute to the security of the system. The security aspects used to evaluate a system and the relevant security properties are:

1. Organizational Security Objectives
2. Qualitative and Quantitative Properties
3. Correctness and Effectiveness
4. Measurement of Large versus the Small
5. Leading versus Lagging Indicators

To evaluate the security of our access control scheme, we follow the **Qualitative and Quantitative Properties** approach. Qualitative properties are based on measures or characteristics of complexity, scalability, portability, etc. of the software. Quantitative properties are based on figures and numbers statistics.

6.2.2 Evaluation Scenarios - II: By launching Critical Attacks from IoT Devices

To evaluate the above explained setup different critical attacks have been launched keeping the below mentioned parameters in mind. These parameters are evaluated against detection time, packet inspection and application detection.

1. **Excessive Traffic**
   Excessive traffic is sent to evaluate how the access control scheme will reactive in case of a denial of service attack.

2. **Accessing Barred Port**
   In this scenario, it is evaluated what will happen in case a device attempts to use prohibited ports for communication.

3. **Sending false data**
   By sending self-constructed or false data (gibberish) to cloud the attacker will try to overload the system. In this case, it is evaluated whether the fog layer will detect falsification of information or not.

4. **Device not responsive**
   In case the device stops communicating with the cloud will the fog layer able to detect it.

5. **Malicious Traffic**
   If an attacker sends malicious packets i.e. virus, worm, trojan or a piece of code which will be able to exploit the system will the scheme detect it.

6. **Compromised IoT**
   Somebody trying to hijack the device and attempting to use the IoT device credentials to perform some illegitimate operations, while impersonating original IoT device.

Detection Time

For this purpose, we are using BroIDS for the detection of any malicious activity. It takes only one to two minutes to detect any malicious behaviors, as we are not storing any data or doing any passive analysis, our solution inspects each packet and every bit actively on the fly. Hence it reports any abnormal behavior within minutes and take appropriate actions, which involves:

1. Disconnect the IoT device from the cloud.
2. Perform further analysis to identify that which application of IOT cause problem.
3. It also forwards its analysis to other fog nodes, so they can update their rules repository.

Packet Inspection

For packet inspection, deep packet inspection is used as it acts as both intrusion detection and prevention system. It can detect attacks which the deployed intrusion detection, intrusion prevention and firewall is unable to adequately detect. In packet inspection every single packet is opened and its content is inspected. Deep Packet inspection is based on defined rules and policies. This feature can be used to check if unapproved software is being used. It is more suitable for traffic monitoring applications, by disabling specific features that slow down the DPI engine while being them un-necessary for network traffic monitoring.

Our solution utilizes ntop Deep Packet Inspection (nDPI), which is an advanced solution for examining and managing network traffic. It is a form of packet filtering that locates, identifies, classifies, re-routes or blocks packets with specific data or code payloads that conventional packet filtering, which examines only packet headers, cannot detect.

It examines the contents of packets passing through a Fog Node and makes real-time decisions based on rules assigned to it, depending on what a packet contains. It helps in detection and interception of viruses and other forms of malicious traffic but, it can also be used for more nefarious activities like eavesdropping.

The main reason of using nDPI in our solution, that is, it hands encrypted content. The trend of Internet traffic is going towards encrypted content often using SSL. In order to support encrypted connections, we have added a decoder for SSL (both client and server) certificates, thus we can figure out the protocol using the encryption certificate. This allows us to identify protocols, that otherwise would be undetected.

Application detection

Proposed solution is adding an application-layer detection of protocols, regardless of the port being used. This means that it is possible to both detect known protocols on non-standard ports (e.g. detect http on ports other than 80), and also the opposite (e.g. detect Skype traffic on port 80). This is because nowadays the concept of port=application no longer holds.

As define earlier our solution utilizes nDPI which has a wide range of signatures against well-known applications. nDPI update its signature database on daily bases. We are identifying applications through pattern/ signature matching by analyzes each packet against a database.

6.3 Proof of concept setup

The setup for the evaluation scenarios compromise of the following components. Table 6.11 includes both, physical and logical components used to run the experimental test. Furthermore, Table 6.2 ,6.3 consists of information about input datasets used for the experiment as data generated from IoT devices. The purpose of having an input dataset is to test the performance, behavior and output of the deployed scheme. IoT Network Dataset [35] has been used to conduct the experiments. The dataset contains data from public and private IoT devices being used in the city of Santander.

Table 6.1: Logical and physical infrastructure

Component	**Type**	**Purpose**	**Characteristics**
OpenStack	Logical	Cloud	Linux x86 64 8 GB RAM, 4 Processors
Fog Nodes 1, 2, 3	Physical	Fog Layer	Raspberry Pi 2 Model B 1 GB RAM, Arm7 Quad Core Processor
IoT Device	Physical	Data Generator	Raspberry Pi 2 Model B 512 GB RAM, ARM11 Processor

Table 6.2: Devices used in the network and probability they are in users' possession

Device	**Mobility**	**Ownership (%)**	**Device_type**
Smartphone	Mobile	91	1
Car	Mobile	55	2
Tablet	Mobile	40	3
Smart Fitness	Mobile	22	4
Smartwatch	Mobile	5	5
Pc	Static	84	6
Printer	Static	53	7
Home Sensors	Static	15	8

Table 6.3: Public devices available in city of Santander

Data Model	**Description**	**Device_type**
Point of Interest	Specific point location that a user may find useful or interesting.	9
Environment and Weather	Object responsible of the environmental and weather monitoring.	10
Transportation	Vehicles, taxis or buses.	11
Indicator	Digital signage to display information.	12
Garbage Truck	Collection and transport of waste products.	13
Street Light	Street lamp to illuminate roads in the city.	14
Parking	Location designed for parking.	15
Alarms	Security supervisor or traffic monitoring.	16

In table 6.4 software distribution and their corresponding versions are shown.

Table 6.4: Version detail of Softwares used

Softwares	**Version**
Raspbian Stretch with Desktop	4.14
nDPI	2.2.2
Bro	2.5.4
Ubuntu	16.04
OpenStack	Queen
Logstash	6.3.1
Critical Stack	0.4.0

6.4 Results

The results are evaluated on the above mentioned criteria. We have used Raspberry Pis which are affordable and low power devices to implement security. All the softwares used are opensource. After implementing this solution we have reached the conclusion that this scheme provides concrete security.

6.4.1 Evaluation Scenario - I

In our scheme we have identified the security threats related to access control and identified the mechanisms to prevent the breach of security.

Threat vs. Security Mechanism

In table 6.5 threat evaluation has been carried out, the threats identified are unauthorized access, man in the middle and identity theft.

Table 6.5: Threat vs. Security Mechanism

Threat	**Effective**	**Security Control**
Unauthorized Access	Yes	The authentication packet being sent has Sensor ID+Encrypted Data+Time stamp
Man in the middle	Yes	Encryption is using private key
Identity Theft	Yes	For impersonation private key of the device is required

Test Cases

The security requirement of an access control mechanisms are highlighted in *Assessment of access control systems* by NIST [24]. We have identified our security objectives and defined the security-based test cases to evaluate the effectiveness of the proposed scheme.

Test Case 1

Table 6.6: Authentication Test - Accept

Test Case Title	**Authentication Test - Accept**
Test Case ID	Test01
Test Objective	To check the correctness of authentication process
Pre-Condition	Protocol must be in place for authentication
Post-Condition	IoT device will be connected to the Cloud
Procedure	1. Authenticate an IoT device 2. Next time access will be denied if the same private key is not used
Expected Result	Access Allowed
Actual Result	Access Allowed
Status Pass	Pass
Carried out By	Amna Riaz

Test Case 2

Table 6.7: Authentication Test - Deny

Test Case Title	**Authentication Test - Deny**
Test Case ID	Test02
Test Objective	To check the correctness of authentication process
Pre-Condition	Protocol must be in place for authentication
Post-Condition	For impersonation private key of the device is required
Procedure	1. Authenticate an IoT device 2. Next time access will be denied if the same private key is not used
Expected Result	Access Denied
Actual Result	Access Denied
Status Pass	Pass
Carried out By	Amna Riaz

Test Case 3

Table 6.8: Access Control Test - Allow

Test Case Title	**Access Control Test - Allow**
Test Case ID	Test03
Test Objective	To check the correctness of access control scheme
Pre-Condition	Protocol must be in place for authentication
Post-Condition	IoT device should send data to the cloud
Procedure	1. Authenticate an IoT device 2. Send data from the IoT device 3. Cloud should receive the data
Expected Result	Access Allowed
Actual Result	Access Allowed
Status Pass	Pass
Carried out By	Amna Riaz

Test Case 4

Table 6.9: Access Control Test - Terminate Connection

Test Case Title	**Access Control Test - Terminate Connection**
Test Case ID	Test04
Test Objective	To check the correctness of access control scheme
Pre-Condition	Protocol must be in place for authentication Threshold of the criteria defined
Post-Condition	Connection between IoT device and cloud should be terminated
Procedure	1. Authenticate an IoT device 2. Send data continuously from the IoT device 3. Cloud should receive the data 4. After the threshold of the criteria is achieved fog layer should terminate the connection
Expected Result	Access Denied
Actual Result	Access Denied
Status Pass	Pass
Carried out By	Amna Riaz

6.4.2 Evaluation Scenario - II

In the second evaluation scenario different attacks on the implemented scheme have been launched. As discussed in the section 6.2.2.

Table 6.10: Results from the Evaluation Scenario - II

	DoS	Accessing Barred Port	False Measure-ments	Unresponsiveness	Malicious Traffic	Compromised IoT
Detection Time	30 Sec	2 Sec	50 Sec	5 Sec	40 Sec	5 Min
Packet Inspection	Yes	Yes	Yes	N/A	Yes	Yes
Application Detection	Yes	Yes	N/A	N/A	Yes	Yes

Table 6.11: Detection Time of Different Types of DoS Attacks

Sr. No.	**DoS Attack Type**	**Detection Time**
1	ICMP Flood	30 sec
2	SYN Flood	47 sec
3	UDP Flood	72 sec
4	ACK Flood	94 sec
5	DNS Flood	56 sec

The results from table 6.11 are plotted in a graph shown in figure 6.1.

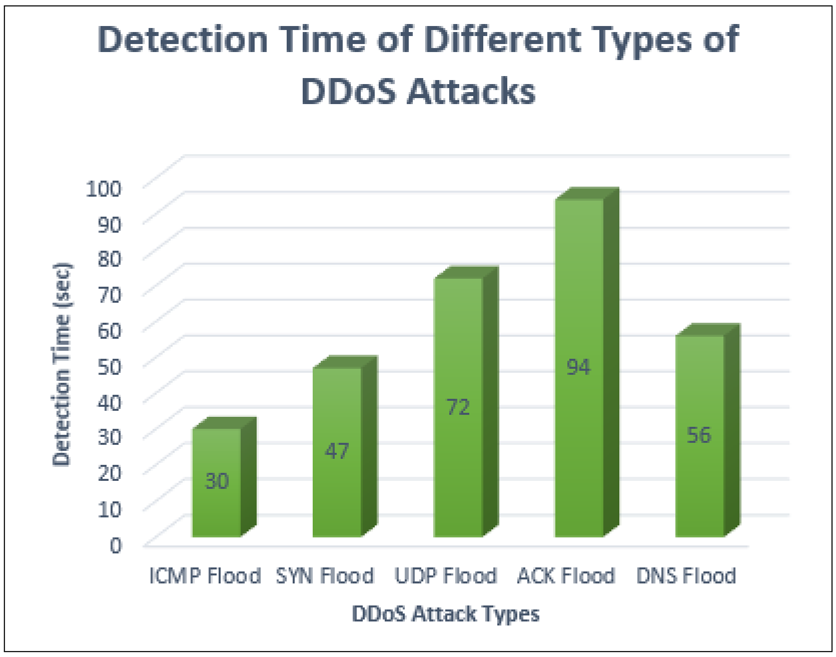


Figure 6.1: Graphical Representation of 6.11

Discussion

In this section, we will discuss the results from the implemented scheme.

1. In Table 6.10 different kind of attacks have been launched to evaluate the proposed scheme on a set of parameters.
   (a) Detection times are highly optimal as no other activity is being performed at the Fog layer and its sole purpose is intrusion detection.
   (b) Packet Inspection by nDPI tool is done for all the applicable scenarios.
   (c) Application detection is also done in all the applicable scenarios by:
      i. Pattern or signature matching
      ii. Protocol anomaly
      iii. IPS solutions
2. In Table 6.11 we have launched different types of DDoS attacks and recorded the detection time taken by the intrusion detection system to successfully detect the attack.

The Table 6.10 and 6.11 are showing results, which are produced after performing different tests in order to evaluate the system under different circumstances. It is noted that the measurements which are mention is the result metrics are approximately. Values may vary depending upon different variables like:

1. How many IoT devices are connected to Fog Node at the same time.
2. How much one device is sending traffic at a time.

Due to time restrictions we had to limit the scope of the research to aforementioned scenarios, anything beyond that is not covered, like:

1. Any other attack/ abnormality detection, which is not mentioned in result metrics.
2. This solution only detects those applications which are supported by nDPI, other application may not be detected by this solution.

6.5 Conclusion

Our validation for proposed access control scheme to stop the compromised IoT devices from launching DDoS attack consisted on two evaluation scenarios. We identified a research gap, defined the objective and followed the research methodology defined to fulfill it. The proposed access control scheme is designed based on existing techniques, standards and technology. The scheme has been evaluated by using the NIST security criteria and launching critical attacks from the IoT devices. Additionally, we have also performed threat evaluation by identifying the security threats against security mechanisms deployed. In the end test case scenario are used to evaluate whether the desired goals have been achieved.

Chapter 7

Conclusions

In this chapter, the concluding remarks about the thesis are given and future research prospects from our research work are highlighted.

7.1 Conclusions

Access Control is well-studied in context to traditional systems and cloud computing. However, IoT devices and fog computing due to their dynamic and scalable nature, cannot adhere to traditional access control solutions. Fog computing reduces the latency of services/ applications and improve the QoS. Smart devices based solution providing companies i.e. transportation systems, vehicular ad hoc networks will adopt fog computing irrespective of the security threats. As a security researcher, the need of time is to develop an access control framework which caters to all the security issues highlighted.

In this thesis, we have highlighted the security threats to cloud/fog enabled IoT devices and identified/ mapped how each layer is vulnerable to which threat. Next, we surveyed and analyzed the existing state of the art access control models in fog/ cloud computing and performed a performance analysis based on virtual environment the access model is deployed in i.e. cloud or fog, whether authentication mechanism is proposed or not, what kind of access control technique is used to devise the solution, can it detect insider attacks, is it scalable and resource utilization i.e. resources of which layer will be consumed (cloud, edge or end user). Then we have proposed an access control model for cloud computing using the resources of fog layer.

We have implemented this access control scheme where cloud resources are minimally utilized for the process of authentication and access control.

The deployed Fog layer is used to provide these crucial security controls. Fog Layer is acting as a bridge between cloud and IoT devices. It basically identifies the suspicious traffic and isolates the IoT device for further investigation. Due to unavailability of required resources proposed scheme couldn't be evaluated in a real life scenario.

Our validation for proposed access control scheme to stop the compromised IoT devices from launching DDoS attack consisted on two evaluation scenarios. We identified a research gap, defined the objective and followed the research methodology defined to fulfill it. The proposed access control scheme is designed based on existing techniques, standards and technology. The scheme has been evaluated by using the NIST security criteria and launching critical attacks from the IoT devices. Additionally, we have also performed threat evaluation by identifying the security threats against security mechanisms deployed. In the end test case scenario are used to evaluate the achievement of thee desired objectives.

IoT devices are exponentially increasing over the year and more computational power is being given to these devices. Need of the hour is to develop security controls tailored made according to their environment.

7.2 Future Work

This research work addresses most of the security issues faced by the cloud to control authentication and access control of IoT devices. Due to time limitation and resource limitation our work was limited. During this research, problem of authentication and access control were countered. We have implemented our proposed solution in a limited and resource constrained deployed environment. Future work from our research work:

1. Implement of this scheme in a real life dynamic and scalable scenario.
2. The limitation of Fog layer in this scenario can also be evaluated.

Appendix A

Screenshots of Components of the Implemented Scheme

The main components of our proposed scheme are;

1. Cloud

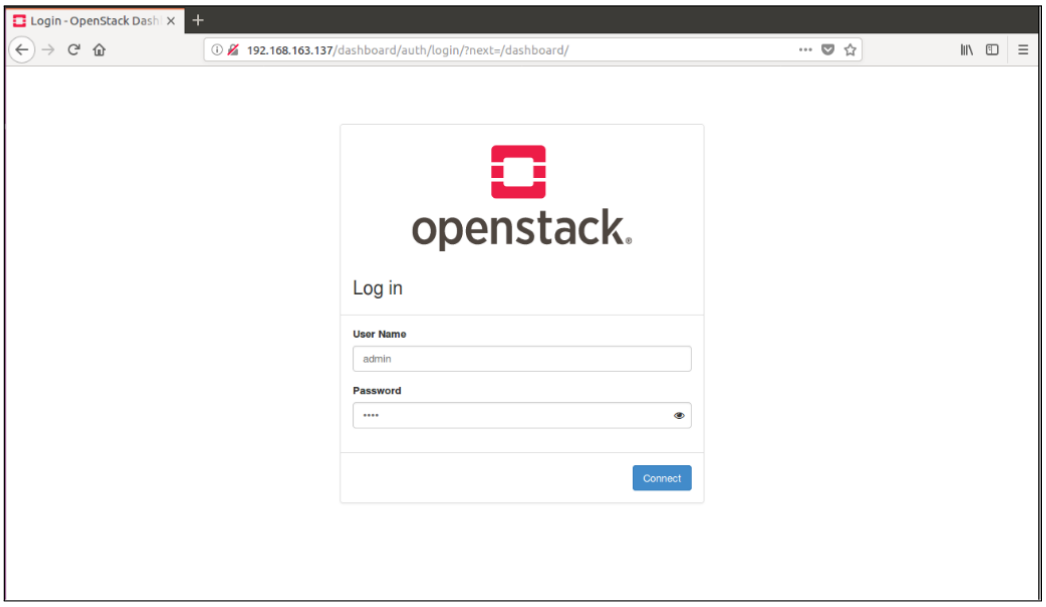


Figure A.1: Openstack

2. Fog Node

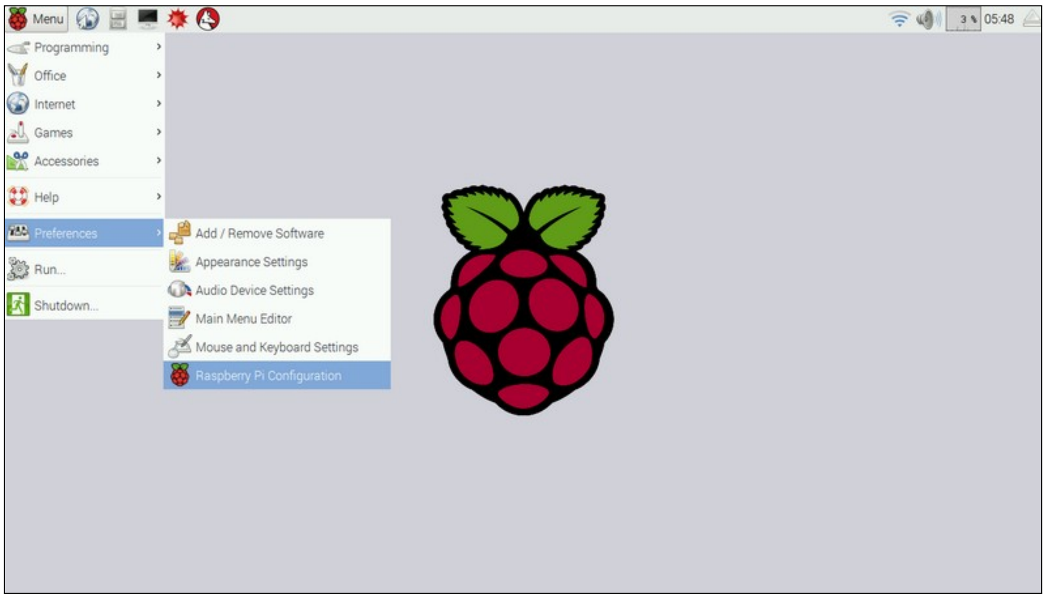
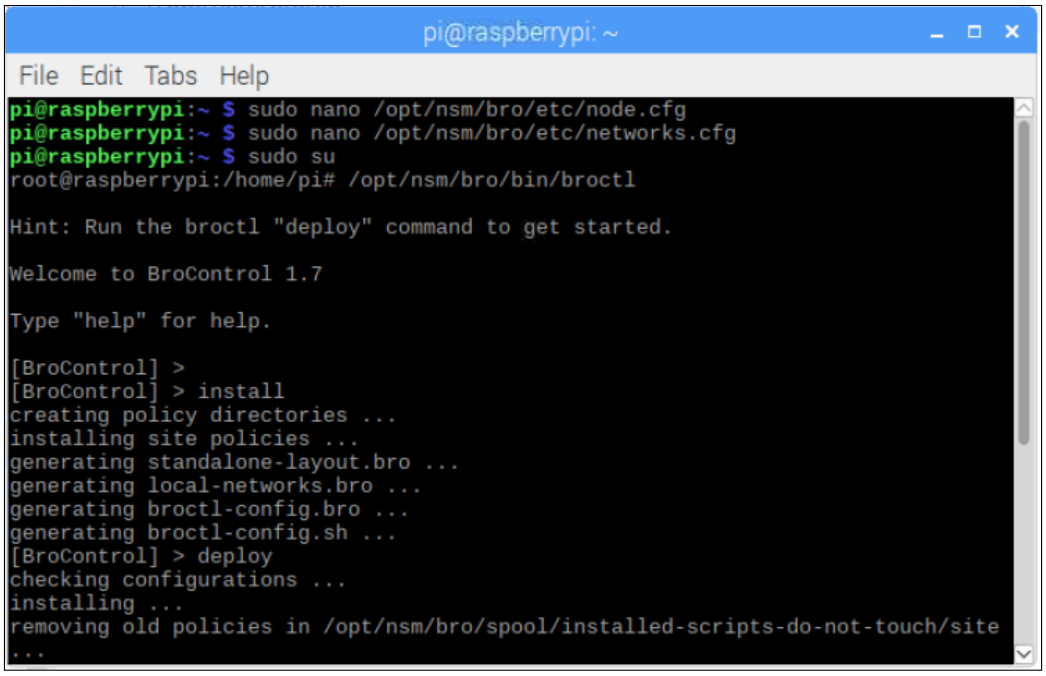


Figure A.2: Fog Node

3. Intrusion Detection System



```
pi@raspberrypi:~ $ sudo nano /opt/nsm/bro/etc/node.cfg
pi@raspberrypi:~ $ sudo nano /opt/nsm/bro/etc/networks.cfg
pi@raspberrypi:~ $ sudo su
root@raspberrypi:/home/pi# /opt/nsm/bro/bin/broctl

Hint: Run the broctl "deploy" command to get started.

Welcome to BroControl 1.7

Type "help" for help.

[BroControl] >
[BroControl] > install
creating policy directories ...
installing site policies ...
generating standalone-layout.bro ...
generating local-networks.bro ...
generating broctl-config.bro ...
generating broctl-config.sh ...
[BroControl] > deploy
checking configurations ...
installing ...
removing old policies in /opt/nsm/bro/spool/installed-scripts-do-not-touch/site
...
```

Figure A.3: Bro IDS

Appendix B

Configuration Problems and Solutions

During the installation of different softwares on Raspberry Pi a number of configuration issues popped up frequently. This section collates common configuration problems encountered during configuration and presents their solutions. The main components installed on Raspberry Pi devices are nDPI and BroIDS and. Each issue is discussed in the order it was faced during the configuration of the respective software.

Keep in mind Raspberry Pi is a credit card sized small and affordable computer, explicitly developed for the purpose of learning programming languages and implementing prototypes of projects for educational learning.The tiny size of Raspberry Pi makes it portable and a great development environment for researchers. However, this also results in limited processing power and memory. Since it runs on ARM processor Raspberry pi cannot run X86 operating systems. Furthermore, alot of softwares are compatible with Raspberry Pi but still a number of softwares run irregularly.

B.1 Bro IDS

Our Aim is to install security solutions on Raspberry Pi and use them for experimentation purpose in our proposed scheme. The Bro IDS was deployed on Raspberry Pi 2 Model B, with 1G RAM and 4 CPU cores, running the Jessie version of Raspbian.

1. During the build process at "sudo make", X509 certificate error was generated.

```
~~~~~~~~~~ src/file_analysis/analyzer/x509
CMakeFiles/plugin-Bro-X509.dir
build.make:119: recipe for target
    'src/file_analysis/analyzer/x509
CMakeFiles/plugin-Bro-X509.dir/X509.cc.o' failed
```

The reason for this is while building Bro on Debian 9, OpenSSL library after version 1.0 is incompatible. So to successfully build Bro install libssl1.0-dev instead of libssl-dev. To solve this error run the following commands:

```
sudo apt-get remove libssl-dev -y
sudo apt-get install libssl1.0-dev -y
```

2. During the build process at "sudo make", the linking process is stopped.

```
[ 92%] Linking CXX executable bro
collect2: fatal error: ld terminated with signal 9 [Killed]
compilation terminated.
src/CMakeFiles/bro.dir/build.make:3166: recipe for target
    'src/bro' failed
make[3]: *** [src/bro] Error 1
make[3]: *** Deleting file 'src/bro'
make[3]: Leaving directory '/home/pi/bro-2.5.4/build'
CMakeFiles/Makefile2:990: recipe for target
    'src/CMakeFiles/bro.dir/all' failed
make[2]: *** [src/CMakeFiles/bro.dir/all] Error 2
make[2]: Leaving directory '/home/pi/bro-2.5.4/build'
Makefile:149: recipe for target 'all' failed make[1]: ***
[all] Error 2 make[1]: Leaving directory
    '/home/pi/bro-2.5.4/build'
Makefile:15: recipe for target 'all' failed make: ***
[all] Error 2
```

The cause of this error is limited RAM of 1 GB in Raspberry Pi model B. To solve this issue linux swap can be used. In Raspbian OS dphys-swapfile is used, which is a swap-file based solution instead of the "standard" swap-partition based solution. By default the swap size defined is 100 Mb. To solve this error run the following commands:

```
1. Stop the Swap
sudo /etc/init.d/dphys-swapfile stop
```

```
2. Modify the size of the Swap. For a stable environment it
   is recommended to keep the Swap size less than 2048Mb
   (2Gb).
cd /etc
sudo nano dphys-swapfile
CONF_SWAPSIZE=1024

3. Start the swap
sudo /etc/init.d/dphys-swapfile start
```

3. During the build process at "sudo make", the CMake file creation process is stopped.

```
Run Build Command:"/usr/bin/make"
   "cmTryCompileExec67693562/fast"
/usr/bin/make -f
   CMakeFiles/cmTryCompileExec67693562.dir/build.make
   CMakeFiles/cmTryCompileExec67693562.dir/build
make[1]: Entering directory
   '/root/Projects/BRO-IDS/bro-2.3.2/build/CMakeFiles/CMakeTmp'
/usr/bin/cmake -E cmake_progress_report
   /root/Projects/BRO-IDS/bro-2.3.2/build/CMakeFiles/CMakeTmp/CMakeFiles
   1
Building C object
   CMakeFiles/cmTryCompileExec67693562.dir/CheckFunctionExists.c.o
/usr/bin/cc -Wall -Wno-unused
   -DCHECK_FUNCTION_EXISTS=pthread_create -o
   CMakeFiles/cmTryCompileExec67693562.dir/CheckFunctionExists.c.o
   -c /usr/share/cmake-3.0/Modules/CheckFunctionExists.c
Linking C executable cmTryCompileExec67693562
/usr/bin/cmake -E cmake_link_script
   CMakeFiles/cmTryCompileExec67693562.dir/link.txt
   --verbose=1
/usr/bin/cc -Wall -Wno-unused
   -DCHECK_FUNCTION_EXISTS=pthread_create
   CMakeFiles/cmTryCompileExec67693562.dir/CheckFunctionExists.c.o
   -o cmTryCompileExec67693562 -rdynamic -lpthreads
/usr/bin/ld: cannot find -lpthreads
collect2: error: ld returned 1 exit status
CMakeFiles/cmTryCompileExec67693562.dir/build.make:88: recipe
   for target 'cmTryCompileExec67693562' failed
make[1]: Leaving directory
```

```
    '/root/Projects/BRO-IDS/bro-2.3.2/build/CMakeFiles/CMakeTmp'
make[1]: *** [cmTryCompileExec67693562] Error 1
Makefile:118: recipe for target
    'cmTryCompileExec67693562/fast' failed
make: *** [cmTryCompileExec67693562/fast] Error 2
```

To solve this issue check that whether all the dependencies have been installed or not.

```
sudo apt-get install bison cmake cmake-data libarchive13
    libbison-dev libpcap-dev libpython-dev libpython2.7-dev
    libssl-dev-1.0 python-dev python2.7-dev swig swig2.0 -y
```

B.2 nDPI

The nDPI is a deep packet inspection tool from ntop. The nDPI was deployed on Raspberry Pi 2 Model B, with 1G RAM and 4 CPU cores, running the Jessie version of Raspbian OS.

1. During the build process at “sudo make”, error during the make command.

```
make[1]:living directory
    ‘/home/raspberrypi/Desktop/ntop-5.0.1/nDPI/src/lib’
make[1]: Entering directory
    ‘/home/raspberrypi/Desktop/ntop-5.0.1/nDPI’
make[1]: Nothing to be done for ‘all-am’.
make[1]: Leaving directory
    ‘/home/raspberrypi/Desktop/ntop-5.0.1/nDPI’
```

By running the “sudo make install” the issue is resolved.

```
test -z "/usr/local/lib" || /bin/mkdir -p "/usr/local/lib"
/bin/bash ../../libtool --mode=install /usr/bin/install -c
libndpi.la
    (http://www.google.com/url?q=http%3A%2F%2Flibndpi.la&sa
=D&sntz=1&usg=AFQjCNGlf087ybeUqubvz5Q0VxyyqFe6-w)
'/usr/local/lib'
libtool: install: /usr/bin/install -c .libs/libndpi.so.1.0.3
/usr/local/lib/libndpi.so.1.0.3
```

```
libtool: install: (cd /usr/local/lib && { ln -s -f
   libndpi.so.1.0.3
libndpi.so.1 || { rm -f libndpi.so.1 && ln -s libndpi.so.1.0.3
libndpi.so.1; }; })
libtool: install: (cd /usr/local/lib && { ln -s -f
   libndpi.so.1.0.3
libndpi.so || { rm -f libndpi.so && ln -s libndpi.so.1.0.3
   libndpi.so;
}; })
libtool: install: /usr/bin/install -c .libs/libndpi.lai
libtool: install: /usr/bin/install -c .libs/libndpi.a
   /usr/local/lib/libndpi.a
libtool: install: chmod 644 /usr/local/lib/libndpi.a
libtool: install: ranlib /usr/local/lib/libndpi.a
libtool: finish:
PATH="/usr/local/sbin:/usr/local/bin:/usr
/sbin:/usr/bin:/sbin:/bin:/usr/games:/sbin"
ldconfig -n /usr/local/lib
```

2. During the build process of “sudo ./configure” command following error arises.

```
make all-recursive
make[1]: Entering directory '/root/nDPI'
Making all in src/lib
make[2]: Entering directory '/root/nDPI/src/lib'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/root/nDPI/src/lib'
Making all in example
make[2]: Entering directory '/root/nDPI/example'
/bin/bash ../libtool --tag=CC --mode=link gcc -pthread -g -O2
   -static -ldl -o ndpiReader ndpiReader.o ndpi_util.o
   ../src/lib/libndpi.la
   /root/PF_RING/userland/libpcap/libpcap.a
   /root/PF_RING/userland/lib/libpfring.a -lnuma -lrt -ldl
   -lrt -ldl -lm
libtool: link: gcc -pthread -g -O2 -o ndpiReader ndpiReader.o
   ndpi_util.o ../src/lib/.libs/libndpi.a
   /root/PF_RING/userland/libpcap/libpcap.a
   /root/PF_RING/userland/lib/libpfring.a -lnuma -lrt -ldl
   -lm -pthread
/usr/bin/ld: cannot find -lnuma
collect2: error: ld returned 1 exit status
```

```
Makefile:393: recipe for target 'ndpiReader' failed
make[2]: *** [ndpiReader] Error 1
make[2]: Leaving directory '/root/nDPI/example'
Makefile:462: recipe for target 'all-recursive' failed
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory '/root/nDPI'
Makefile:371: recipe for target 'all' failed
make: *** [all] Error 2
```

The issue is trying to install to the root directory. Instead of root use the home or any other directory/ folder. This will resolve the issue.

3. During "sudo ./configure" command following error arises.

```
cp ndpi_cpy/lib/third_party/include/.h ndpi_cpy/lib -R;
cp ndpi_cpy/lib/third_party/include/.h ndpi_cpy/include -R;
sed -i "s/^\s*void ndpi_free_flow///void ndpi_free_flow/"
    ndpi_cpy/include/ndpi_api.h;
make -C /lib/modules/4.9.0.minkernel-mid/build M=$PWD;
make[2]: Entering directory
    '/root/ndpi-master/ndpi-netfilter-master/src'
make[2]: *** /lib/modules/4.9.0.minkernel-mid/build: No such
    file or directory. Stop.
make[2]: Leaving directory
    '/root/ndpi-master/ndpi-netfilter-master/src'
Makefile:155: recipe for target 'all' failed
make[1]: *** [all] Error 2
make[1]: Leaving directory
    '/root/ndpi-master/ndpi-netfilter-master/src'
Makefile:5: recipe for target 'all' failed
make: *** [all] Error 2
```

Before starting the "sudo ./autogen" command, install the essentials of the OS.

```
sudo apt-get install build-essentials
```

Bibliography

[1] "Alliance cs (2016) the treacherous 12 cloud computing top threats in 2016." [Online]. Available: https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pdf

[2] "Vortex dds intelligent data sharing platform for business critical internet of things — adlink technology ist." [Online]. Available: http://www.prismtech.com/vortex/

[3] J. E. v. Aken, "Management research based on the paradigm of the design sciences: the quest for field-tested and grounded technological rules," *Journal of management studies*, vol. 41, no. 2, pp. 219–246, 2004.

[4] R. Amin, S. H. Islam, G. Biswas, M. K. Khan, L. Leng, and N. Kumar, "Design of an anonymity-preserving three-factor authenticated key exchange protocol for wireless sensor networks," *Computer Networks*, vol. 101, pp. 42–62, 2016.

[5] S. Arasteh, S. F. Aghili, and H. Mala, "A new lightweight authentication and key agreement protocol for internet of things," in *Information Security and Cryptology (ISCISC), 2016 13th International Iranian Society of Cryptology Conference on.* IEEE, 2016, pp. 52–59.

[6] T. Ashwini and S. G. Mrs Anuradha, "Fog computing to protect real and sensitivity information in cloud."

[7] H. F. Atlam, A. Alenezi, R. J. Walters, G. B. Wills, and J. Daniel, "Developing an adaptive risk-based access control model for the internet of things," 2017, pp. 655–661.

[8] M. Auxilia and K. Raja, "Dynamic access control model for cloud computing." IEEE, 2014, pp. 47–56.

[9] Z. A. Baig, S. M. Sait, and F. Binbeshr, "Controlled access to cloud resources for mitigating economic denial of sustainability (edos) attacks," *Computer Networks*, vol. 97, pp. 31–47, 2016.

[10] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing.* ACM, 2012, pp. 13–16.

[11] R. Charanya and M. Aramudhan, "Survey on access control issues in cloud computing." IEEE, 2016, pp. 1–4.

[12] R. Charanya, M. Aramudhan, K. Mohan, and S. Nithya, "Levels of security issues in cloud computing," *International Journal of Engineering and Technology*, vol. 5, no. 2, pp. 1912–20, 2013.

[13] P. A. J. de Magalhães Costa, "Implementing itil using a workflow tool," 2015.

[14] R. K. Deka, D. K. Bhattacharyya, and J. K. Kalita, "Ddos attacks: Tools, mitigation approaches, and probable impact on private cloud environment," *arXiv preprint arXiv:1710.08628*, 2017.

[15] P. K. Dhillon and S. Kalra, "A lightweight biometrics based remote user authentication scheme for iot services," *Journal of Information Security and Applications*, vol. 34, pp. 255–270, 2017.

[16] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: Preliminary framework and a case study." IEEE, 2014, pp. 16–23.

[17] M. S. Farash, M. Turkanović, S. Kumari, and M. Hölbl, "An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the internet of things environment," *Ad Hoc Networks*, vol. 36, pp. 152–176, 2016.

[18] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication protocols for internet of things: a comprehensive survey," *Security and Communication Networks*, vol. 2017, 2017.

[19] C. GmbH, "Rsa authentication manager." [Online]. Available: https://www.conguide.com/produkte/rsa-security/rsa-authentication-manager.aspx

[20] J. Granjal, E. Monteiro, and J. S. Silva, “Security for the internet of things: a survey of existing protocols and open research issues,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[21] K. Hashizume, D. G. Rosado, E. Fernndez-Medina, and E. B. Fernandez, “An analysis of security issues for cloud computing,” *Journal of Internet Services and Applications*, vol. 4, no. 1, p. 5, 2013.

[22] R. Housley, W. Polk, W. Ford, and D. Solo, “Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile,” Tech. Rep., 2002.

[23] P. Hu, H. Ning, T. Qiu, H. Song, Y. Wang, and X. Yao, “Security and privacy preservation scheme of face identification and resolution framework using fog computing in internet of things,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1143–1155, 2017.

[24] V. C. Hu, D. Ferraiolo, and D. R. Kuhn, *Assessment of access control systems.* US Department of Commerce, National Institute of Standards and Technology, 2006.

[25] V. C. Hu and K. A. Kent, *Guidelines for access control system evaluation metrics.* Citeseer, 2012.

[26] W. Jansen, *Directions in security metrics research.* Diane Publishing, 2010.

[27] A. O. Joseph, J. W. Kathrine, and R. Vijayan, “Cloud security mechanisms for data protection: a survey,” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 9, pp. 81–90, 2014.

[28] G. P. Kanna and V. Vasudevan, “Enhancing the security of user data using the keyword encryption and hybrid cryptographic algorithm in cloud.” IEEE, 2016, pp. 3688–3693.

[29] S. Khairnar and D. Borkar, “Fog computing: A new concept to minimize the attacks and to provide security in cloud computing environment,” *IJRET: International Journal of Research in Engineering and Technology*, vol. 3, no. 06, 2014.

[30] S. Khan, S. Parkinson, and Y. Qin, “Fog computing security: a review of current applications and security solutions,” *Journal of Cloud Computing*, vol. 6, no. 1, p. 19, 2017.

[31] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, “Ddos in the iot: Mirai and other botnets,” *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[32] B. Kuechler and V. Vaishnavi, “Promoting relevance in is research: An informing system for design science research,” *Informing science: The international journal of an emerging transdiscipline*, vol. 14, no. 1, pp. 125–138, 2011.

[33] F. Li, Y. Rahulamathavan, M. Conti, and M. Rajarajan, “Robust access control framework for mobile cloud computing network,” *Computer Communications*, vol. 68, pp. 61–72, 2015.

[34] S. T. March and G. F. Smith, “Design and natural science research on information technology,” *Decision support systems*, vol. 15, no. 4, pp. 251–266, 1995.

[35] C. Marche, L. Atzori, and M. Nitti, “A dataset for performance analysis of the social internet of things,” 07 2018.

[36] F. F. Moghaddam, S. G. Moghaddam, S. Rouzbeh, S. K. Araghi, N. M. Alibeigi, and S. D. Varnosfaderani, “A scalable and efficient user authentication scheme for cloud computing environments.” IEEE, 2014, pp. 508–513.

[37] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, “Secure data sharing and searching at the edge of cloud-assisted internet of things,” *IEEE Cloud Computing*, vol. 4, no. 1, pp. 34–42, 2017.

[38] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, “Security and privacy in fog computing: Challenges,” *IEEE Access*, vol. 5, pp. 19 293–19 304, 2017.

[39] S. Niu, S. Tu, and Y. Huang, “An effective and secure access control system scheme in the cloud,” *Chinese Journal of Electronics*, vol. 24, no. 3, pp. 524–528, 2015.

[40] K. Piirainen, R. A. Gonzalez, and G. Kolfschoten, “Quo vadis, design science?–a survey of literature,” in *International Conference on Design Science Research in Information Systems.* Springer, 2010, pp. 93–108.

[41] L. Popa, M. Yu, S. Y. Ko, S. Ratnasamy, and I. Stoica, “Cloudpolice: taking access control out of the network.” ACM, 2010, p. 7.

[42] S. Quealy, “Openi - open source, web-based framework for integrating applications with cloud-based services and personal cloudlets.” [Online]. Available: http://www.cloudwatchhub.eu/openi-open-source-web-based-framework-integrating-applications-cloud-based-services-and-

[43] F. Y. Rashid, “The dirty dozen: 12 cloud security threats,” *InfoWorld*, 2016.

[44] F. Rehman, S. Akram, and M. A. Shah, “The framework for efficient passphrase-based multifactor authentication in cloud computing.” IEEE, 2016, pp. 37–41.

[45] S. Roschke, F. Cheng, and C. Meinel, “Intrusion detection in the cloud.” IEEE, 2009, pp. 729–734.

[46] R. S. Ross, “Recommended security controls for federal information systems and organizations [includes updates through 9/14/2009],” Tech. Rep., 2009.

[47] S. Ruj, M. Stojmenovic, and A. Nayak, “Decentralized access control with anonymous authentication of data stored in clouds,” *IEEE transactions on parallel and distributed systems*, vol. 25, no. 2, pp. 384–394, 2014.

[48] S. Salonikias, I. Mavridis, and D. Gritzalis, “Access control issues in utilizing fog computing for transport infrastructure.” Springer, 2015, pp. 15–26.

[49] M. Satyanarayanan, “A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets,” *GetMobile: Mobile Computing and Communications*, vol. 18, no. 4, pp. 19–23, 2015.

[50] S. Sciancalepore, G. Piro, G. Boggia, and G. Bianchi, “Public key authentication and key agreement in iot devices with minimal airtime consumption,” *IEEE Embedded Systems Letters*, vol. 9, no. 1, pp. 1–4, 2017.

[51] H. Simon, “The sciences of the artificial. cambridge, ma (p. 252),” 1969.

[52] I. Stojmenovic and S. Wen, “The fog computing paradigm: Scenarios and security issues.” IEEE, 2014, pp. 1–8.

[53] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, “An overview of fog computing and its security issues,” *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2991–3005, 2016.

[54] S. J. Stolfo, M. B. Salem, and A. D. Keromytis, “Fog computing: Mitigating insider data theft attacks in the cloud,” in *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*. IEEE, 2012, pp. 125–128.

[55] H. Takeda, P. Veerkamp, and H. Yoshikawa, “Modeling design process,” *AI magazine*, vol. 11, no. 4, p. 37, 1990.

[56] M. Turkanović, B. Brumen, and M. Hölbl, “A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion,” *Ad Hoc Networks*, vol. 20, pp. 96–112, 2014.

[57] A. R. Vinod, B. S. Sunildatta, K. U. Rani, and P. P. Sasidharan, “Hindering data theft attack through fog computing,” *International Journal of Research in Engineering and Technology*, vol. 3, no. 09, pp. 427–429, 2014.

[58] A. Wadhwa and V. K. Gupta, “Proposed framework with comparative analysis of access control & authentication based security models employed over cloud,” *International Journal of Applied Engineering Research*, vol. 12, no. 24, pp. 15 715–15 722, 2017.

[59] S. Yi, Z. Qin, and Q. Li, “Security and privacy issues of fog computing: A survey.” Springer, 2015, pp. 685–695.

[60] B. Zaghdoudi, H. K.-B. Ayed, and W. Harizi, “Generic access control system for ad hoc mcc and fog computing.” Springer, 2016, pp. 400–415.

[61] P. Zhang, J. K. Liu, F. R. Yu, M. Sookhak, M. H. Au, and X. Luo, “A survey on access control in fog computing,” *IEEE Communications Magazine*, vol. 56, no. 2, pp. 144–149, 2018.

[62] D. Zissis and D. Lekkas, “Addressing cloud computing security issues,” *Future Generation computer systems*, vol. 28, no. 3, pp. 583–592, 2012.