A less attack-prone, Internet deployment of iLanga

Submitted in partial fulfilment
of the requirements of the degree of

BACHELOR OF SCIENCE (HONOURS)

of Rhodes University

Courage Samuel Radu

Grahamstown, South Africa

Abstract

Communication has become very important in the 21st century in a sense that people require communication services to be always available whenever they need them. As such, telephony has developed into a ubiquitous service. iLanga is a Voice over IP (VoIP) telephony system built using open source software with Asterisk Private Branch Exchange (PBX), Kamailio proxy server and MySQL database as main components. It provides its services over the Internet where there is a mixture of both legitimate users and potential attackers. This makes the system prone to security attacks. To make it less prone to attacks, vulnerabilities and threats were identified on the iLanga system and mitigated. A secure version of iLanga was implemented after considering the previous attacks and envisioned threats. A simple mechanism was developed using the perl programming language to quarantine attackers from legitimate users, allowing users to effectively access services without causing denial of service. The system also monitors user passwords and alerts the administrator of any weak passwords via a browser. A step by step guide was developed on how to deploy a less attack-prone iLanga. Today, iLanga system is serving its users in a secure and reliable state.

ACM Computing Classification System Classification

“Thesis classification under the ACM Computing Classification System (1998 version, valid through 2011):”

D.4.6 [Security and Protection]: Authentication

K.6.5 [Security and Protection]: Unauthorized access (e.g., hacking, phreaking)

General-Terms

Security

Acknowledgements

I would like to thank the following individuals and groups for their support and contribution in the completion of this thesis.

The Andrew Mellon Foundation and Rhodes University for their generous scholarship which enabled me to further my studies.

Rhodes University and the Department of Computer Science for the opportunity to complete this Honours Degree.

Mosiuoa Tsietsi for his attentive supervision during the entire year. His contribution is greatly appreciated in suggestions, knowledge and direction in understanding Voice over IP (VoIP) telecommunication systems.

Professor Alfredo Terzoli for his co-supervision during the course of the year. His contribution is greatly appreciated in overall project success.

Convergence Research Group for supporting me with ideas and their team work support.

My Family and Friends for supporting me during my studies. In particular Julia Radu and Wellington Radu for their love, encouragement and everyday support.

I would like to also acknowledge the financial and technical support of Telkom, Tellabs, Stortech, Eastel, Bright Ideas Project 39 and THRIP through the Telkom Centre of Excellence in the Department of Computer Science at Rhodes University.

Contents

1 Introduction **1**

1.1 Problem statement 1

1.2 Project goals 2

1.3 Document structure 2

2 Literature Review **4**

2.1 Session Initiation Protocol (SIP) 4

2.2 Password Security 5

2.2.1 Attacks on password security 6

2.2.2 Guidelines for creating good passwords 6

2.3 Brute force attacks 7

2.3.1 Ways to mitigate brute force attacks 8

2.4 Denial of Service (DoS) and Distributed Denial-of-Service (DDoS) attacks . 8

2.4.1 DoS attack 8

2.4.2 DDoS attack 9

2.4.3 Ways to mitigate DoS and DDoS 9

2.5 Toll fraud 9

2.5.1 Ways to mitigate toll fraud 10
2.6 Eavesdropping 10
2.6.1 Ways to mitigate eavesdropping 10
2.7 Spam over Internet Telephony (SPIT) 10
2.7.1 Ways to mitigate Spam over the Internet 11
2.8 Secure Shell (SSH) authentication 11
2.9 Firewalling 11
2.10 System Hardening 12
2.11 Summary 13

3 iLanga System 14

3.1 Introduction 14
3.2 High level system architecture 14
3.3 Low level system architecture 15
3.4 Summary 17

4 Security Analysis and Implementation 18

4.1 Methodology 18
4.2 A Secure Asterisk Installation 20
4.2.1 Run Asterisk as a non-root 20
4.2.2 Changing the default port 20
4.3 Public Key Authentication for SSH login 21
4.4 Securing User Passwords 22
4.5 Securing User Accounts 23

4.6 Dialplan Security 23

4.6.1 Default context 24

4.6.2 Dialplan injection 24

4.7 Monitoring suspicious events 25

4.7.1 Banning offending IP addresses 25

4.8 The less attack-prone iLanga 28

4.9 Summary 30

5 Experiments and Discussion 31

5.1 Introduction 31

5.2 Service discovery on the server and an IP phone 32

5.2.1 UDP scan 32

5.2.2 TCP scan 32

5.2.3 IP phone scan 33

5.3 Experiments on Asterisk 34

5.3.1 Enumeration attempt with variable `alwaysauthreject=no` 34

5.3.2 Enumeration attempt with variable `alwaysauthreject=yes` 35

5.4 Enumerating valid usernames on Kamailio server 36

5.5 Summary 38

6 Conclusion and Future Work 39

6.1 Summary of work 39

6.2 Future Work 40

6.2.1 Enabling password aging 40

6.2.2 Modifying Kamailio server responses 40

6.2.3 Monitoring Call Detail Records (CDR) 41

6.2.4 Enabling a port knocking mechanism 41

6.3 Summary 42

A How to install Asterisk as non-root 46

B How to configure public key authentication 48

C Perl Script that blocks offending IP Addresses 50

D A concise guide on how to secure iLanga 53

List of Figures

2.1 SIP setup Source: [1] 5

2.2 Packet filtering process. Source: [2] 12

3.1 iLanga system overview 15

3.2 iLanga low level architecture 16

4.1 Iterative approach 19

4.2 Log file 21

4.3 Monitoring password strength 22

4.4 Quarantining offending IP address 26

4.5 Code snippet from the perl script 27

4.6 Summary of blocked IP addresses 28

4.7 The less attack iLanga 29

5.1 UDP scan 32

5.2 TCP scan 33

5.3 IP phone scan 33

5.4 403 Forbidden - Server Response 36

5.5 401 Unauthorised - Server Response 37

Chapter 1

Introduction

In our modern society telephony has developed into an ubiquitous service. People communicate anytime and anywhere via telephony. As such, telephony has moved to the Internet where everyone can easily access it.

Researchers at Rhodes University developed a VoIP telephony system named iLanga [3]. This system is essentially a complete, cost effective, computer-based Private Branch Exchange (PBX) that enables registered users, both on campus and on the open Internet, to communicate. As a VoIP system, iLanga has a rich set of features that includes interactive voice response, call conferencing, music on hold, support for multiple devices for a single user and many more.

As an open source system designed to facilitate service delivery over the Internet, iLanga is subject to a mixture of both legitimate users and attackers. This makes security a very important issue to consider. In this project, we will carefully look into how security can be considered and be prioritised, in order to deploy a reliable communication system.

1.1 Problem statement

In the past, iLanga has been compromised and at the beginning of this project it was shut down to avoid further exploitation rendering it unavailable to its users. The problem can be largley attributed to malicious elements on the Internet. The most prevalent threats to VoIP deployments today are the same security threats that exist in traditional data networks [4, 5]. Among these threats are brute force attacks, Denial of Service (DoS) and Distributed Denial of Service (DDoS). Brute force attack is a technique whereby an attacker simply guesses username and password until he finds a combination that works. DoS and DDoS attack are characterised by an attempt to prevent the legitimate use of a service. Additionally, in the telephony world, there is a new threat known as Spam over Internet Telephony (SPIT) which refers to unsolicited bulk calls or instant messages [6].

1.2 Project goals

The primary goal of this project was to secure iLanga from attacks such as those that have been experienced in the past and to identify other potential threats to it. We tried to mitigate these threats by considering the traditional aspects of security, that is, Confidentiality, Integrity and Availability (CIA). Confidentiality ensures that only authenticated users can use the system and that their information is not disclosed to unauthorised individuals. Integrity ensures that the system behaviour and data are not modified without being detected. Availability ensures that the telephony system is always available for use by legitimate users when they need to communicate.

The secondary goal was to develop a guide with best security practices that will apply to iLanga and similar VoIP systems. This guide will help administrators to carefully consider security when deploying such systems.

1.3 Document structure

The remainder of this thesis is organised into the following chapters:

Chapter 2 discusses possible attacks on VoIP systems and suggested solutions.

Chapter 3 describes the high and low level architectures of the iLanga system.

Chapter 4 discusses the security threats identified on the iLanga system and the counter-measures that were implemented.

Chapter 5 describes experiments that were carried out, followed by a discussion on the results.

Chapter 6 presents a summary of this thesis and suggests areas of future work.

Chapter 2

Literature Review

This chapter reviews some important concepts in telecommunications that will help the reader undestand the context of our work. It also discusses some security threats that affect VoIP systems and possible countermeasures to mitigate them.

2.1 Session Initiation Protocol (SIP)

SIP is a protocol for initiating, modifying, and terminating interactive multimedia sessions [7, 8]. This protocol is not used for media transport but rather for session setup. It differs from other protocols like H.323, which specifies all aspects of signalling, media, features, services, and session control [7]. The primary function of SIP is session initiation even though it has other important functions such as user location, and user availability [8].

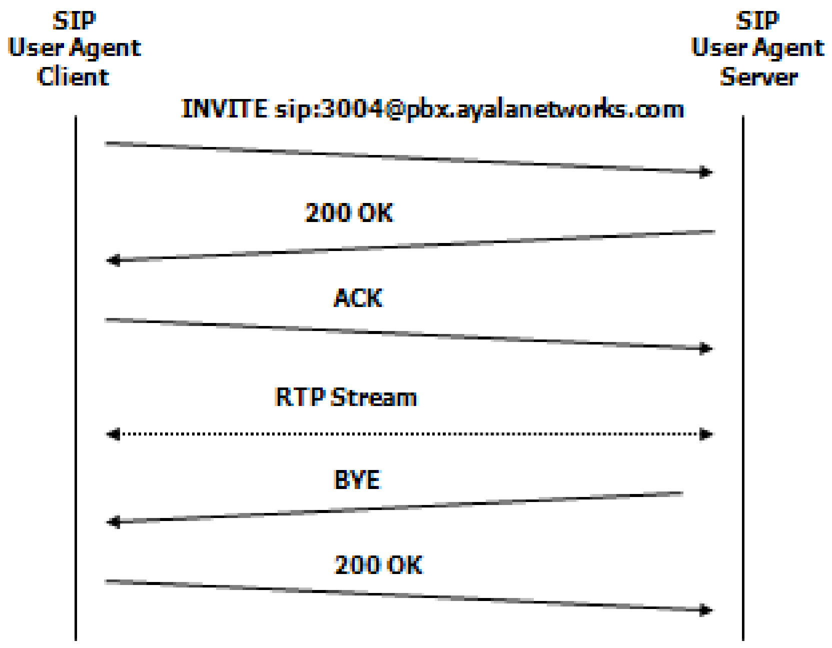


Figure 2.1: SIP setup Source: [1]

Figure 2.1 shows a SIP User Agent Client making an INVITE request to a SIP User Agent Server. The server response 200 OK shows that user 3004 is available and has answered. After acknowledgement RTP handles the media. This is a demonstration of a simple SIP call initiation. After communication is done, the session is terminated by a BYE method.

2.2 Password Security

Passwords are used to verify a user's identity. Therefore, passwords used in authentication processes of any critical system should be strong to avoid them being easily cracked [9]. Generally, the most important part of securing computer systems is securing user passwords [9]. Users are generally not good at creating strong passwords, especially if there are no guidelines on how to create them. They tend to pick short passwords that are easy to remember. Such kinds of passwords are easy targets for hackers. As mentioned in [9] the entire system's security can be endangered by one weak password within that system.

2.2.1 Attacks on password security

- According to Singh [10], the following are known methods of attack on password security.

  1. Observation - an intruder can watch while a password is being typed.
  2. Eavesdropping - an intruder can record the interaction between the user and the system trying to determine the password.
  3. Cryptanalysis - an intruder copies a password file and analyses it.
  4. Password search - a program can be used to carry out an exhaustive search for a password of a given length.

2.2.2 Guidelines for creating good passwords

A case study on Turkish users showed that if users are to create their passwords without any restrictions on the characteristics of key words or without any guidelines for strong characteristics, they pick the ones that will be easily remembered and are short [9]. Such passwords are then easy targets for password crackers or system hackers [11, 9].

- Below are guidelines for users in helping them to create strong passwords [12] :

  1. Do not use only words such as “john” or numbers such as “1234” or dictionary words such as “jacaranda”.
  2. Do not use words in foreign languages because password cracking programs check against word lists that encompass dictionaries of many languages.
  3. Do not use the same password for all machines. If one machine is compromised then other machines are safe.
  4. Do not write down your password. The best way is to memorise it.
  5. Do not use personal information when creating a password. If an attacker knows your identity, then your password can be easily deduced. Avoid using your date of birth or your name, etc
  6. Do not invert recognisable words. Inverting a bad password will never make it secure. An example nauj which is juan in inverted form.

In order to enhance password strength, the following should be included: at least the password should be more than 12 characters, a mixture of upper and lower case characters, digits between 0-9, and special characters such as $, #,&,@.

2.3 Brute force attacks

In a brute force attack, the attacker simply tries to guess username and password combinations until he finds one that works. Weak passwords are easily guessed. Automated tools can be used for brute force attacks that can make thousands of requests per minute with credentials generated from a large list of possible values [13]. The large lists can be online dictionaries or text files with many character combinations.

According to Federal Bureau of Investigation (FBI) Situational Intelligence Report [14] the following two cases were reported:

1. *"In March 2009, the Charlotte Division was notified of an intrusion into a VoIP server located at an undisclosed corporation in Greenville, South Carolina. It was determined that the intruder, using a Romanian based IP address, first conducted a port scan and determined port 5060 was utilized on the compromised server. Port 5060 is the standard port used for Session Initiation Protocol (SIP). SIP is responsible for the setup, modification, and termination of sessions in an IP-based network and is typically the protocol used for VoIP servers. Then the hacker conducted a brute force attack and was able to crack the passwords to two extensions on the VoIP server due to weak passwords. The logs show several password attempts per second, indicating a script was used by the hacker. The hacker then proceeded to make 1,376 calls from the compromised phone extensions attempting to trick victims into providing their bank account information."*

2. *"In February 2009, a non-profit organization located in Charlotte, North Carolina, experienced a computer intrusion into their VoIP server. A review of the server logs revealed IP addresses resolving to France and Florida as being responsible for the intrusion. The intrusion took place through port 5060 and compromised SIP on a server running Trixbox Community Edition. After gaining access, the hackers made approximately 1,850 calls from the compromised system. The calls were made to customers of small regional banks soliciting credit card information via touchtone phone. After victims provided their account information, "money mules" across the country made ATM withdraws using the compromised accounts and sent a portion of the proceeds to Romania."*

In the report analysis of the two cases it was noted that:

- Both attacks were made through port 5060.

- The intruders set up additional extensions on the compromised VoIP servers.
- They then notified victims of a problem with their financial accounts through automated phone calls or mass text messages to cell phones.

Overrally, this shows the importance of using a different port number.

2.3.1 Ways to mitigate brute force attacks

Blake's paper [5] proposed the use of intrusion detection and other monitoring tools is the best way to detect brute force attacks. He added that checking logs for irregularities such as multiple log-on attempts. According to Digium the company that created Asterisk PBX system [15] fail2ban can be used as a reactive way to prevent brute force attempts.

2.4 Denial of Service (DoS) and Distributed Denial-of-Service (DDoS) attacks

2.4.1 DoS attack

A DoS attack is characterised by an explicit attempt to prevent the legitimate use of a service [16, 17]. A network based DoS attack can be achieved in two basic attacks methods [17] :

1. By consuming available resources
2. By bringing the system to a faulty state

These two methods can be combined to consume the available resources by bringing several system processes into a state which needs a lot of resources. Limiting the availability by bringing the system into a faulty state requires compromising either the whole system or only a particular process [17].

A DoS attack by flooding creates resource exhaustion, long term busy signals, and force disconnections of in-session calls. Additionally, brute force attacks cause CPU depletion thus causing a DoS condition.

2.4.2 DDoS attack

A distributed denial of service attack uses multiple hosts to prevent legitimate users from using a service [13]. A DDoS attack is achieved the same way as a DoS attack but the difference is that in a DDoS attack there are multiple hosts. Attackers recruit multiple hosts by:

1. Automatically scanning remote machines, looking for security holes [13]. A discovered vulnerability is then exploited with attack code.
2. Distributing attack software by E-mail attachments and other digital means.

2.4.3 Ways to mitigate DoS and DDoS

In [13] the following systems are suggested for use:

1. Applications that download and install security patches
2. Firewall systems
3. Intrusion detection systems
4. Worm defense systems

The above mentioned systems should enable the victim to endure attack attempts without denying service to legitimate clients. This is done either by enforcing policies for resource consumption or by ensuring that abundant resources exist so that legitimate clients will not be affected by the attack.

2.5 Toll fraud

Toll fraud refers to unauthorised access and use of a VoIP network which involves the establishment of toll bearing calls, especially to international toll numbers [5]. It is a serious threat because it utilises the organisations bandwidth and also incurs heavy costs.

2.5.1 Ways to mitigate toll fraud

Blake's paper [5] mentioned that checking VoIP logs can bring to light irregularities such as international calls made at odd hours.

2.6 Eavesdropping

Eavesdropping on VoIP networks or calls takes place when unauthorised third parties monitor call signal packets. By eavesdropping, third parties can learn user names, passwords, and phone numbers, thereby gaining control over dial plans, voicemail, call forwarding, and billing information [5]. More importantly, third parties may also gain access to confidential business and personal information by eavesdropping on actual VoIP conversations.

2.6.1 Ways to mitigate eavesdropping

Virtual Local Area Network (VLAN) can be used to protect conversations from being eavesdropped [5]. This is because a VLAN is a closed loop of servers or computers that does not allow any other computer access to its network or facilities.

2.7 Spam over Internet Telephony (SPIT)

Spam is well known from the email paradigm. In general it refers to any unsolicited communication [6]. However, in telephony, as identified in [6] three different forms of SPIT exist:

1. Call SPIT, which is defined as a bulk unsolicited set of session initiation attempts in order to establish a multimedia session.
2. Instant Message SPIT, which is defined as a bulk unsolicited set of instant messages.
3. Presence SPIT, which is defined as a bulk unsolicited set of presence requests in order the initiator of SPIT to become a member of the address book of a user or potentially of multiples users.

2.7.1 Ways to mitigate Spam over the Internet

SPIT can be mitigated by blacklisting and whitelisting users especially if it is call SPIT [18]. If it is instant message SPIT a proxy can be used to open the message and see the contents and determine if it is SPIT or not before forwarding to the end users.

2.8 Secure Shell (SSH) authentication

According to Ubuntu documentation [19] use of public key authentication instead of passwords especially for SSH servers that are visible on Internet is more secure. With public key authentication, every computer has a public and a private key. As further stated, key-based authentication has several advantages over password authentication, for example the key values are significantly more difficult to brute force attack [19]. SSH can use either Rivest-Shamir-Adleman (RSA) or Digital Signature Algorithm (DSA) keys of which RSA is recommended over DSA because DSA has been seen as less secure in recent years [19].

2.9 Firewalling

By instituting a firewall [20], unauthorised access can be prevented to services at the network level before an attacker is given the chance to exploit them. Netfilter and IPtables are integrated in the Linux kernel since 2.4.x series and later series of kernels [2].

The firewall makes packet filtering process by following specified rules. There is flexibility to add, edit and remove the rules. The rules are grouped together into chains. The kernel defines three chains by default (INPUT, FORWARD, OUTPUT chains), but new chains can be specified and linked to the predefined chains [2, 20]. Figure 2.2 shows the packet filtering process.

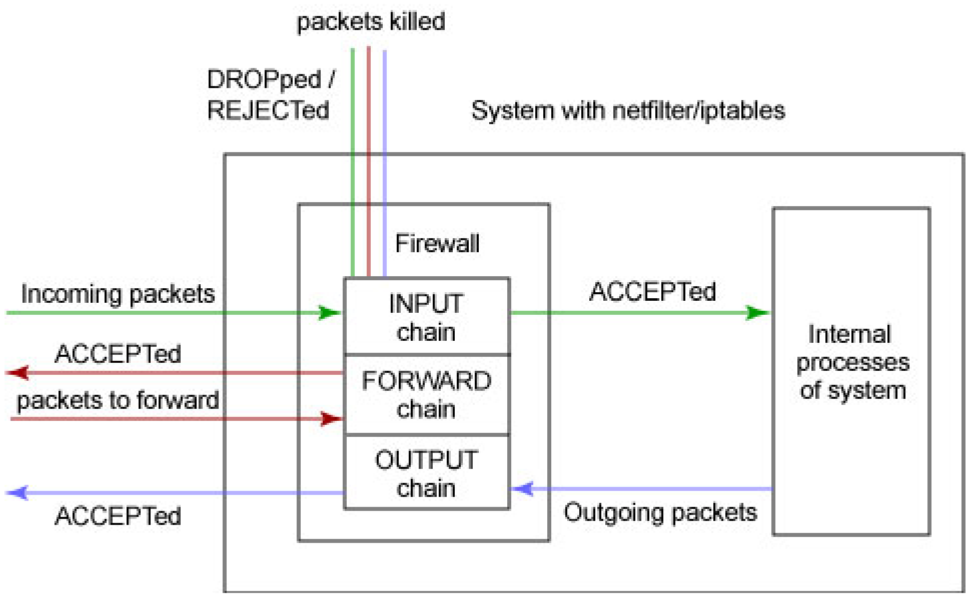


Figure 2.2: Packet filtering process. Source: [2]

- In general, firewalls as stated for example in [2, 20] offer the following three core benefits especially to administrators;

  1. Help understand where the threats to your system are coming from
  2. It gives administrator total control over firewall configuration and packet filtering.
  3. Administrator can make own rules that will suit specific needs

2.10 System Hardening

A very important step in securing a Linux system is to determine the primary function or role of the server [11]. When the primary function is determined, it is then important to know what is on the system. A fresh installation of a system result in some unused software packages and these should be removed. For instance there is no point of keeping Apache server if it is not used. Removing unnecessary software packages means that there are fewer packages to update when security alerts are released. Additionally, unnecessary packages can be potential vectors for attackers especially when they reside without being updated.

2.11 Summary

This chapter described the main attacks that iLanga is likely to suffer from when deployed. It described how certain attacks are achieved and possible ways to mitigate them. These attacks include brute force attacks, DoS and DDoS, eavesdropping and SPIT. The ways to mitigate have been discussed as preventive and reactive measures. Preventive measures ensure that the system is secure by the use of strong passwords, use of public key authentication, use of firewall, etc. Reactive measures include the use of Intrusion Detection Systems.

Chapter 3

iLanga System

3.1 Introduction

This chapter describes the iLanga system and its software components. The hardware components are not of major interest so they are not discussed. This is because the Asterisk PBX server can run on any PC (in its minimal configuration) [3].

3.2 High level system architecture

Figure 3.1 shows a high level overview architecture of iLanga. This is a generic overview of what the system looks like. The server is the main hardware component and has Asterisk installed on it. Asterisk is software that turns an ordinary computer into a communications server [21]. It receives requests from users and processes them accordingly. Asterisk server is connected to the public Internet via campus network. Moreover, the server is connected to Telkom (the South African fixed line operator) via Integrated Service Digital Network (ISDN) connection. The server acts as a gateway between the PSTN and the IP based network. Two more servers are also installed and these are Kamailio proxy server and MySQL database server.

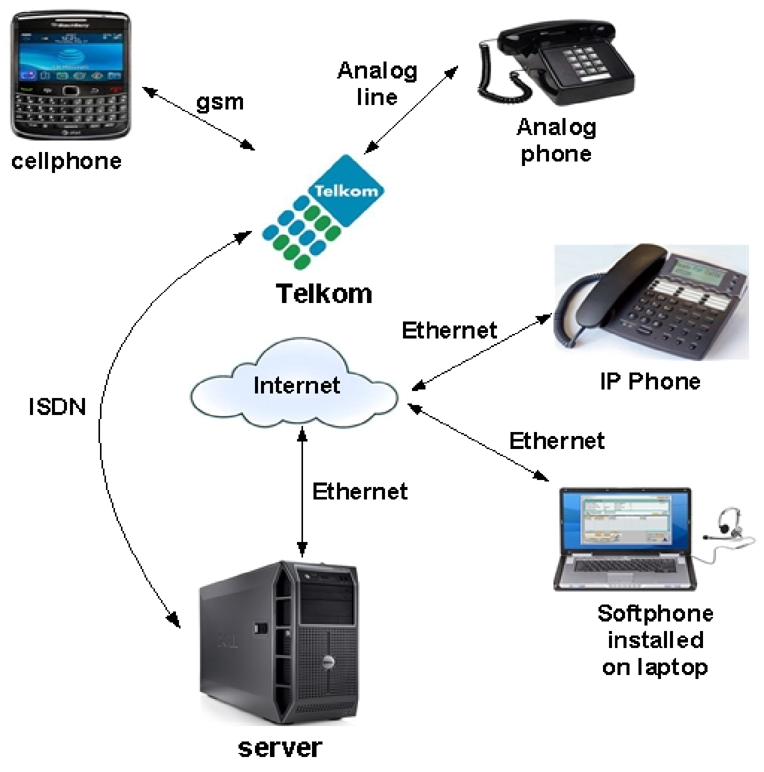


Figure 3.1: iLanga system overview

Kamailio proxy server serves three main purposes 1) it allows parallel forking 2) it is used for load balancing and 3) it is used for authentication. Parallel forking allows a single user to use the same extension on different devices in different locations. If a caller dials that extension, the proxy server forks the request and rings all the devices. When one device is picked, the rest will be terminated. Asterisk lacks this functionality so it is an advantage to have Kamailio to serve this purpose in the system.

MySQL server is used to store user information. Any other database can be used for this purpose. User information includes usernames, passwords, email addresses and so on. In iLanga, MySQL is an ideal database for storing user and information as well as billing information because of its flexibility and scalability.

3.3 Low level system architecture

Figure 3.2 shows a low level iLanga system overview.

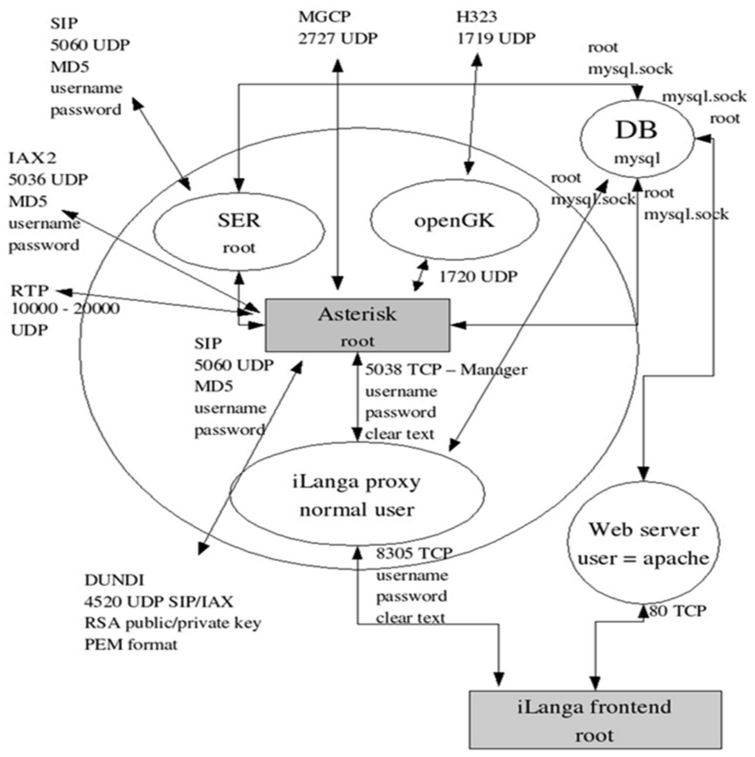


Figure 3.2: iLanga low level architecture

The main component is Asterisk and is found at the centre. MySQL database is at the top right corner. Kamailio, which was drawn from previous project known as Sip Express Router (SER) is just above Asterisk. Asterisk, MySQL and Kamailio servers run as super users. The three servers are all interlinked. iLanga has a frontend that allows users to register and manage their accounts. The frontend is deployed on an Apache web server. OpenGK is no longer in use, but was used as a gateway between asterisk and H.323 endpoints.

The protocols that are of interest are the SIP and Real-Time Transport protocol (RTP). SIP is responsible for the setup, modification, and termination of sessions in an IP based network while RTP is responsible for transporting the media. These protocols are well discussed in Chapter 2. Generally, the components are connected to each other in a complex way.

3.4 Summary

The iLanga architecture shows the various components that constitute it. As show, it is made up of open software, therefore, it is a low cost VoIP system that can be deployed at tertiary institutions and small business enterprises. It is made up of open source software. The only hardware extensions required is the ISDN card. The low level architecture shows some complexity in the way which components interfaces with each other. This might cause some security issues as well as management of the system.

Chapter 4

Security Analysis and Implementation

This chapter discusses security threats that have been identified on the iLanga system and possible countermeasures to mitigate them. The final section shows a less attack-prone deployment.

4.1 Methodology

We used an iterative approach to secure the system. The first step was to record versions for each component as part of the preliminary phase. These versions would help to find any security vulnerabilities that have been posted on the websites of that particular component. For instance Asterisk has a recent project known as the Asterisk Project Security Advisory found on www.asterisk.org which releases vulnerabilities discovered and the affected versions.

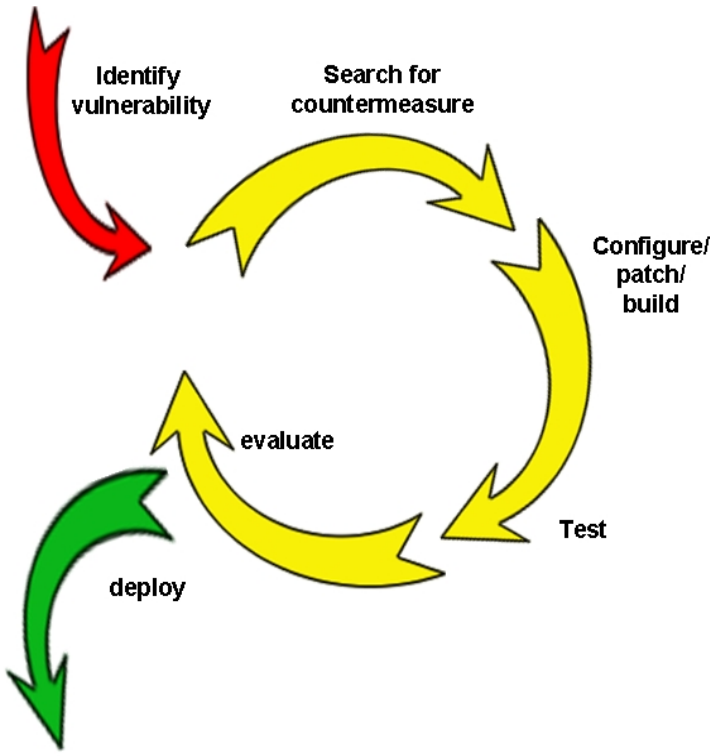


Figure 4.1: Iterative approach

Figure 4.1 shows the representation of the iterative approach we used in mitigating the threats. Step 1 was to identify a vulnerability or threat from the system. It was followed by a search for a countermeasure. The countermeasure might require a configuration or, a patch or perhaps building a new solution. The countermeasure was then applied, and tested. After testing, the countermeasure was the evaluated to see if it is worth deploying. If it was not worth deploying, then another countermeasure was considered. This was an on-going process in the building of the system security.

In order to understand how the system works, we had to replicate the iLanga system on a test server with the following components:

Hardware components:

1. Server: Processor – Intel(R) Core(TM) i7 CPU @2.93Ghz , RAM – 4.00 GB , and System Type – 64-bit Operating System

2. IP Phone: AT-320 H323 v1.39

3. Laptop/Desktop: Any

Software components:

1. Operating System: Ubuntu linux 10.10 maverick
2. Servers: Asterisk v1.6.0.6 , Kamailio v2.0, MySQL v5.5
3. Softphone: Twinkle v1:1.4.2-2build1

The replicated system acted as a test bed where we carried out all the experiments before moving the countermeasures to the production machine.

4.2 A Secure Asterisk Installation

4.2.1 Run Asterisk as a non-root

When installing Asterisk, it is good practice to run it as a non-root [22, 23]. By default, Asterisk is configured to run as root, that is, as a super user. This is useful so that if the Asterisk is compromised, it cannot be used to take over the entire machine.

It is possible to run Asterisk with reduced privileges by adding a new user account on the system. Like any account, a username and password will be provided. When Asterisk is reconfigured to run as a normal user, it will then need permissions to be able to read and write certain files. Some files need to be writable so that Asterisk will be able to append lines for instance registration status, while other files have to be readable for instance the users.conf. A step by step procedure on how to install Asterisk as a non-root is given in Appendix A.

4.2.2 Changing the default port

Port 5060 is the standard port used for SIP signalling. Changing the default port to another arbitrary port will add a smaller layer of protection, as discussed in Chapter 2.

This is because the default port is well known and that can make the attacker's work easy. Changing the default port will thus add an extra layer of security and will force the attacker to work harder in order to determine the properties of the system. This intervention however will require users to alter the server settings on their phones.

4.3 Public Key Authentication for SSH login

Figure 4.2 shows a summary from the actual log file of a Linux server. It shows attempts to crack the root password by intruders. This extract was made in June 2011 from the iLanga production machine before it was shut down to avoid further exploitation. From the extract below, on June 16 2011 at 12:16pm an intruder with IP address 95.141.193.46 made 40 attempts guessing the root password. On June 17 2011 from 02:27am until 02:42am an intruder with IP address 124.205.190.217 made several attempts with different usernames such as prueba, services, droguri, sshadmin, abcs, sshdu, ospite, postgres, Duane, Oscar, VPN, info, gnats, and Rome.

```
From the /var/log/auth.log.1log file June 12 13:17 Failed password for
root from 109.237.214.  6 attempts
June 12 22:59 Failed password for root from 122.225.96.156 6attempts
June 16 12:16 Failed password for root from 95.141.193.46 about 40
attempts Failed password for invalid user test from 95.141.193.46 3
attempts Failed password for invalid user nagios 2 attempts Failed
password for invalid user postgres  2 attempts Failed password for
invalid user oracle 1 attempt
June 17 02:27 until 02:42 Failed password for root from 124.205.190.217 6
attempts Follwed by Single attempts for users prueba, services, droguri,
sshadmin, abcs, sshdu, ospite, postgres,Duane, Oscar, VPN, info, gnats,
Rome
June 17 10:16 Server listening on 0.0.0.0 port 22 Received signal 15
terminating
June 18 17:08 until 17:10 Reverse mapping checking getaddressinfo for
airtelbroadband.in [182.79.254.14] failed POSSIBLE BREAK IN ATTEMPT
Invalid user test from 182.79.254.14
June 19 03:46 until 03:53 Similar attack to June 17 02:27 Invalid user
from 38.110.72.164 with usernames root, eb, nagios, postgres etc
```

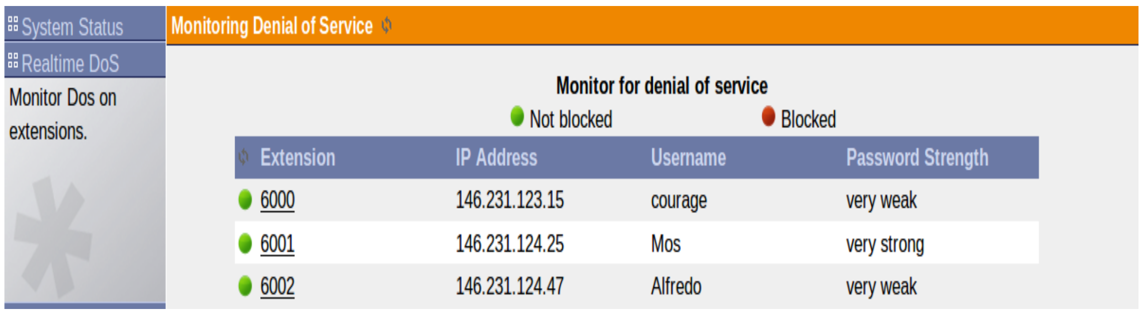
Figure 4.2: Log file

Secure Shell (SSH) is a network protocol used to securely access a remote machine. It allows secure data communication and command execution. SSH uses public key authentication if necessary. Public key authentication is highly recommended since the server is visible over the Internet [19]. The goal is to prevent random username and password guesses before the attacker finds the correct combination. Appendix B show how to setup SSH so that you can securely login without having to provide a password. With public key authentication, private and public keys are generated. The public key is then transferred to the server. The private key will remain on the client side. When logging in, the private key is used instead of providing the password. If the private key matches with the public key, after a methematical computation, then it gets authenticated. The length of the private key is relatively large and thus makes it hard to use brute force methods on it.

4.4 Securing User Passwords

In order to make phone calls, one needs to be a registered user. Registered users are uniquely identified by a username and a password. Asterisk is not programmed to detect weak passwords, thus users end up using weak passwords. This weakness is within the system because there are no guidelines for creating strong passwords that users can follow. Additionally, when the administrator creates a new user account with a weak password, the system does not complain.

To mitigate against this, we designed a perl script that checks the strength of passwords for users and alerts the administrator of any weak passwords via the browser. Figure 4.3 shows the password strength for three registered users. This allows the administrator to always be alert of any weak passwords within the system.



Monitor for denial of service

● Not blocked ● Blocked

	Extension	IP Address	Username	Password Strength
●	6000	146.231.123.15	courage	very weak
●	6001	146.231.124.25	Mos	very strong
●	6002	146.231.124.47	Alfredo	very weak

Figure 4.3: Monitoring password strength

The perl script looks for the following features in a password:

1) Length (at least 12 characters)

2) Lowercase characters (at least two)

3) Uppercase characters (at least two)

4) Digits 0-9 (at least two)

5) Special character eg #,$,ˆ,_,@, and & (at least one)

The system should always have strong passwords that are hard to guess. The features above enhance the strength of passwords.

4.5 Securing User Accounts

Since the Asterisk PBX is pointing to the public Internet, it is most likely to be scanned for valid user accounts. The intruder typically, checks for common usernames and then goes for numbered accounts, since it is common for administrators to name SIP accounts with the same name as the extensions on the PBX. When enumerating usernames, the attacker tries to register a phone using different usernames. A server response will enable the attacker to determine whether the username exists or not.

Setting the variable `alwaysauthreject=yes` in the sip.conf file will resolve this problem. This will prevent the attacker from enumerating the usernames on the server. Additionally, the use of non-numeric usernames for VoIP accounts will make them harder to guess [23].

4.6 Dialplan Security

The Asterisk dialplan is another area where security should be considered important. In Asterisk, a dialplan is the most important part of the Asterisk system [ ]. It defines how Asterisk handles incoming and outgoing calls. The configuration file `extensions.conf` contains the dialplan with all valid extension numbers. A new installation of Asterisk has the `extensions.conf` file installed by default. Modifying this default file by adding new extensions is must be done carefully because some of the dialplan syntax has security risks. One can use the default `extensions.conf` file as a reference. A good approach is to create a new file and populate it with contents. Basically, the dialplan looks like this:

```
[general]
some settings go here
[globals]
global variables go here
[context1]
extension 1, priority 1, application
extension 1, priority 2, application
[context2]
extension 2, priority 1, application
extension 2, priority 2, application
```

4.6.1 Default context

There is a context known as the default context. The default context should be secure. It should not have extensions that can cost the organisation money.

4.6.2 Dialplan injection

A dialplan should be built with great care in order to prevent one of the more recent dialplan vulnerabilities that have been discovered [23, 24]. The channel variable ${EXTEN} is commonly used in the dialplan. If this variable is used with wildcard pattern matches, it can lead to possible string injection vulnerability. The following example shows the use of a wildcard match in a dialplan.

```
 exten => _X.,1,Dial(SIP/${EXTEN})
```

It may be possible for an attacker to craft an INVITE which sends data such as `300&DAHDI/g1/4165551212` which would create an outgoing channel leg that was not originally intended. If evaluated the extension will become

```
exten => _X.,1,Dial(SIP/300&DAHDI/g1/4165551212)
```

If the system has an interface to the PSTN installed and configured, this will cause the call to go out on the number chosen by the attacker, even though the administrator did not grant access to that caller. This will probably cost the organisation.

Administrators or developers should be careful on how foreign data will flow in the system when designing the dialplan. This problem can be solved by filtering string data from external sources. This can be done using the `FILTER()` dialplan function. All incoming context data should be filtered before it starts to flow in the system. Strict pattern matching can be used if the length of the extension is known before hand, for example `exten => _XXXX,1,Dial(SIP/${EXTEN})`.

4.7 Monitoring suspicious events

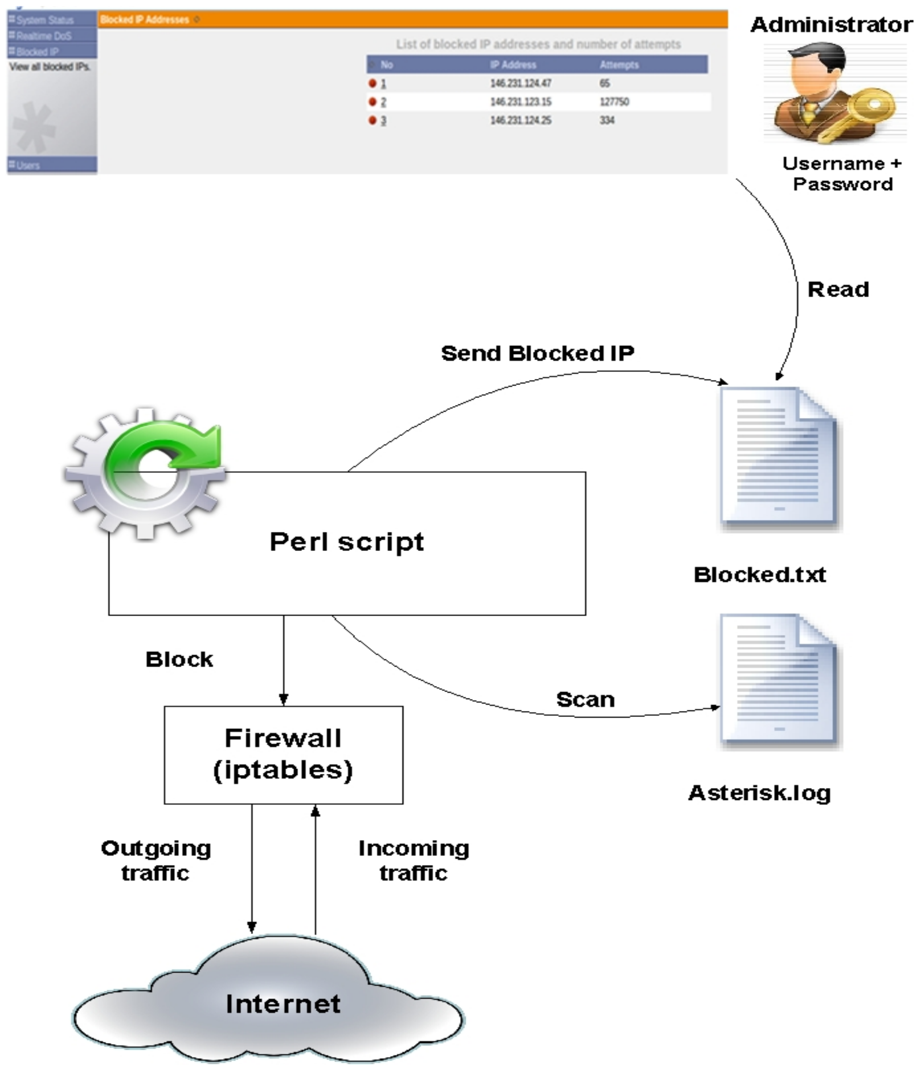
It is highly recommended that an intrusion prevention mechanism be used in order to protect VoIP systems like iLanga. Fail2ban [23] can be used as a basic countermeasure to quarantine offending IP addresses. Fail2ban bans an IP that makes too many failed login attempts. It then updates the firewall to reject that particular IP address.

It is useful to alert the system administrator about the status of blocked IP addresses. Fail2ban has a useful functionality that sends a mail everytime it bans an IP address. Nevertheless, this functionality requires a Mail Transfer Agent (MTA) like Postfix [23] to be installed in order to work. While it is beneficial to receive a blocked IP address this has some weaknesses. Firstly, the administrator might receive thousands of blocked IP addresses in a day which does not give extra information like the number of attempts made by each particular IP address. Secondly, there is no way the administrator can tell if the blocked IP is being used by a legitimate user. Thirdly, the option `ignoreip = ,` where one can specify IP addresses that should never be blocked especially the server IP can be a risk if the attacker spoofs that server IP address.

4.7.1 Banning offending IP addresses

We developed a perl script that behaves the same way as Fail2ban in order to suite our domain problem. This script is an extension of a perl script originally created by an Asterisk consulting company Teamforest [25]. Asterisk PBX comes with an inbuilt mini web server that is enabled so that the administrator will be able to view the blocked IP addresses via the browser. An administrator will have easy access since the browser is accessible everywhere.

The program scans the asterisk.log log file, and looks for particular patterns. In other words, the program has a predefined pattern to watch, otherwise it will not know when to block or not. Figure 4.5 shows what the program watches for. Line 19 shows that

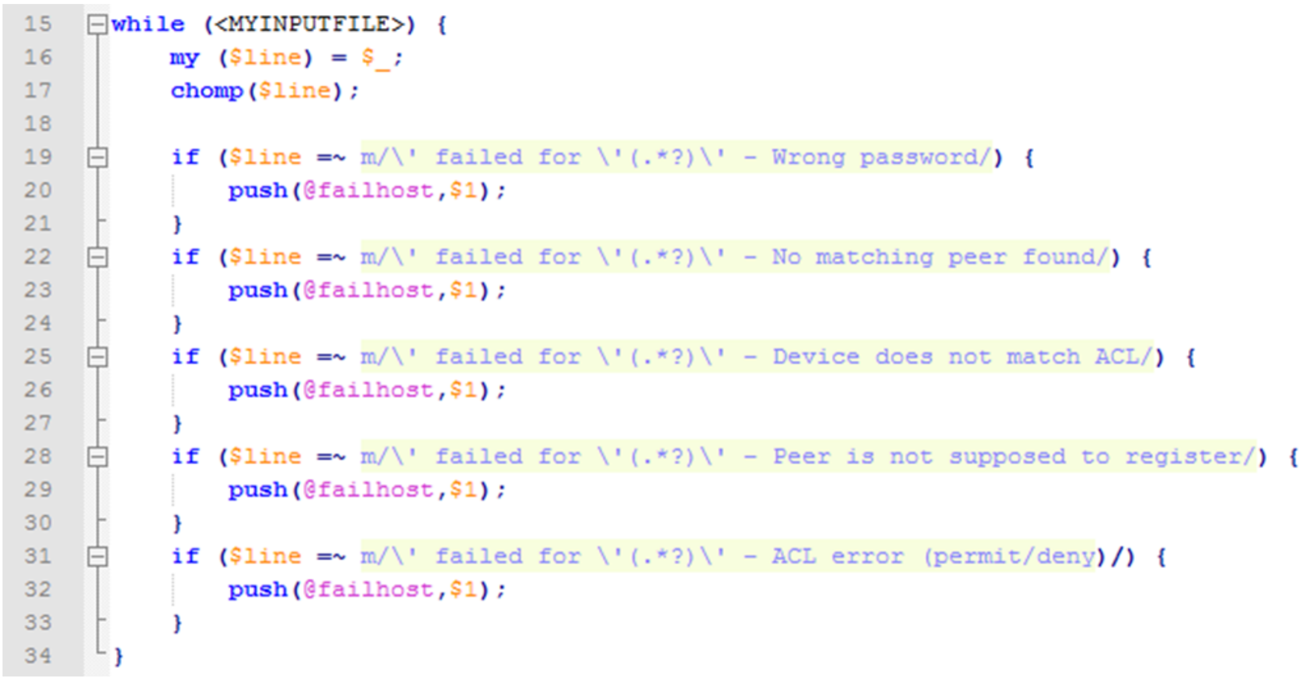


List of blocked IP addresses and number of attempts

No	IP Address	Attempts
1	146.231.124.47	65
2	146.231.123.15	127750
3	146.231.124.25	334

Figure 4.4: Quarantining offending IP address

if the program has to watch for “Wrong password”, line 22 shows that it has to watch for “No matching peer found” and so on. The program will keep track of the number of times such patterns appear for each IP. If the counter exceeds six, it sends an action to the firewall, thereby blocking the IP address. For the full program, refer to appendix C.



```perl
while (<MYINPUTFILE>) {
    my ($line) = $_;
    chomp($line);

    if ($line =~ m/\' failed for \'(.*?)\' - Wrong password/) {
        push(@failhost,$1);
    }
    if ($line =~ m/\' failed for \'(.*?)\' - No matching peer found/) {
        push(@failhost,$1);
    }
    if ($line =~ m/\' failed for \'(.*?)\' - Device does not match ACL/) {
        push(@failhost,$1);
    }
    if ($line =~ m/\' failed for \'(.*?)\' - Peer is not supposed to register/) {
        push(@failhost,$1);
    }
    if ($line =~ m/\' failed for \'(.*?)\' - ACL error (permit/deny)/) {
        push(@failhost,$1);
    }
}
```

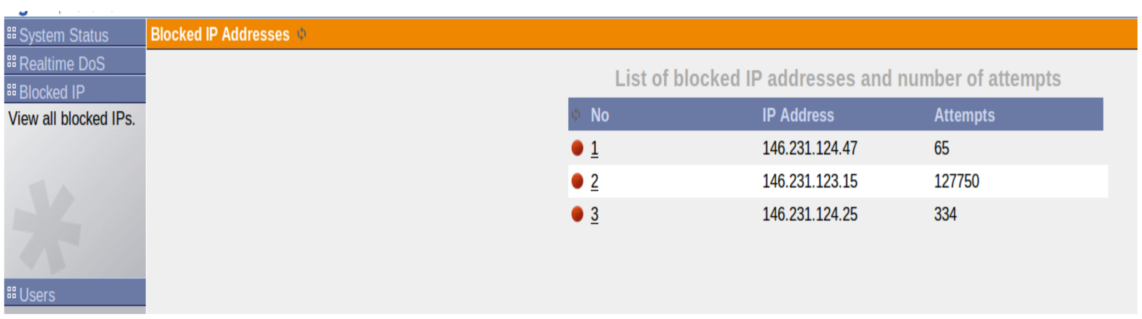
Figure 4.5: Code snippet from the perl script

The perl script is set in the crontab so that it can run in the background. Crontab is a Linux utility that allows tasks to be automatically run in the background of the system at regular intervals. The minute, hour, day, month and weekday parameters which your program has to be executed should be specified. To add a cron job, you edit the crontab file. Once you save the file, the crontab is running. The command `#crontab -e` will allow one to edit the contab file. Adding the following line will allow the program to be executed after every minute.

```
*/1 * * * * perl /home/courage/Desktop/check.pl &> /dev/null
```

In this statement `*/1` means execute the program after every one minute. The `perl` means that it is a perl program. The path to the program being executed is given by `/home/courage/Desktop/check.pl`. Every time the job finishes executing, the crontab's way of notifying is sending an email. In order to disable this feature we use `/dev/null`. This essentially writes the email out to nowhere.

An administrator can view this information anywhere via the browser. Figure 4.6 shows the web interface with three blocked IP addresses and the number of attempts made.



List of blocked IP addresses and number of attempts

No	IP Address	Attempts
● 1	146.231.124.47	65
● 2	146.231.123.15	127750
● 3	146.231.124.25	334

Figure 4.6: Summary of blocked IP addresses

4.8 The less attack-prone iLanga

Figure 4.7 shows the less attack-prone iLanga architecture after all implementation.

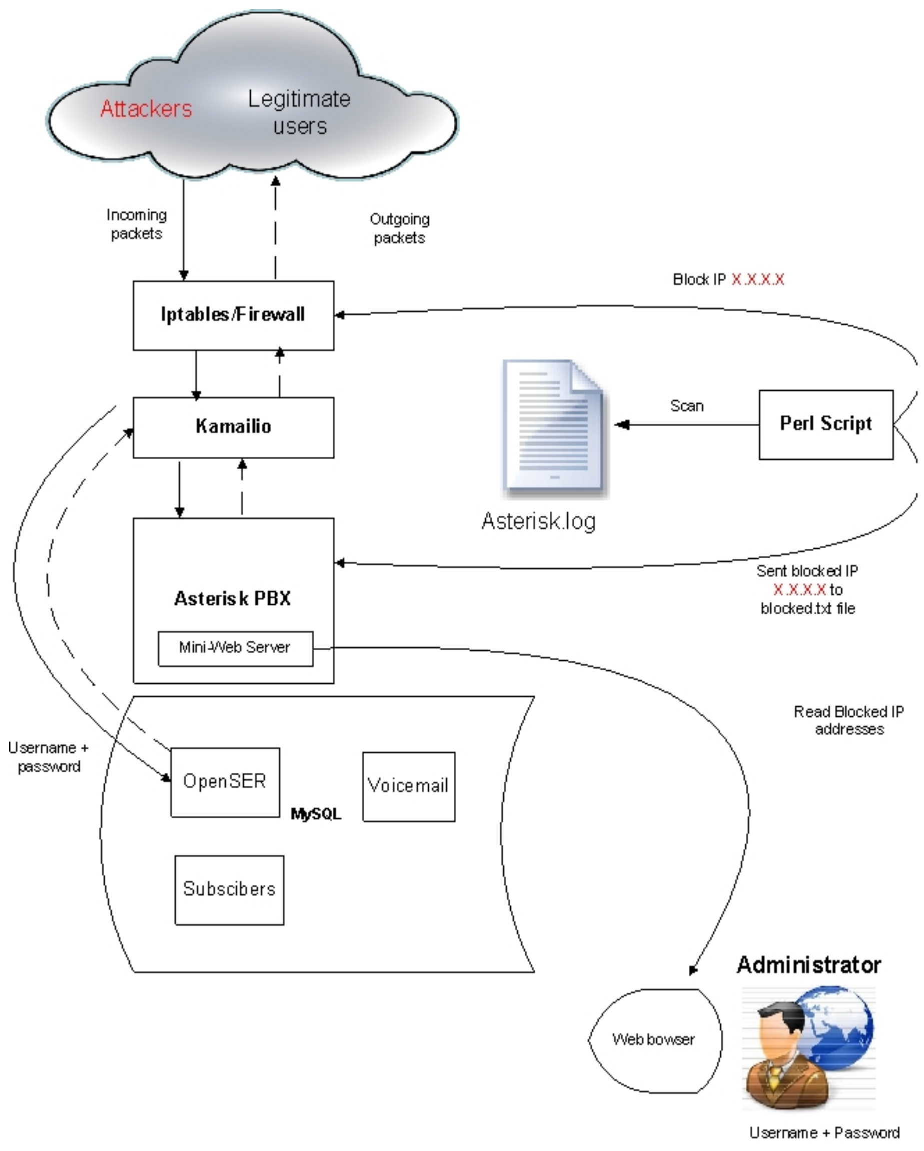


Figure 4.7: The less attack iLanga

The secure implementation shows the use of the inbuilt Linux firewall (iptables) working with the perl script blocking offending IP addresses, the Asterisk PBX server running as a non-root, variable `alwaysauthreject=yes`, the administrator can SSH login using private key, all user passwords are strong, a carefully designed diaplan and the administrator can monitor security related information via the browser.

4.9 Summary

Due to the nature of the design of the system, it is prone to attacks that originate from the Internet. The threats identified on the iLanga system are more similar to the security threats that exist in traditional data networks. These threats are not carefully considered by administrators and developers when developing or deploying systems. Some security features are inbuilt but must be enabled in order to protect the system. This chapter includes some basic security features from initial installation of the Asterisk PBX up to a point where a basic system was built to enhance the security.

The most important things to do are to run the Asterisk PBX as a non-root, change the default port 5060, making sure all user passwords are strong, setting the variable `alwaysauthreject = yes,` creating a diaplan with great care, configure public key authentication for SSH login, and quarantining offending IP addresses. The web based user interface eases administration and is accessible everywhere.

Chapter 5

Experiments and Discussion

5.1 Introduction

Information gathering is one of the most powerful tools at the attacker's disposal [4]. This chapter will discuss a few experiments using Nmap for information gathering. Nmap is an open source tool for network exploration and security auditing [26]. Experiments carried out using Nmap are intended to show the amount of information that is visible to the Internet that can be used by an attacker.

This chapter will also discuss the use of SIPvicious suite. SIPvicious is a suite of tools that is used to audit SIP based VoIP systems [27]. It has three tools that are of interest to this work: svmap, svwar, and svcrack. Svmap is a sip scanner that lists SIP devices that are found on a particular IP range. Svwar is used to identify active extensions on a PBX. Svcrack is an online password cracker for a SIP PBX. We used SIPvicious tool to carry an audit on our test bed discussed in Chapter 4.

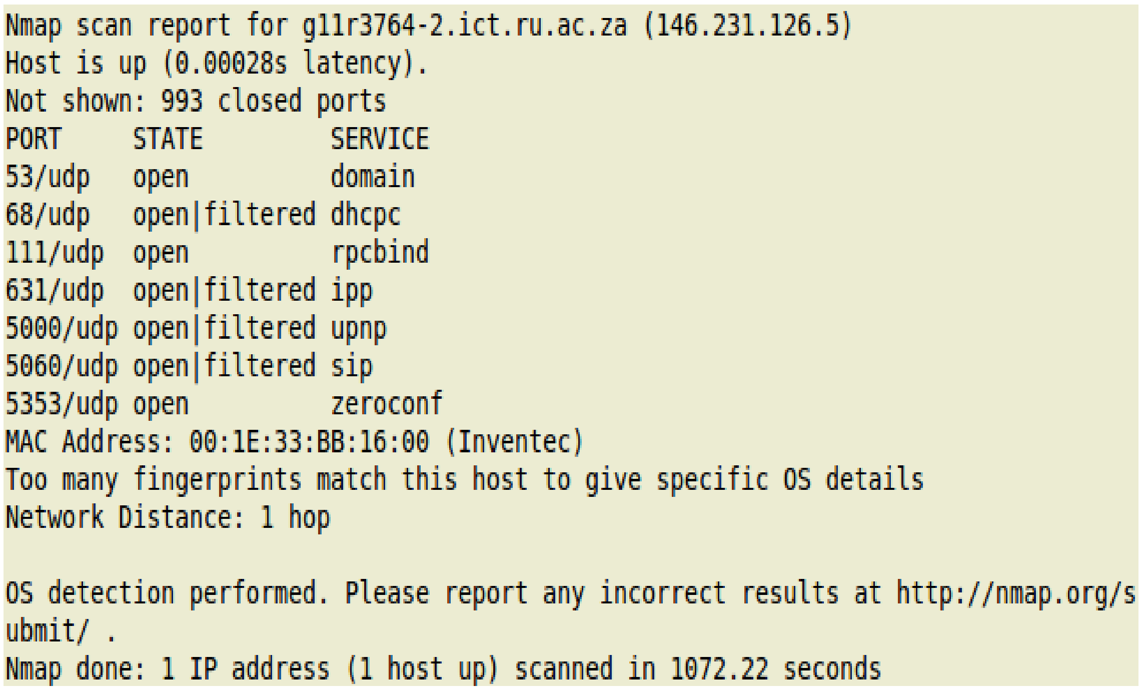
Furthermore, this chapter will discuss the use of the REGISTER method for enumerating usernames by analysing server responses. Twinkle softphone was used to carry out these experiments.

We chose the above tools because they are open source and they suite the goal of our investigation. There are certainly other tools that we could have used.

5.2 Service discovery on the server and an IP phone

5.2.1 UDP scan

The objective of this experiment was to determine the UDP ports that are open and the service they provide.



```
Nmap scan report for g11r3764-2.ict.ru.ac.za (146.231.126.5)
Host is up (0.00028s latency).
Not shown: 993 closed ports
PORT     STATE         SERVICE
53/udp   open          domain
68/udp   open|filtered dhcpc
111/udp  open          rpcbind
631/udp  open|filtered ipp
5000/udp open|filtered upnp
5060/udp open|filtered sip
5353/udp open          zeroconf
MAC Address: 00:1E:33:BB:16:00 (Inventec)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/s
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 1072.22 seconds
```

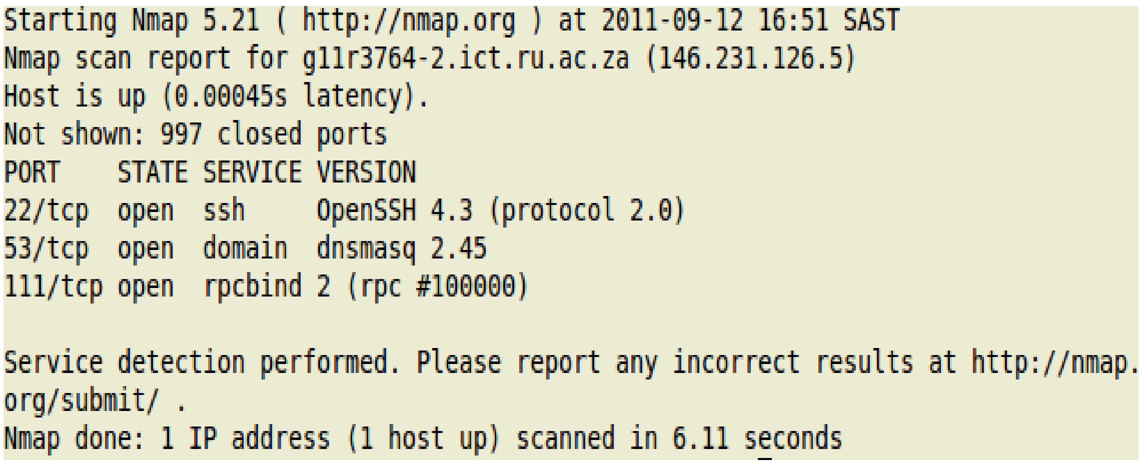
Figure 5.1: UDP scan

Discussion:

Figure 5.1 provides a snapshot of the results obtained from a UDP scan. The results show that port 5060 is `open | filtered` and it provides a SIP service. This port is used by Asterisk. According to Nmap's man page definition, `open | filtered` means that Nmap is unable to determine whether a port is open or filtered and this occurs for scan types in which ports give no response [4].

5.2.2 TCP scan

The objective of this experiment was to determine the TCP ports that are open and the service they provide.



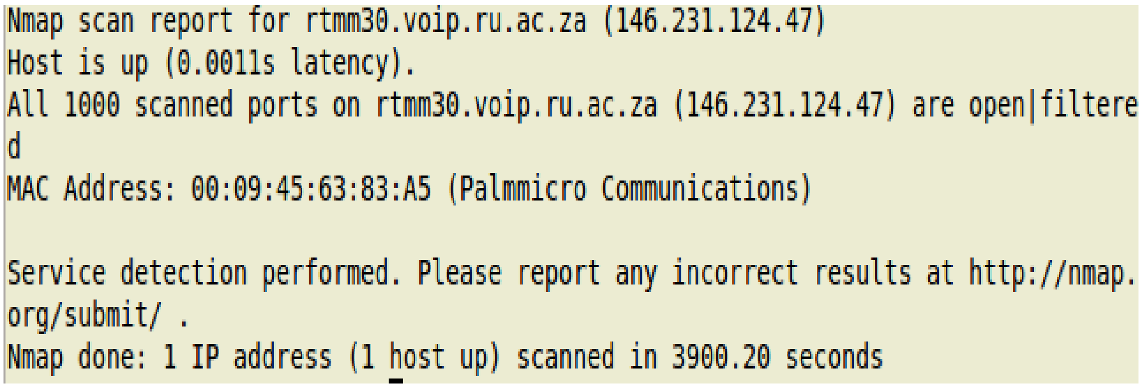
```
Starting Nmap 5.21 ( http://nmap.org ) at 2011-09-12 16:51 SAST
Nmap scan report for g11r3764-2.ict.ru.ac.za (146.231.126.5)
Host is up (0.00045s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 4.3 (protocol 2.0)
53/tcp  open  domain  dnsmasq 2.45
111/tcp open  rpcbind 2 (rpc #100000)

Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.11 seconds
```

Figure 5.2: TCP scan

Discussion:

Figure 5.2 provides a snapshot of the results obtained from a TCP scan. The results show that three ports are open and none of these are used by Asterisk.

5.2.3 IP phone scan



```
Nmap scan report for rtmm30.voip.ru.ac.za (146.231.124.47)
Host is up (0.0011s latency).
All 1000 scanned ports on rtmm30.voip.ru.ac.za (146.231.124.47) are open|filtere
d
MAC Address: 00:09:45:63:83:A5 (Palmmicro Communications)

Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3900.20 seconds
```

Figure 5.3: IP phone scan

Discussion:

Figure 5.3 provides a snapshot of the results obtained from an IP phone scan. The results show that 1000 ports are `open | filtered`. Furthermore, we can see the MAC Address of the phone and the company that created the phone, Palmmicro Communications.

Port scanning will help an attacker to determine which services are running on your server as well as the ports they are listening on. The attacker can further investigate the versions of the applications running a service. For instance, Figure5.2 show that openSSH is version 4.3 . The attacker can use this version to find any vulnerability that could have been reported and exploit it. All open ports should be filtered and unnecessary services should be disabled.

5.3 Experiments on Asterisk

The following experiments will demonstrate the importance of setting the variable `alwaysauthreject=yes`. This prevents the attacker from enumerating valid users on Asterisk [24].

5.3.1 Enumeration attempt with variable `alwaysauthreject=no`

Step 1: Device identification

```
courage@courage-desktop:~/Desktop/sipvicious$ ./svmap.py 146.231.123.15
WARNING:DrinkOrSip:could not bind to 0.0.0.0:5060 - some process might
already be listening on this port.  Listening on port 5061 instead
| SIP Device          | User Agent           | Fingerprint |
-------------------------------------------------------------
| 146.231.123.15:5060 | Asterisk PBX 1.6.0.6 | Asterisk /
Linksys/PAP2T-3.1.15(LS) / Asterisk PBX
```

Step 2: Scanning for valid users

```
courage@courage-desktop:~/Desktop/sipvicious$ ./svwar.py -e 6000-6005
146.231.123.15
WARNING:TakeASip:could not bind to :5060 - some process might already be
listening on this port.  Listening on port 5061 instead
| Extension | Authentication |
------------------------------
| 6002      | reqauth        |
| 6000      | reqauth        |
| 6001      | reqauth        |
```

Step 3: Cracking the password

```
courage@courage-desktop:~/Desktop/sipvicious$ ./svcrack.py -u6002
-r1-6005 146.231.123.15
WARNING:ASipOfRedWine:could not bind to :5060 - some process might
already be listening on this port.  Listening on port 5061 instead
/home/courage/Desktop/sipvicious/helper.py:387:  DeprecationWarning:  the
md5 module is deprecated; use hashlib instead import md5
| Extension | Password |
------------------------
| 6002      | 6002     |
```

Discussion:

When the variable `alwaysauthreject=no`, we were able to enumerate valid users on Asterisk. The results of step 2 show that there are three users registered and all require passwords for authentication. In step 3 we picked user 6002 and ran svcrack an online password cracker. The results of step 3 show that user 6002 has password 6002.

This is where password security becomes important. Chapter 2 mentioned that users should have strong passwords. We repeated the same experiment using a password that is found in a dictionary and we were able to crack it. Moreover we used a strong password like `R%u7&Fq@^c441U` which we could not manage to crack.

5.3.2 Enumeration attempt with variable `alwaysauthreject=yes`

Step 1: Device identification

```
courage@courage-desktop:~/Desktop/sipvicious$ ./svmap.py 146.231.123.15
WARNING:DrinkOrSip:could not bind to 0.0.0.0:5060 - some process might
already be listening on this port.  Listening on port 5061 instead
| SIP Device          | User Agent          | Fingerprint          |
--------------------------------------------------------------------
| 146.231.123.15:5060 | Asterisk PBX 1.6.0.6 | Asterisk /
Linksys/PAP2T-3.1.15(LS) / Asterisk PBX
```

Step 2: Scanning for valid users

```
root@courage-desktop:~/Desktop/sipvicious# ./svwar.py 146.231.123.15
WARNING:TakeASip:could not bind to :5060 - some process might already be
listening on this port.  Listening on port 5061 instead
ERROR:TakeASip:SIP server replied with an authentication request for an
unknown extension.  Set --force to force a scan.
WARNING:root:found nothing
```

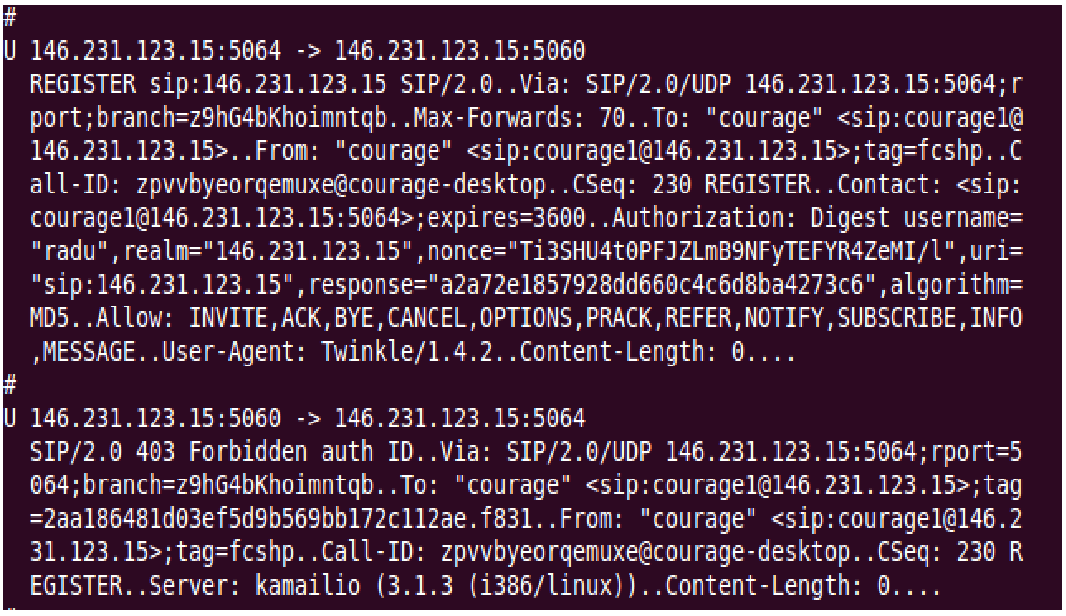
Discussion:

Step 1 show that the server is up and running. With step 2 we were not be able to enumerate valid usernames on the server. The server responded with the following error. `"ERROR:TakeASip:SIP server replied with an authentication request for an unknown extension.  Set --force to force a scan."`

5.4 Enumerating valid usernames on Kamailio server

The objective of this experiment is to show that an attacker can enumerate valid users by sending a REGISTER requests to the server and analyse the response.

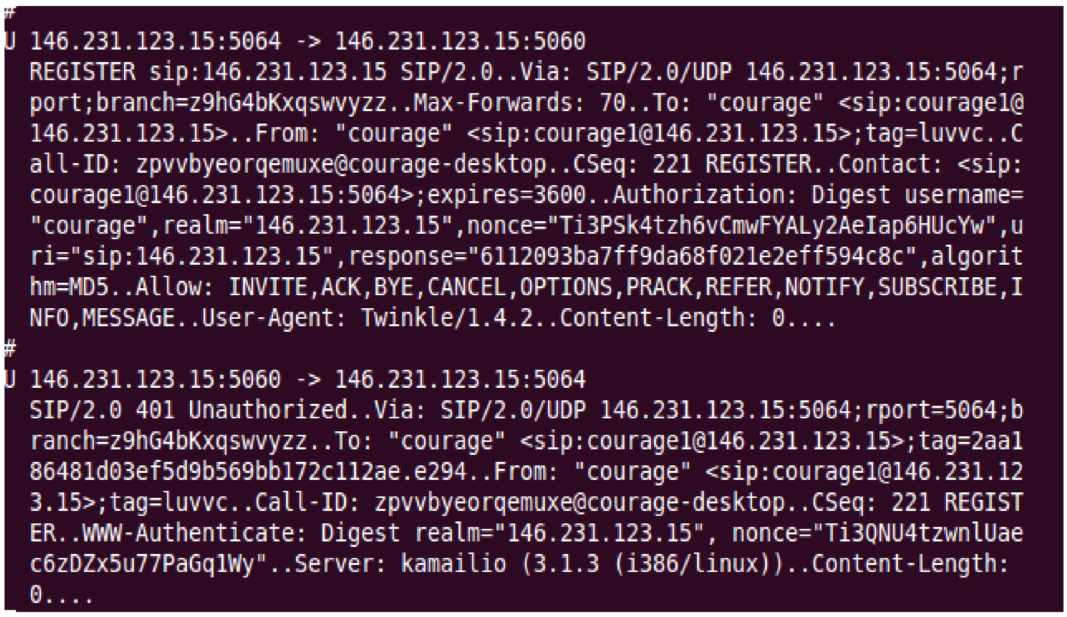
Step 1: Valid username and wrong password



```
#
U 146.231.123.15:5064 -> 146.231.123.15:5060
  REGISTER sip:146.231.123.15 SIP/2.0..Via: SIP/2.0/UDP 146.231.123.15:5064;r
  port;branch=z9hG4bKhoimntqb..Max-Forwards: 70..To: "courage" <sip:courage1@
  146.231.123.15>..From: "courage" <sip:courage1@146.231.123.15>;tag=fcshp..C
  all-ID: zpvvbyeorqemuxe@courage-desktop..CSeq: 230 REGISTER..Contact: <sip:
  courage1@146.231.123.15:5064>;expires=3600..Authorization: Digest username=
  "radu",realm="146.231.123.15",nonce="Ti3SHU4t0PFJZLmB9NFyTEFYR4ZeMI/l",uri=
  "sip:146.231.123.15",response="a2a72e1857928dd660c4c6d8ba4273c6",algorithm=
  MD5..Allow: INVITE,ACK,BYE,CANCEL,OPTIONS,PRACK,REFER,NOTIFY,SUBSCRIBE,INFO
  ,MESSAGE..User-Agent: Twinkle/1.4.2..Content-Length: 0....
#
U 146.231.123.15:5060 -> 146.231.123.15:5064
  SIP/2.0 403 Forbidden auth ID..Via: SIP/2.0/UDP 146.231.123.15:5064;rport=5
  064;branch=z9hG4bKhoimntqb..To: "courage" <sip:courage1@146.231.123.15>;tag
  =2aa186481d03ef5d9b569bb172c112ae.f831..From: "courage" <sip:courage1@146.2
  31.123.15>;tag=fcshp..Call-ID: zpvvbyeorqemuxe@courage-desktop..CSeq: 230 R
  EGISTER..Server: kamailio (3.1.3 (i386/linux))..Content-Length: 0....
```

Figure 5.4: 403 Forbidden - Server Response

Step 2: Invalid username and wrong password



```
#
U 146.231.123.15:5064 -> 146.231.123.15:5060
  REGISTER sip:146.231.123.15 SIP/2.0..Via: SIP/2.0/UDP 146.231.123.15:5064;r
  port;branch=z9hG4bKxqswvyzz..Max-Forwards: 70..To: "courage" <sip:courage1@
  146.231.123.15>..From: "courage" <sip:courage1@146.231.123.15>;tag=luvvc..C
  all-ID: zpvvbyeorqemuxe@courage-desktop..CSeq: 221 REGISTER..Contact: <sip:
  courage1@146.231.123.15:5064>;expires=3600..Authorization: Digest username=
  "courage",realm="146.231.123.15",nonce="Ti3PSk4tzh6vCmwFYALy2AeIap6HUcYw",u
  ri="sip:146.231.123.15",response="6112093ba7ff9da68f021e2eff594c8c",algorit
  hm=MD5..Allow: INVITE,ACK,BYE,CANCEL,OPTIONS,PRACK,REFER,NOTIFY,SUBSCRIBE,I
  NFO,MESSAGE..User-Agent: Twinkle/1.4.2..Content-Length: 0....
#
U 146.231.123.15:5060 -> 146.231.123.15:5064
  SIP/2.0 401 Unauthorized..Via: SIP/2.0/UDP 146.231.123.15:5064;rport=5064;b
  ranch=z9hG4bKxqswvyzz..To: "courage" <sip:courage1@146.231.123.15>;tag=2aa1
  86481d03ef5d9b569bb172c112ae.e294..From: "courage" <sip:courage1@146.231.12
  3.15>;tag=luvvc..Call-ID: zpvvbyeorqemuxe@courage-desktop..CSeq: 221 REGIST
  ER..WWW-Authenticate: Digest realm="146.231.123.15", nonce="Ti3QNU4tzwnlUae
  c6zDZx5u77PaGq1Wy"..Server: kamailio (3.1.3 (i386/linux))..Content-Length:
  0....
```

Figure 5.5: 401 Unauthorised - Server Response

Discussion:

Firstly, from the results, we could easily determine the server version by looking at the responses (Kamailio 3.1.3 running on linux). This is important information to an attacker because outdated versions of applications are often vulnerable to exploits that have been published and archived for anyone to view [4]. For instance the following vulnerability report [28].

"**Description:**
IPTel has confirmed vulnerabilities in the SIP (Session Initiation Protocol) implementation in all versions of SIP Express Router up to 0.8.9.
The vulnerabilities have been identified in the INVITE message used by two SIP-endpoints during the initial call setup. The impact of successful exploitation of the vulnerabilities has not been disclosed but could potentially result in a compromise of a vulnerable device.
Solution:
Upgrade to version 0.8.10 and apply patch:"

Secondly, the server responses will help the attacker to enumerate all valid extensions on the Kamailio server. Server response "401-Unauthorised", "403-Forbbiden" and "404-Not Found" will tell if the username exists or not. If both username and password are wrong the server response is "401-Unauthorised". If the peer exist and has wrong password the response will be "403-Forbbiden". If the peer does not exist the response will be "404-Not Found".

5.5 Summary

It is essential to know what kind of information attackers get from server responses. Nmap show the status of ports on the system and on IP phone. All open ports should be filtered and unnecessary ports should be closed. SIPvicious experiments show that it is important to set the variable `alwaysauthreject=yes`. This will prevent an attacker from knowing the identities of registered users on the system. Users should have strong passwords that are difficult to brute force. Characteristics of good passwords are mentioned in Chapter 2. Additionally, an attacker can use leaking information like server versions to check archived vulnerabilities and use them to his advantage. Finally, Kamailio server responses can be used to enumerate all valid extensions on the system.

Chapter 6

Conclusion and Future Work

This chapter summarises main coclusions of this thesis. It also provides some insights into some of the possible future extensions.

6.1 Summary of work

We managed to achieve the two goals in securing the iLanga system to be less prone to attacks. However, it is important to stress that security is an on-going process and as new threats emerge, they must be mitigated. Over the course of the investigation, we identified threats as well as vulnerabilities within the system. Among the threats was to brute force attacks, on both sip user accounts and the server root account. Vulnerabilities were also noted which include dialplan injection, ability to enumerate user accounts on Kamailio proxy server. Some weaknesses are within the Asterisk server itself, for instance, it allows users to create weak passwords. Toll fraud is another problem that we noted on the system whereby unauthourised users were making international calls.

There are several countermeasures we implemented on the system to make iLanga less attack-prone. We used the inbuilt Linux firewall (iptables) to filter incoming traffic. The firewall was together with the perl script that we developed to quarantine offending IP addresses. To prevent denial of service to legitimate users, we developed a web UI that will alert the admnistrator when they got blocked. During the initial installation, we installed the Asterisk server as a non-root as recommended by the definitive guide in Chapter 4. There were basic configurations that we did, for instance, setting the variable `alwaysauthreject=yes` and changing the default port 5060 to another arbitrary port. In order to prevent toll fraud, we created a secure dialplan with a secure default context and all incoming contexts being filtered to prevent dialplan injection. Administrators always need to remotely login to the server machine so we configured public key authentication and this would make brute force methods hard to perform on the root account.

We also made sure that all default passwords were changed especially on Kamailio which comes with two default passwords `openserrw` and `openserro`. During the initial MySQL installation, we followed a guide provided by MySQL for securing the initial MySQL accounts. We immediately changed all default passwords in the grant tables and making sure all accounts have passwords. Finally, we made an update on all servers so that they use the latest server versions.

6.2 Future Work

The following are considered as key areas for possible extension:

6.2.1 Enabling password aging

A mechanism can be developed that will ensure that all passwords expire after a certain period of time, for example a month. If a compromised account changes password it will prevent an attacker from continuously using that account.

6.2.2 Modifying Kamailio server responses

Experiments in Chapter 5 show that an attacker can enumerate the usernames on the PBX due server responses. The attacker simply sends a REGISTER request to the server, capture the packets, and analyse the response.

Server response “401-Unauthorised”, “403-Forbbiden” and “404-Not Found” will tell if the username exists or not. If both username and password are wrong the server response is “401-Unauthorised”. If the peer exist and has wrong password the response will be “403-Forbbiden”. If the peer does not exist the response will be “404-Not Found”.

The server responses can be modified to prevent the ability to enumerate usernames. That is, the responses can be hardened such that only successful registration will receive a response. A point to note is that this is how the protocol works which results in the problem.

6.2.3 Monitoring Call Detail Records (CDR)

It is difficult to monitor log files because of the rate at which the files grow. Every call that is made is recorded in call data record file. The best way to monitor call events is to write a simple program that scans all records and looks for suspicious calls. For example checking for international calls made at odd hours. Of course the program should be able to detect different times of different countries. This can be done by extracting the caller ID from the variable `${EXTEN}` to determine which country the call is being made from. The program will then adjust the time appropriately relative to the server time. An example of a suspicious event would be a caller in South Africa making an international call to United States of America (USA) from 1:00am to 5:00am (USA time).

6.2.4 Enabling a port knocking mechanism

Manually changing the default port might not effectively solve the problem. Another way is to keep the default port closed. If a legitimate user wants to use a service, the user should knock and the port will be opened. In order to open the port, the client should provide a key. This key should be included in the header of the packet. The firewall can open the packet and read the field with the key. If the key matches with what it knows, then it should open the port.

Besides port knocking, another way is to write a program that randomly generates a port number within a specified range (avoiding known ports). The program will then check if that port number is being used by another application. If not, this port number will become the listening port for the server. The server should then broadcast its new port number to its clients continuously so that all clients are updated. This should be feasible by including a new field in the packet header, which the client application would open to see the new port number. The challenge however might be writing an application on the client's side that will automatically switch to the new port without the user manually changing the port.

6.3 Summary

This project has identified past threats and potential key threats to the iLanga system. Countermeasures for these threats were implemented. A basic configuration to make the system secure has also been provided. Appendix D has the concise guide on how to secure iLanga.

While the iLanga system is currently providing its service in a secure state, some weaknesses were identified in the protocols as evidenced by experiments in Section 5.4. This makes our system not entirely secure, but with the current state of the system it will require well determined attackers to attack it. With the secure iLanga we managed to monitor Denial of Service on legitimate users. Attackers who try flooding the server with requests in order to bring the service down may not achieve this because latest version of Kamailio comes with anti-flood functionality enabled. Many authors mentioned SPIT as one of the possible threats to VoIP systems and proposed solutions to it. In this thesis, we did not investigate much on SPIT as a threat because of time limitations.

It should be emphasised that further enhancements can be done to make iLanga more secure and easy to monitor. To aid with the latter, an extension of the web UI has been created to provide a simple mechanism to monitor security issues via a browser. The administrator is able to view all offending IP addresses that have been blocked, monitor strength of user passwords and monitor the user status to prevent denial of service.

Bibliography

[1] Douglas C. Sicker and Tom Lookabaugh. Voip security: Not an afterthought. pages 1–9, 2004.

[2] Mugdha Vairagade. Introduction to netfilter/iptables. [Online at http://www.ibm.com/developerworks/linux/library/s-netip/index.html] last visited June 2011.

[3] A. Terzoli J. Penton. ilanga: A next generation voip-based, tdm-enabled pbx. In SATNAC 2004.

[4] David Endler and Mark Collier. *Hack Exposed VoIP: Voice Over IP Security Secrets & Solutions*. McGraw-Hill, 2007.

[5] Errol A. Blake. Network security: Voip security on data network–a guide. In *Proceedings of the 4th annual conference on Information security curriculum development*, InfoSecCD '07, pages 27:1–27:7, New York, NY, USA, 2007. ACM.

[6] M Theoharidou G F Marias D Gritzalis S Dritsas, J Mallios. Threat analysis of the session initiation protocol regarding spam, 2007.

[7] Allan B. Johnston Henry Sinnreich ˙ *Internet Communications Using SIP*. Wiley Publishing, Inc, 2006.

[8] Paul Stalvig. Session initiation protocol (sip): A five-function protocol, August 2007.

[9] Ilker Korkmaz and Mehmet Emin Dalkilic. The weak and the strong password preferences: a case study on turkish users. In *Proceedings of the 3rd international conference on Security of information and networks*, SIN '10, pages 56–61, New York, NY, USA, 2010. ACM.

[10] Kamaljit Singh. On improvements to password security. *SIGOPS Oper. Syst. Rev.*, 19:53–60, January 1985.

[11] Werner Puschitz. Securing and hardening red hat linux production systems. [Online at http://www.puschitz.com/SecuringLinux.shtml] last visited June 2011.

[12] Chlotia Posey Garrison. Encouraging good passwords. pages 1–4.

[13] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34:39–53, April 2004.

[14] Federal Bureau of Investigation. Fbi voip server intrusions in north carolina banks and businesses. Technical report, Federal Bureau of Investigation, 2009.

[15] John Todd. Seven steps to better sip security with asterisk. [Online at http://blogs.digium.com/2009/03/28/sip-security/] last edited March 2009, last visited June 2011.

[16] Nils Aschenbruck, Matthias Frank, Peter Martini, Jens Tolle, and Heinz dieter Richmann. Present and future challenges concerning dos-attacks against psaps in voip networks, 2006.

[17] Peter Steinbacher, Florian Fankhauser, Christian Schanes, and Thomas Grechenig. Black-box approach for testing quality of service in case of security incidents on the example of a sip-based voip service: work in progress. In *Principles, Systems and Applications of IP Telecommunications*, IPTComm '10, pages 101–110, New York, NY, USA, 2010. ACM.

[18] Dr. Andreas, U. Schmidt, Nicolai Kuntze, and Rachid El Khayari. Spam over internet telephony and how to deal with it.

[19] David Planella. Sshopensshkeys. [Online at https://help.ubuntu.com/community/SSH/OpenSSH/Keys] last edited March 2011, last visited June 2011.

[20] A. Lockhart. *Network security hacks.* Hacks series. O'Reilly, 2007.

[21] Digium. Asterisk. [Online at http://www.asterisk.org/] last visited October 2011.

[22] VoIP-Info. Asterisk slimming. [Online at http://www.voip-info.org/wiki/view/Asterisk+Slimming] last visited June 2011.

[23] Leif Madsen, Jim Van Meggelen, and Russell Bryant. *Asterisk : The Definitive Guide.* O'REILLY, 2011.

[24] Digium. Security advisories. [Online at http://www.asterisk.org/security] last visited October 2011.

[25] Teamforest. Automatically block failed sip peer registrations. [Online at http://www.teamforrest.com/blog/171/asterisk-no-matching-peer-found-block/] last visited July 2011.

[26] Nmap. Nmap security scanner. [Online at http://nmap.org/book/man.html] last visited June 2011.

[27] SecTechno. Hacking exposed voip/sip. [Online at http://www.sectechno.com/2011/05/23/hacking-exposed-voipsip/] last visited October 2011.

[28] Secunia. Iptel sip express router sip vulnerabilities. [Online at http://secunia.com/advisories/8119] last visited October 2011.

Appendix A

How to install Asterisk as non-root

In order to compile Asterisk from the source code, the following dependencies should be installed.

```
# apt-get install gcc
# apt-get install g++
# apt-get install make
# apt-get install libncurses5-dev
```

```
# cd /usr/local/src
# wget http://downloads.asterisk.org/pub/telephony/asterisk/
asterisk-1.8-current.tar.gz
# tar xvfz asterisk-1.8.current.tar.gz
```

Install Asterisk

```
# cd aterisk-1.8
# ./configure
# make
# make install
```

Create a new user asteriskpbx

```
# useradd asteriskpbx && passwd asteriskpbx
```

At this point we have installed files in their default locations so we need to change file permissions to match the asteriskpbx user.

```
# chown -R asteriskpbx:asteriskpbx /usr/lib/asterisk/
# chown -R asteriskpbx:asteriskpbx /var/lib/asterisk/
# chown -R asteriskpbx:asteriskpbx /var/spool/asterisk/
# chown -R asteriskpbx:asteriskpbx /var/log/asterisk/
# chown -R asteriskpbx:asteriskpbx /var/run/asterisk/
# chown asteriskpbx:asteriskpbx /usr/sbin/asterisk/
```

Create /etc/asterisk/ directory

```
# mkdir -p /etc/asterisk
# chown asteriskpbx:asteriskpbx /etc/asterisk
```

Copy the sample asterisk.conf file into the `/etc/asterisk` and

```
# cp /usr/local/src/asterisk.con.sample /etc/asterisk/asterisk.conf
```

Change the runuser and rungroup to have values of `asteriskpbx`

```
# vim /etc/asterisk/asterisk.conf
```

Giving sudo access to the asteriskpbx is done by modifying the sudoers file.

```
# visudo
```

In the sudoers file, you should see

```
## allows people in a group to run as all commands
%wheel ALL=(ALL) ALL
```

Save and exit (`Esc` +:wq + `Enter`)

Open the `/etc/group` using nano or any editor of your choice

```
# nano /etc/group
```

Find the line that starts with wheel. Modify it to look like

```
wheel:x:10:root,asteriskpbx
```

Save and exit

Exit root

At this point, asteriskpbx should run as a non-root.

Appendix B

How to configure public key authentication

The following steps can be found on [19].

Creating public and private key on the command line

1. Create directory named ssh (may exist by default)

   ```
   # mkdir ~/.ssh
   ```

2. Change the directory permissions

   ```
   # chmod 700 ~/.ssh
   ```

3. Generate the key and this is where you set key encryption level to 4096 bits

   ```
   # ssh-keygen -t rsa -b 4096
   ```

4. The created key by default will be in ~/.ssh/ directory and its named id_rsa and id_rsa_pub

   Now copy the public key to the server

   ```
   # scp ~/.ssh/id_rsa.pub X.X.X.X(eg 146.231.123.15) :
   ~/.ssh/authorized_keys
   ```

5. On both client and server issue the following command

   ```
   # chmod 0600 ~/.ssh/*
   ```

6. Create SSH users on the server side

   ```
   # groupadd sshusers
   # usermod -a -G sshusers courage
   ```

7. SSH server configuration

   Using vim editor or any editor of your own choice open the file sshd-config

8.
   ```
   # vim /etc/ssh/sshd-config
   ```

9. Make the following changes

   ```
   ListenAddress 146.231.123.15 [server IP address]
   PermitRootLogin no
   AuthourizedkeyFile %h/.ssh/authourized_keys
   UsePAM no
   ```

After this one should be able to login using SSH. The private key has to be stored on a secure place.

Appendix C

Perl Script that blocks offending IP Addresses

```
#!/usr/bin/perl -w

use strict;

use warnings;

my (@failhost);

my %currblocked;

my %addblocked;

my $action;

my $count;

my $counter;


open (MYINPUTFILE, "/var/log/asterisk/messages") or die "\n", $!, "Does log
file file exist\?\n\n";

open(MYOUTPUTFILE, '>/etc/asterisk/data.txt');


while (<MYINPUTFILE>) {

        my ($line) = $_;
        chomp($line);

        if ($line =~ m/\' failed for \'(.*?)\' - Peer is not supposed to register/)
{
                push(@failhost,$1);
        }
        if ($line =~ m/\' failed for \'(.*?)\' - No matching peer found/){

                push(@failhost,$1);
        }
        if ($line =~ m/\' failed for \'(.*?)\' { Wrong password/) {
                 push(@failhost,$1);
        }
}

my $blockedhosts = `/sbin/iptables -n -L INPUT`;
while ($blockedhosts =~ /(.*)/g) {
        my ($line2) = $1;
        chomp($line2);
        if ($line2 =~ m/(\d+\.\d+\.\d+\.\d+)(\s+)/) {
                 $currblocked{ $1 } = 'blocked';
       }
}

while (my ($key, $value) = each(%currblocked)){
        print $key .  "\n";
}
if (@failhost) {
        &count_unique(@failhost);

        $counter=1;

        while (my ($ip, $count) = each(%addblocked)) {

                if (exists $currblocked{ $ip }){

                        print "$ip already blocked\n";

                        print MYOUTPUTFILE "[".$counter."]"."\nip = ".$ip."\nattemp
= ".$count ."\n";

                        $counter++;

} else {

        if($count > 6)

                  { $action = ‘/sbin/iptables -I INPUT -s $ip -j DROP‘; # create
an user chain

                  print MYOUTPUTFILE "[".$counter."]"."\nip = ".$ip."\nattempts
= ".$count ."\n";

                  $counter++;

        }

          print "$ip has $count attempts.\n"; } }

}

else {

        print "no failed registrations.\n";

}

close(MYOUTPUTFILE);


sub count_unique {

        my @array = @_;

        my %count;

        map { $count{$_}++ } @array;

        map {($addblocked{ $_ } = ${count{$_}})} sort keys(%count);

}
```

Appendix D

A concise guide on how to secure iLanga

1. Install Asterisk as a non-root. Refer to Appendix A for detailed steps.
2. Set the variable `alwaysauthreject=yes` in the sip.conf file.
3. Change the default port 5060 to an arbitrary port. This will require port modification on client side.
4. Enforce strong passwords for all users. Users should be provided with guidelines on how to create strong passwords. A program or a script should run in the background checking password strength for all users on the system and alerting the administrator of any weak passwords. Change all default passwords.
5. Create a secure dialplan. The default context should not have a context that will cost the organisation money. Prevent dialplan injection by using the inbuilt `FILTER()` to filter anything defined as incoming context.
6. Make use of the firewall (iptables works well for Unix systems) to quarantine offending IP addresses.
7. Monitor suspicious activities especially long distance calls made during odd hours.
8. Monitor denial of service on legitimate users. The system should be always available to users when they need it.

9. Use public key authentication for remote login. Generate the private and public keys. The public key will reside on the server. Login will be done using the private key instead of password. Refer to Appendix B for detailed steps.

10. Subscribe to Asterisk Security Advisories for vulnerability updates and versions affected on `www.asterisk.org/security`.