6-2016

Design and Implementation of a Real-Time Honeypot System for the Detection and Prevention of Systems Attacks

Farouk Samu
fsamu2001@yahoo.com

Design and Implementation of a Real-Time Honeypot System for the Detection and Prevention of Systems Attacks

by

Farouk Samu

A Thesis

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Information Assurance

St. Cloud, Minnesota

June, 2016

Thesis Committee:
Dr. Amos O. Olagunju, Chairperson
Dr. Ezzat Kirmani
Dr. Jerry Wellik

Abstract

A honeypot is a deception tool, designed to entice an attacker to compromise the electronic information systems of an organization. If deployed correctly, a honeypot can serve as an early-warning and an advanced security surveillance tool. It can be used to minimize the risks of attacks on IT systems and networks. Honeypots can also be used to analyze the ways attackers try to compromise an information system and to provide valuable insights into potential system loopholes. This research investigated the effectiveness of the existing methodologies that used honeynet to detect and prevent attacks. The study used centralized system management technologies called Puppet and Virtual Machines to implement automated honeypot solutions. A centralized logging system was used to collect information about the source IP address, country, and timestamp of attackers. The unique contributions of this thesis include: The research results show how open source technologies is used to dynamically add or modify hacking incidences in a high-interaction honeynet system; the thesis outlines strategies for making honeypots more attractive for hackers to spend more time to provide hacking evidence.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor Professor Amos O. Olagunju for the endless support and guidance in my thesis study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my thesis committee members: Professors Wellik Jerry and Prof. Ezzat Kirmani, for their insightful advice and encouragement.

More importantly, I take this opportunity to express the profound gratitude from my deep heart to my senior brother Tayib Samu and his family, for their love and continuous support– both spiritually and materially.

Finally, I would like to thank my wife, Tawadatu Samu, for supporting me throughout writing this thesis and my life in general.

Table of Contents

Chapter Page

Chapter Page

List of Tables

List of Figures

Figure Page

Chapter 1: Introduction

Introduction

In recent times, there has been a growing interest in security and information protection for network systems. Network systems contain valuable data and resources that must be protected from attackers. Security experts often use honeypots and honeynets to protect network systems. Honeypot is an outstanding technology that security experts use to tap new hacking techniques from attackers and intruders.

According to Spitzner (2002), founder of the Honeynet Project, "a honeypot is security resource whose value lies in being probed, attacked, or compromised" (p. 58). It can also be defined as "an information system resource whose value lies in unauthorized or illicit use of that resource" (Spitzner, 2003). In other words, a honeypot is a decoy, put out on a network as bait to lure attackers. Honeypots are typically virtual machines, designed to emulate real machines, act or create the appearance of running full services and applications, with open ports that might be found on a typical system or server on a network (Sahu & Richhariya, 2012).

In this research, a centralized system management called Puppet was used to automate the configuration of four servers. A VMware Virtual Machine was used to implement automated honeypot solutions. The study provided targets with interesting services such as Apache webserver, MYSQL server, File Transfer Protocol (FTP) server and Simple Mail Transfer Protocol (SMTP) server. A centralized Logstash server was used to process and index logs. Elasticsearch was used to store logs. Kibana was used to search and visualize the logs.

Problem Statement

Attempts by attackers to breach security systems are rising every day. Intruders use tools like SubSeven, Nmap and LoftCrack to scan, identify, probe and penetrate Enterprise systems.

Firewalls are put in place to prevent such unauthorized access to the Enterprise Networks. However, Firewalls cannot prevent attacks coming from Intranet.

An Intrusion Detection System (IDS) reviews network traffic and identify exploits and vulnerabilities; it is able to display alert, log event, and e-mail administrators of possible attacks. An Intrusion Prevention System on the other hand makes attempts to prevent known intrusion signatures and some unknown attacks due to the knowledge of attack behaviors in its database. However, an IDS can generate thousands of intrusion alerts every day, some of which are false positives. This makes it difficult for an IDS to detect and identify the actual threats and to protect assets. Thus, human intervention is required to investigate the attacks detected and reported by an IDS (Kaur, Malhotra, & Singh, 2014).

Nature and Significance of the Problem

Honeypots can dramatically reduce false positives. Honeypots are designed to track illegal activities. This makes it extremely efficient to use honeypots for detecting attacks. Honeypots only collect data from human or processes interactions. Organizations that may log thousands of alerts a day with traditional technologies will only log a hundred alerts with honeypots (Kaur et al., 2014). Honeypots, on the other hand, can easily be used to identify and capture new attacks. New attacks can easily be detected by a honeypot because any illegal activity is an anomaly. Thus honeypots can be used to collect, manage and analyze more attack data.

Objective of the Study

The objectives of this experiment are: (1) to use free and open-source technologies and methods to reduce the amount of manual intervention needed to add to or modify a high-

interaction honeypot system suitable for academic research, and (2) to detect attack patterns on services and come out with solution to mitigate the attacks.

Research Questions

1. How should open source technologies be used to dynamically add or modify hacking incidences in a high-interaction honeynet system?
2. How should honeypots be made more attractive for hackers to spend more time to provide hacking evidences?

Definition of Terms

Honeypot: Honey Pot Systems are decoy servers or systems setup to gather information about attackers who intrude into a system.

Puppet: a unique approach to IT automation for discovering, configuring, and managing network infrastructure

Virtual Machines: a virtual machine is a type of computer application used to create a virtual environment, which is referred to as "virtualization." Some types of virtualization let a user run multiple operating systems on one computer at the same time.

HonSSH: a great tool for high-level honeypot interaction

Chapter 2: Background and Review of Literature

Introduction

This chapter discusses the background literature associated with the issues of the design and implementation of effective honeypots and honeynets.

Background Related to the Problem

During the last few years, many different uses of honeypots have been proposed. Some of them are deployed to waste hackers' time (Liston, 2002), others to reduce spam activity (Krawetz, 2004) or to deceive attackers (Virvilis, Serrano, & Vanautgaerden, 2014), and some to analyze hacker intrusion steps (Akkaya & Thalgott, 2010). For several years, the security community has used honeypots to analyze different techniques deployed by attackers.

Spitzner (2003) introduced two types of honeypots: low-level interaction honeypots are computer software that emulate operating systems and services–usually applied in a production environment within an organization, and high-level interaction honeypots that involve the deployment of actual operating systems on real or virtual machines–it is also used by security research. Also, Spitzner (2003) defined two critical requirements of honeynet architecture: (1) the data control reduced risk by ensuring that once an attacker breaks into honeynet systems, those compromised systems cannot be used to attack or harm other systems, and (2) data capture ensures that security experts can detect and capture all activities performed by the attacker, even if they are encrypted. This study used high interaction honeypots to capture data from real-life systems.

Sobesto et al. (2011) introduced DarkNOC, a management and monitoring tool for complex honeynets. It consists of different types of honeypots as well as other data collection

devices. They designed a solution that is able to process large amount of malicious traffic received by a large honeynet and effectively provide a user-friendly web interface to show potential compromised hosts to network security administrators, and also provide the overall network security status. This research used Kibana to implement, a user-friendly web interface, to visually show the attacks eith high incidence. Dittrich (2004) noted that a distributed honeynet can grow in size. A single honeypot cannot effectively monitor and manage attacks. The data analysis of attacks originating from a large number of individual honeypots within a network is difficult. Dittrich recommended the use of Manuka, a database tool with front and back end client applications for resolving this difficult problem. The database includes attributes of systems augment a search for operating system type, version and services installed. He stated that the Manuka tool can only take one person to install honeypots, upload it to a database and rapidly deploy it in a distributed network. Likewise, this study used an open source Puppet automation management tool to deploy servers and services on honeypots without manual intervention.

Weiler (2002) proposed a system that can be applied by large organizations to defend against distributed denial of service attacks. His study relied on honeypot technology that has two advantages. First, the system can defend the operational network of the organization against known distributed denial of service attacks (DDoS) and against new future types of attacks. Second, the system can trap the attacker and record the compromised components to provide evidence for use in a legal action. Weiler said the lessons learned can be implemented in the rest of the network as a protective shield.

Weiler (2002) designed includes a demilitarized zone network that implemented services such as web, mail, ftp and DNS for access by external networks. The local internal network (LAN) of the organization is in another zone protected by a firewall. The infrastructure, then introduced a new system: a honeypot that will mimic the internal network and attract DDoS attackers. Furthermore, Weiler made internal systems in the organization to act as a honeypot. "For example, if the attacker's compromised packets to the webserver of the corporation are detected, the packets go to the honeypot for processing. The reply the attacker gets can be indistinguishable from a real reply of the web server" (Weiler, 2002, p. 5).

The above design solved three problems, these are: attacks can be detectable, attack packets can be actively directed to the honeypot, and the honeypot is able to simulate the organization's network infrastructure, at least the parts known to the attacker.

Wilson, Maimon, Sobesto, and Cukier (2015) studied the effect of a surveillance banner in the attacked computer system that reduced the probability of commands being typed in the system during longer first system trespassing incidents. The study further stated the probability of commands being typed during subsequent system trespassing incidents, on the same system, is conditioned by the presence of a surveillance banner and by whether commands have been entered by previous trespassing incidents.

Wilson et al.'s (2015) results show that the average number of system trespassing incidents per computer are 4.44 and 4.48 for no surveillance banner and surveillance banner computers, respectively. This is not a statistically significant difference; however, the presence of a banner did affect the seriousness of trespassing incidents. The study further stated that intruders receiving a surveillance banner, that spent at least 50 seconds on the system, are 8%

less likely to enter commands into the system than respective intruders that do not receive a surveillance banner. Of those that did not previously enter commands into the system, 38% of those with the surveillance banner and 47% of those with no surveillance banner entered commands during the second trespassing incident. Moreover, of those that do enter commands into the system, 67% of those with the surveillance banner and 63% of those with no surveillance banner entered commands during the second trespassing incident.

The study originates from a randomized controlled trial conducted at a large public university in the United States. Over a 7-month experimental period, 660 target computer systems were deployed after being compromised by system trespassers. These systems produced 2,942 system trespassing incidents (Wilson et al., 2015). Again, they said, each of these systems was randomly assigned to either display a surveillance banner (n=324) or not display a surveillance banner (n=336) upon each entry to the relevant system. The surveillance banner is depicted at left exactly as the intruders saw it, with the message: “This system is under continuous surveillance. All user activity is being monitored and recorded” (Wilson et al., 2015, p. 11)

Literature Related to the Problem

Stockman, Rein, & Heile (2015) used open-source honeynet system to study system banner message effects on hackers. The study was performed for 25 days, gathering data on almost 200,000 events.

According to Stockman et al. (2015), there were 510 logins allowed via the password spoofing, and of the 510 logins, 280 were served a warning banner, and 230 were served no banner at all. The mean duration of attempts for those with the warning banner was 15.29

seconds, and for those without, 23.45 seconds. The median duration of those with the warning banner was 9 seconds, and of those without, 11 seconds. In addition, intruders did nothing while logged in due to the fact that the system recorded only a handful of commands. However, they stated that, the study did not allowed attackers to log in as root, and that might cause lower levels of activity.

Stockman et al.'s (2015) works used the manual intervention in setting up the honeynet. However, this study explored ways to provide targets of greater interest for hackers to attack services such as web or database servers which appear to be resources of actual value for attackers.

Döring (2005) proposed five test cases within which honeypot can be deployed differently in separate environment and explains their individual attributes. According his study, "the amount of attacks occurring in a protected environment are less than the number of attacks coming from the unprotected environment at least they should. Therefore, a comparison of results afterwards needs to focus on the environment."

The results of Döring's (2005) research provided a pratical approach for honeypot deployment. He discussed four phases: analysis, development, realization and conclusion phase. According to him, the first phase of development start by gathering knowledge about Honeypots and defining requirements. The development phase concentrates on selecting a particular solution and preparing the requirements for an experiment. The next phase deals with the practical research and empirical analysis and the final phase the gathered results and conclusions are summarized.

Hoque and Bikas (2012) presented an Intrusion Detection System (IDS), using genetic algorithm (GA) to efficiently detect various types of network attacks. Their approached used evolution theory to information evolution in order to filter the traffic data and thus reduce complexity. They used randomly generated population of chromosomes, which represent all possible solution of a problem that are considered candidate solutions. From each chromosome different positions are encoded as bits, characters or numbers.

Similarly, Jaiganesh, Sumathi, and Vinitha (2013) compared two data mining classification algorithms for intrusion detection system, Iterative Dichotomiser2 (ID3) algorithm and C4.5 algorithm. The results presented by Jaiganesh et al. (2013) shows C4.5 algorithm was best suited for intrusion detection, because it uses numeric and nominal data.

In addition, Stiawan, Abdullah, and Idris (2011) presented mapping problem and challenges of Intrusion Prevention System (IPS). They summarized current methods in implementing IPS, the future directions and challenges in reseach field. Stiawan et al. stated in their results that, "Hybrid techniques is one of solution for classification and detection intrusion threat." The hybrid IPS is able to data capture accuracy and precision for threats.

Chapter 3: Methodology

Introduction

This chapter discusses the details of the network design used to investigate attacks on our honeypots and to detect and prevent this events in real time. The approach used for this network was a mixed one, using both literature studied and laboratory experiments.

Design of the Study

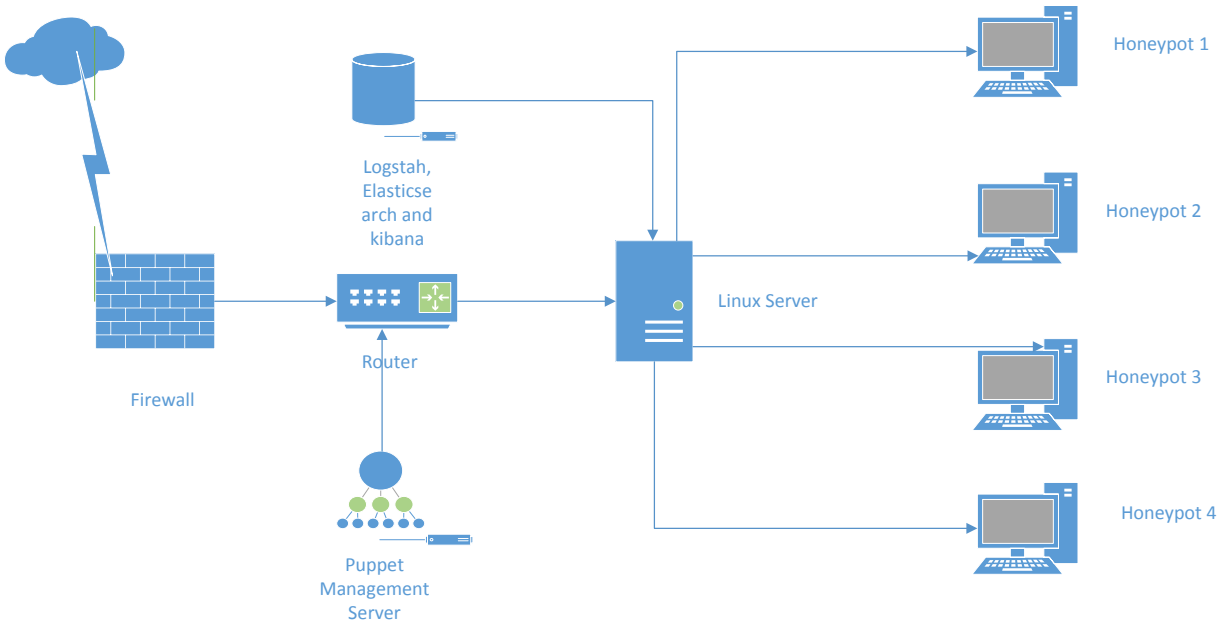

Figure 1: Network Design Diagram

Figure 1 shows the system configuration used to examine, detects, and prevent attacks on our honeypots. It consists of a firewall, router, Bifrozt Linux Server, HonSSH, Filebeat Elasticsearch, Logstash, Kibana, Puppet and four virtual machine honeypots. The system setup is explained below.

Tools and Techniques

Puppet is an open-source configuration management utility. It runs on many Unix-like systems as well as on Microsoft Windows. It is an automation software for IT system administrators. It is used to automate repetitive tasks such as the installation of applications and services, patch management, and deployments. Puppet was selected because it was used to install servers and automate services in this design.

Elasticsearch was used because it is able to achieve fast search responses instead of searching the text directly, it searches an index instead. Elasticsearch is designed to be scalable and distributed.

Kibana is an open source (Apache Licensed) browser based analytics and search dashboard for Elasticsearch that visualized the data to provide a better interpretation. It was used to visualize captured logs from compromised honeypots.

HonSSH was used because it is the only currently available up-to-date open-source high-interaction honeypot software. Alternatives examined included Honeyd and Honeywall. However, all these other open-source honeypots are either restricted to low- or medium-interaction, or are badly out of date and difficult or impossible to install on modern Linux distributions (Stockman et al., 2015).

Hardware and Software Environment

The honeynet system consist of Linux virtual machines running on VMware Workstation, version 12, hosted on Intel Core i7-4765T CPU, with 12 Gigabytes of RAM and 500 Gigabytes of storage. The virtual machine based honeynet has of 4 honeypot systems, a centralized logging host and Puppet automation host. Each honeypot system is consisted with three individual hosts;

a router, a firewall, and a Linux server host. The Linux server provided a Secure Shell (SSH) service to act as the target for attackers.

Data Collection

Data were collected using Logstash, an open source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations. It cleanses and democratizes data for diverse advanced downstream analytics and visualization use. This was chosen because of its centralized log management which was used to collect all logs from compromised honeypots.

Chapter 4: Implementation, Data Analysis, and Results

Introduction

This chapter presents a detailed description of honeypot implementation. The detailed installation and configuration steps of the honepots are provided. The chapter outlines the resources required for data analysis and generating reports.

Puppet Configurations

Puppet configure and automate user management on all puppet agents (honeypots). First, the hostname of puppet master was set as pupperserver.com in the hostname configuration file. In addition, the /etc/hosts configuration file of puppet master was edited. Each node was set to point to the proper IP address as shown below.

192.168.1.10 puppetserver.com

192.168.1.12 node1

192.168.1.6 node2

192.168.1.8 node3

192.168.1.13 node4

Installing Puppet Enterprise

1. *Download puppet-enterprise-2015.3.2-sles-12-x86_64.tar.gz*
2. *Tar –xf puppet-enterprise-2015.3.2-sles-12-x86_64.tar.gz*
3. *Run sudo ./puppet-enterprise-installer*
4. *On the browser https://puppetserver.com:3000*
5. *Add security exceptions*
6. *Click get started*

7. *Select Monolithic for deployment type*
8. *Enter puppet master FQDN: puppetserver.com*
9. *Select Install PostgreSQL*
10. *Enter password*
11. *Submit*
12. *Click deploy now*

Installing Puppet Enterprise Agent

The agents (honeypots) were installed with the same operating system and architecture as the Puppet master. The process started by logging into all the agent nodes and run the following command: *curl -k https://puppetserver.com:8140/packages/2015.3.2/install.bash | sudo bash*. This script detected the operating system on which the agent is running, sets up an apt that refers back to the Puppet master, and pulls down and installs the puppet-agent packages. After the installation is completed, certificate request is sent to the puppet master for approval. On the master console the agents are accepted as follows:

1. *Click on nodes*
2. *Select unsigned certificates*
3. *Click Accept All*

Each honeypot (node) was configured with Network Time Protocol (NTP) to synchronize the clocks to Chicago time reference by running *# puppet module install puppetlabs-ntp* on the puppet master and *puppet agent–t* on all the agents (honeypots). Secure Shell (SSH) module was installed by running *puppet module install ghoneycutt-ssh* to allow remote access to any of the honeypots.

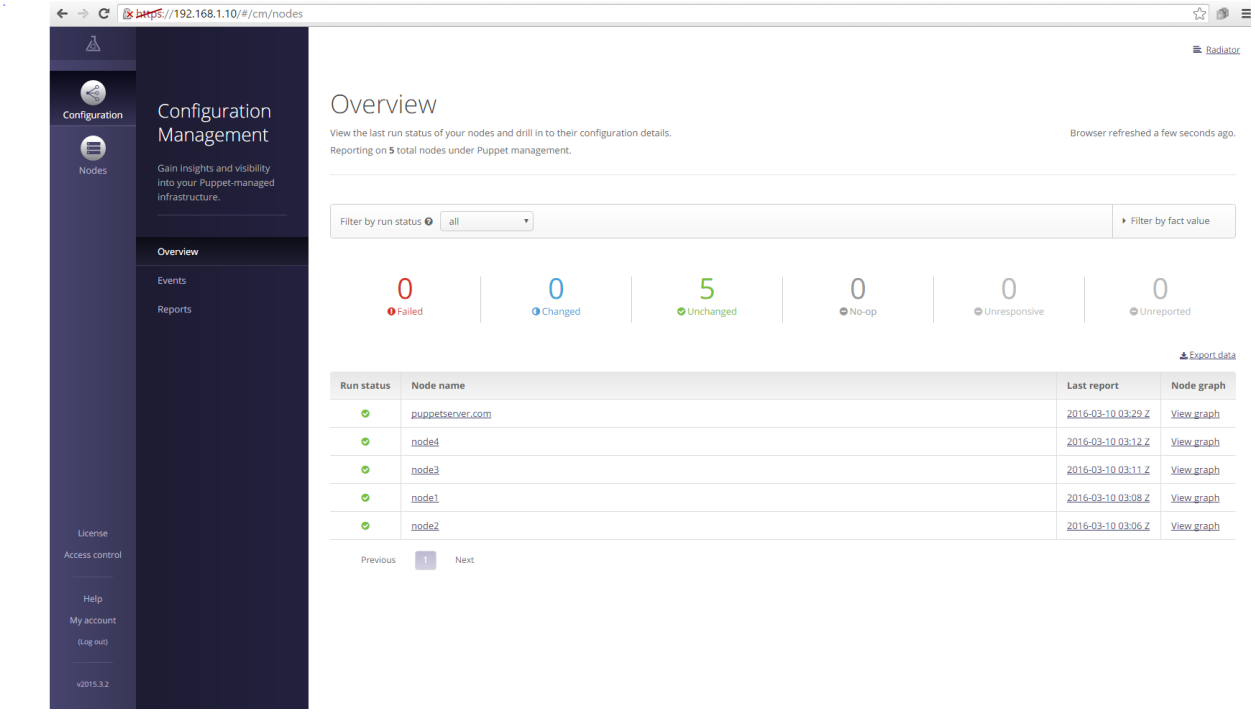

Figure 2: Overview Configuration of the Honeypots

Figure 2 shows the overview of management console that was used to automate four honeypots which includes Puppet master, honeypot 1, honeypot 2, honeypot 3, and honeypot 4.

Honeypot 1 Configuration

Apache module was configured to manage webserver to attract attackers using *puppet module install puppetlabs-apache.* See appendix for configuration file.

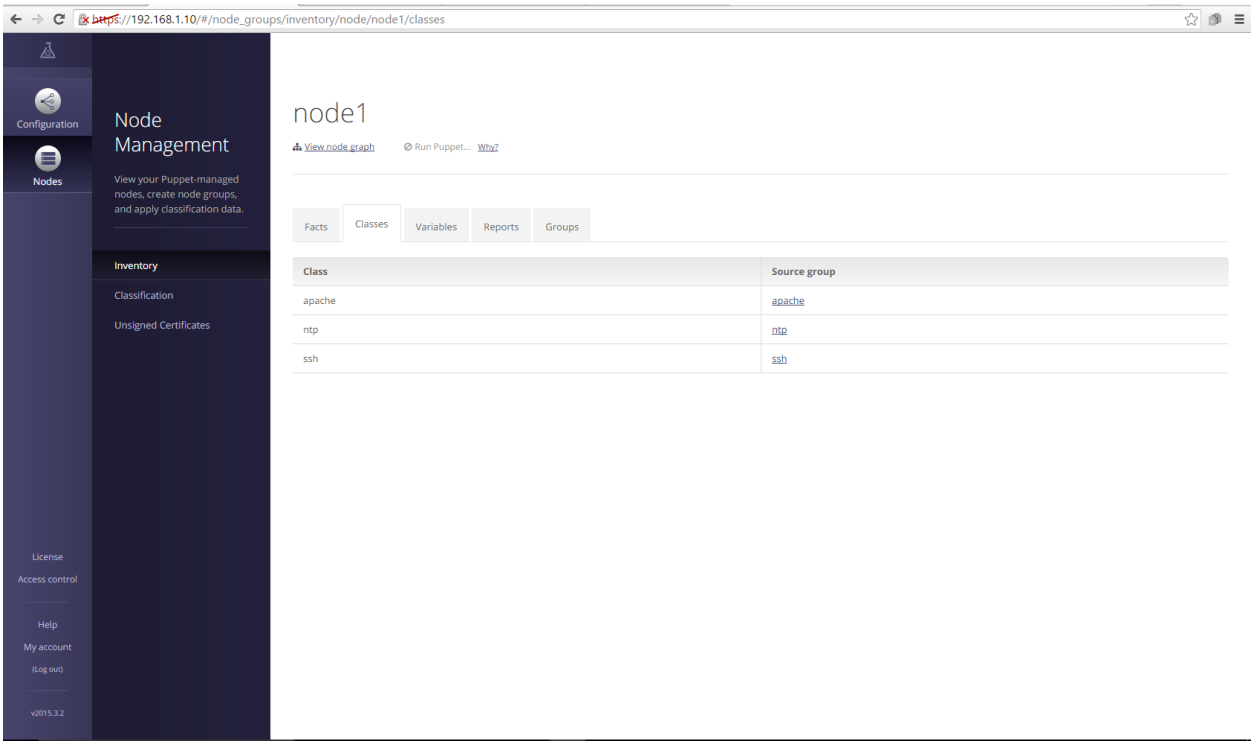

Figure 3: The Configuration of Honeypot 1

Figure 3 shows the automation of Apache webserver, NTP server and SSH server on honeypot 1.

Honeypot 2 Configuration

MYSQL server was installed using the following module from puppetlab forge *puppet module install puppetlabs-mysql*. See appendix for configuration file.

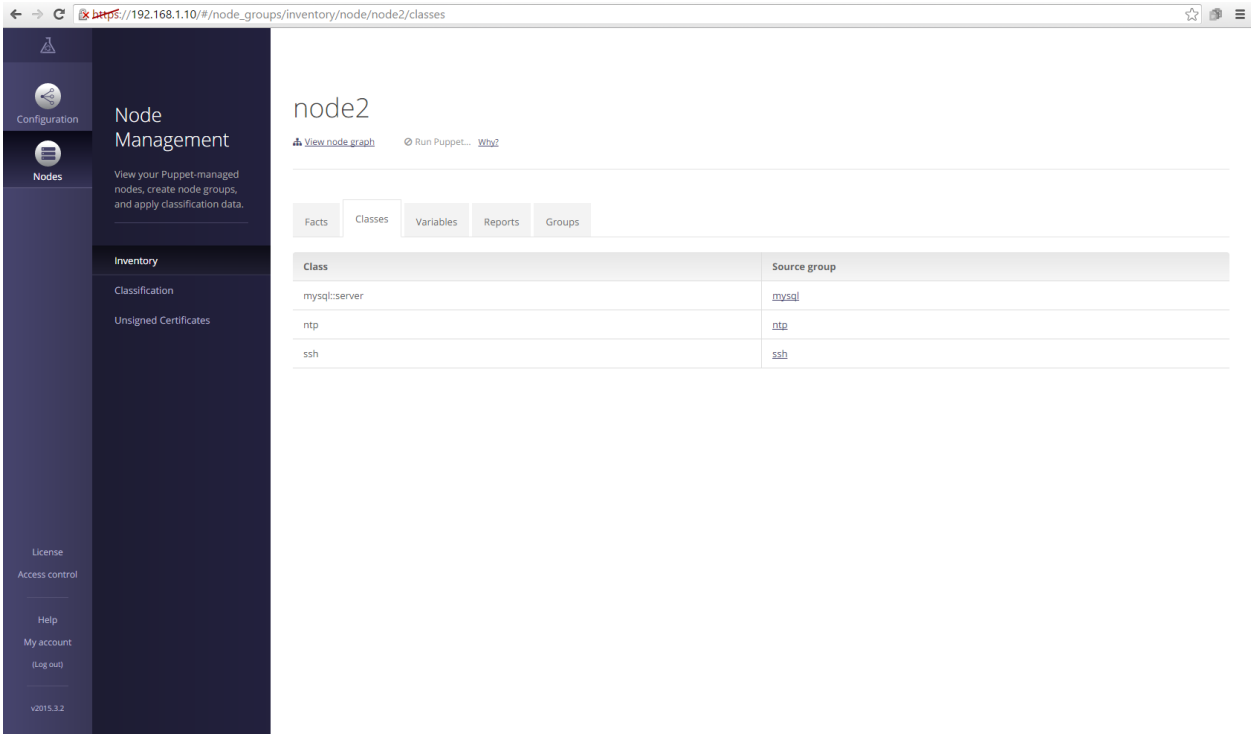

Figure 4: The Configuration of Honeypot 2

Figure 4 shows the automation of MYSQL server, NTP server and SSH server on honeypot 2.

Honeypot 3 Configuration

The File Transfer Protocol (FTP) server was configured by using this setup *puppet module install thias-vsftpd*. See appendix for file configuration.

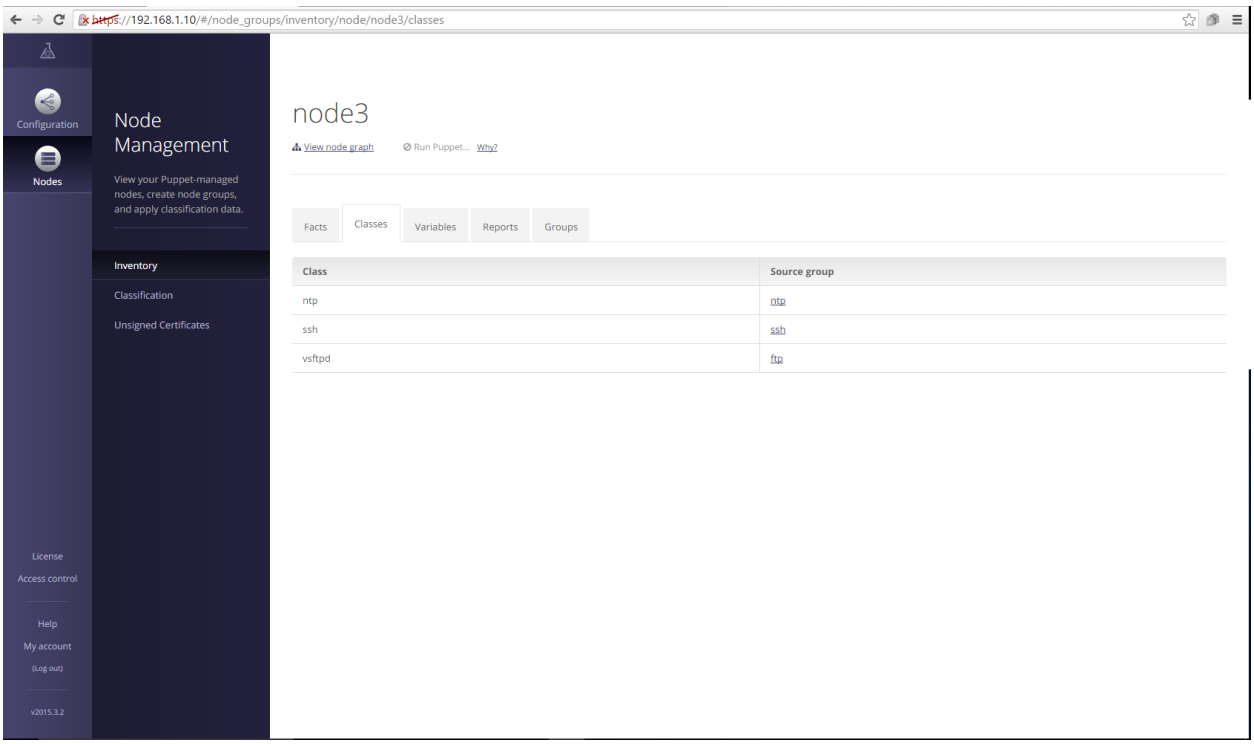

Figure 5: The Configuration of Honeypot 3

Figure 5 shows the automation of FTP server, NTP server and SSH server on honeypot 3.

Honeypot 4 Configuration

The Simple Mail Transfer Protocol (SMTP) server was configured using *puppet module install thias-vsftpd* module. See appendix for file configuration.

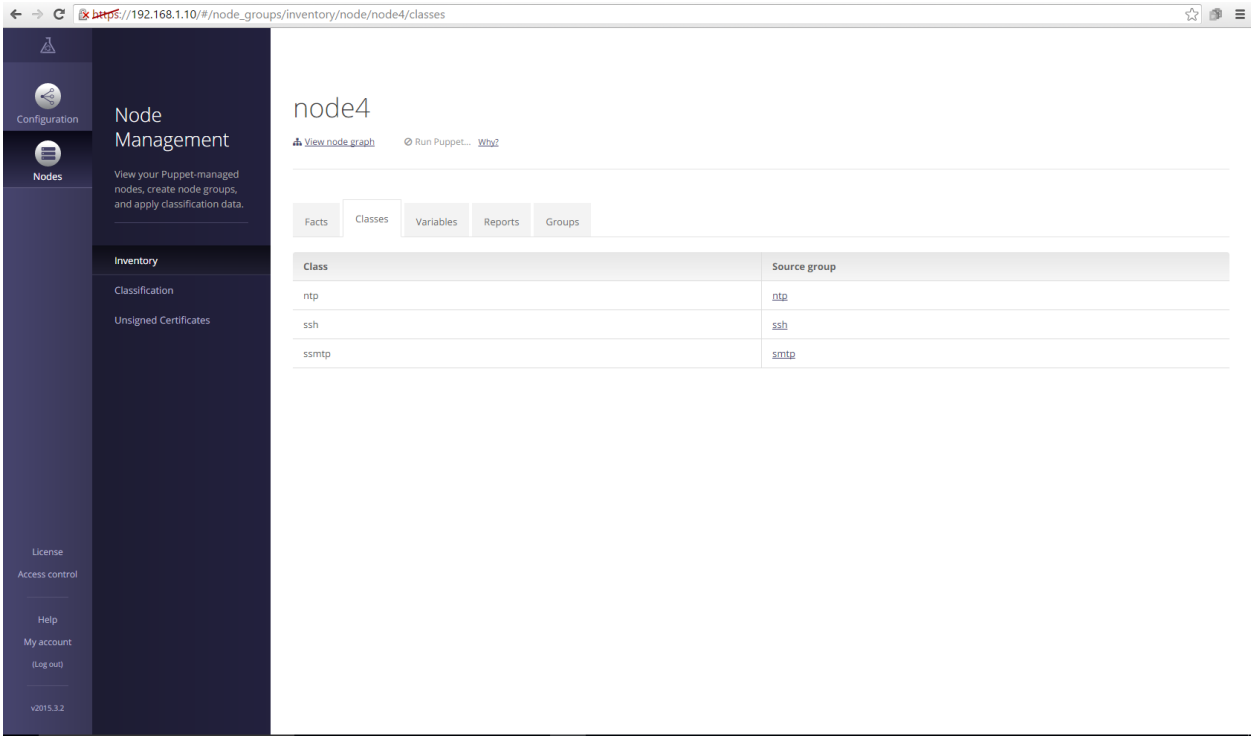

Figure 6: The Configuration of Honeypot 4

Figure 6 shows the automation of SMTP server, NTP server and SSH server on honeypot 4.

HonSSH Configurations

HonSSH is a high-interaction Honeypot solution, when configured as static it sits between an attacker and a honeypot, creating two separate SSH connections between them, and when configured using script it sits between attacker and multiple honeypots, creating one too many SSH connections between them. In this study, HonSSH was configured to sit between the attacker and the four honeypots configured using Puppet automation management tool. HonSSH server was configured on the Bifrozt Linux machine.

1. *Install dependencies sudo apt-get install python-dev openssl python-openssl python-pyasn1 python-twisted python-2.7 python-mysqldb python-geoip*
2. *git clone [https://github.com/tnich/honssh.git.](https://github.com/tnich/honssh.git)*

3. *See appendix for honssh configuration file*

4. *./honsshctrl.sh start*

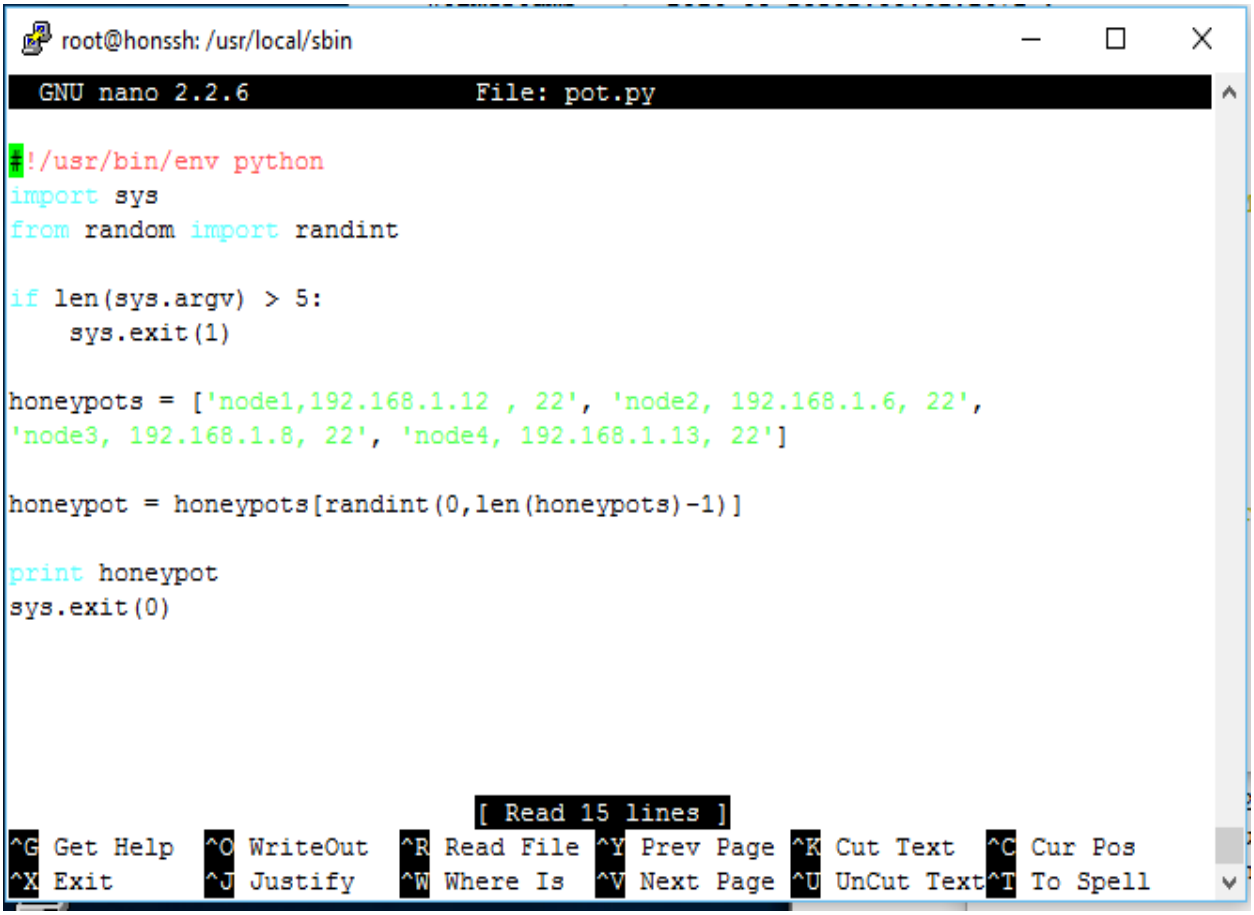

Figure 7: Random Selection of Honeypot

Figure 7 shows a python script to randomly select any of the honeypots for the attacker to connect to any of the four honeypots.

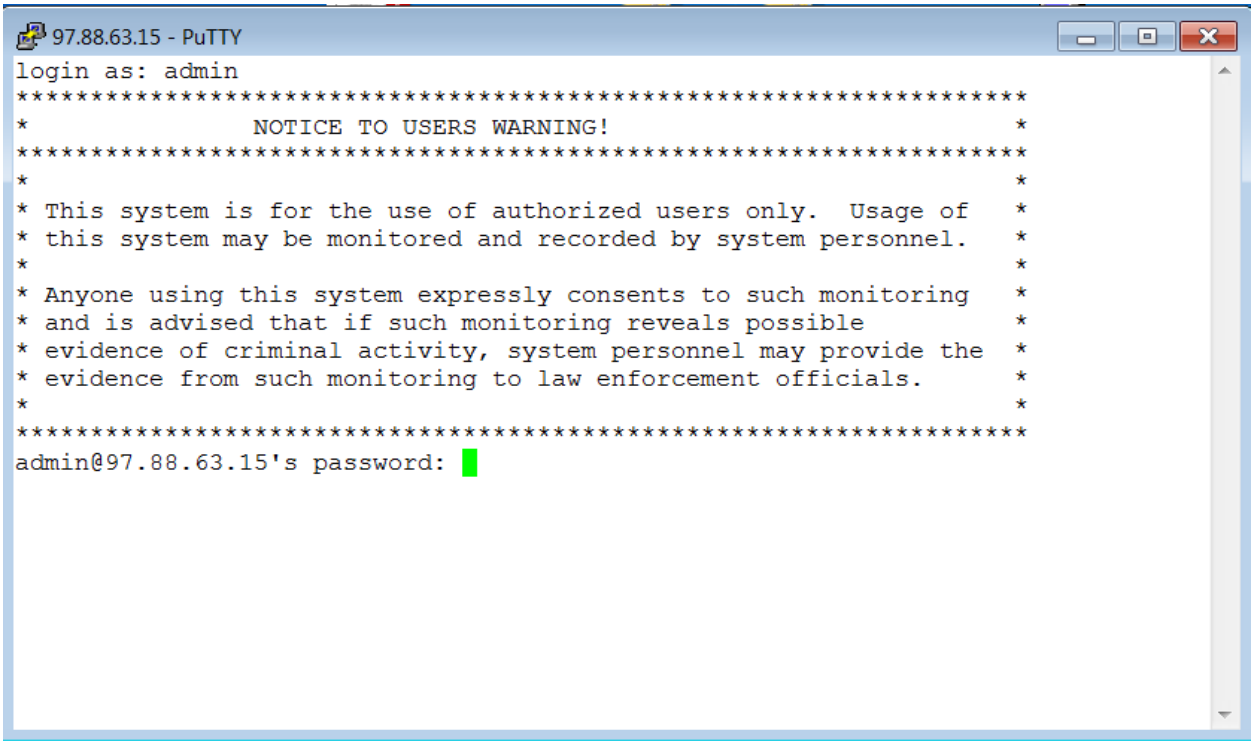

Figure 8: Banner Message

Figure 8, shows banner message that an attacker sees when they login. Puppet was configured to allow a banner message that serves as deterrent to attackers and any bypass of this message to the honeypots is considered as an attack.

Elasticsearch, Logstash, and Kibana Configuration

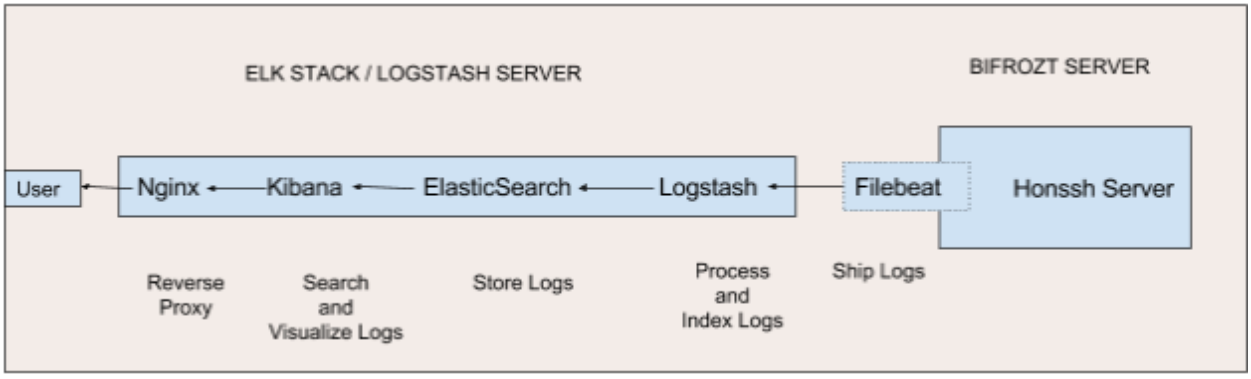

Figure 9: ELK Stack Server and Bifrozt Server

Figure 9 shows detailed configuration between ELK/Logstash server and the Bifrozt server. Elasticsearch 2.2, Logstash 2.2, and Kibana 4.4 was installed on Ubuntu 14.04 Server (ELK/Logstash Server) to gather and visualize the logs of honeypots in a centralized location by using Filebeat 1.1 to ship the logs. Logstash is an open source tool for collecting, parsing, and storing logs for future use. Kibana is a web interface that can be used to search and view the logs that Logstash has indexed. Both of these tools are based on Elasticsearch which is used for storing logs (Anicas, 2015).

Install Java 8

1. *sudo add-apt-repository -y ppa:webupd8team/java*
2. *sudo apt-get update*
3. *sudo apt-get -y install oracle-java8-installer*

Install Elasticsearch

1. *wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add –*
2. *echo "deb http://packages.elastic.co/elasticsearch/2.x/debian stable main" | sudo tee -a /etc/apt/sources.list.d/elasticsearch-2.x.list*

3. *sudo apt-get update*
4. *sudo apt-get -y install elasticsearch*
5. *sudo vi /etc/elasticsearch/elasticsearch.yml*
6. *network.host: localhost*
7. *sudo service elasticsearch restart*
8. *sudo update-rc.d elasticsearch defaults 95 10*

Install Kibana

1. *echo "deb http://packages.elastic.co/kibana/4.4/debian stable main" | sudo tee -a /etc/apt/sources.list.d/kibana-4.4.x.list*
2. *sudo apt-get update*
3. *sudo apt-get -y install kibana*
4. *sudo vi /opt/kibana/config/kibana.yml*
5. *server.host: "localhost"*
6. *sudo update-rc.d kibana defaults 96 9*
7. *sudo service kibana start*

Install Nginx

Because we configured Kibana to listen on `localhost`, we must set up a reverse proxy to allow external access to it. We used Nginx for this purpose.

1. *sudo apt-get install nginx apache2-utils*
2. *sudo htpasswd -c /etc/nginx/htpasswd.users kibanaadmin*
3. *sudo vi /etc/nginx/sites-available/default*
4. *See nginx config file*
5. *sudo service nginx restart*

Install Logstash

The Logstash package was installed from the same repository as Elasticsearch as follows:

1. *echo 'deb http://packages.elastic.co/logstash/2.2/debian stable main' | sudo tee /etc/apt/sources.list.d/logstash-2.2.x.list*
2. *sudo apt-get update*
3. *sudo apt-get install logstash*

Generate SSL Certificates

Filebeat was used to ship logs from the HonSSH server and a SSL certificate and key pair was created. The certificate was used by Filebeat to verify the identity of ELK server. Two directories were created to store the certificate and private key with the following commands:

1. *sudo mkdir -p /etc/pki/tls/certs*
2. *sudo mkdir /etc/pki/tls/private*

The ELK Server's private IP address was added to the `subjectAltName` field of the SSL certificate that was generated:

sudo vi /etc/ssl/openssl.cnf

subjectAltName = IP: 192.168.1.14

1. *cd /etc/pki/tls*
2. *sudo openssl req -config /etc/ssl/openssl.cnf -x509 -days 3650 -batch -nodes -newkey rsa:2048 -keyout private/logstash-forwarder.key -out certs/logstash-forwarder.crt*

Copy SSL Certificate

The SSL certificate was copied from ELK Server to Bifrozt Server:

1. *scp –P 58595 /etc/pki/tls/certs/logstash-forwarder.crt cna@192.168.1.5:/tmp*

On the Bifrozt Server:

1. *sudo mkdir -p /etc/pki/tls/certs*
2. *sudo cp /tmp/logstash-forwarder.crt /etc/pki/tls/certs/*

Install Filebeat Package

On the Bifrozt Linux Server, Filebeat was installed as below:

1. *echo "deb https://packages.elastic.co/beats/apt stable main" | sudo tee -a /etc/apt/sources.list.d/beats.list*
2. *wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -*
3. *sudo apt-get update*
4. *sudo apt-get install filebeat*
5. *sudo service filebeat restart*
6. *sudo update-rc.d filebeat defaults 95 10*

After installation, Filebeat transported logs from the HonSSH server to the Logstash server. Filebeat serves as a log shipping agent that utilizes the lumberjack networking protocol to communicate with Logstash.

Filebeat Configuration

```
filebeat:
  prospectors:
    -
      paths:
        - /opt/honssh/logs/*.log
      document_type: log
  registry_file: /var/lib/filebeat/registry

output:
  logstash:
    hosts: ["192.168.1.14:5044"]
    bulk_max_size: 1024

    tls:
      certificate_authorities: ["/etc/pki/tls/certs/logstash-forwarder.crt"]
shipper:
```

```
logging:
  files:
```

Logstash Server Configuration

```
 input {
  beats {
    port => 5044
    ssl => true
    ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
    ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
  }
}

filter {
    grok {
        match => { "message" => "%{TIMESTAMP_ISO8601:timestamp} \[%{WORD:text},$
    }

    mutate {
        gsub => ["rest", """, "'"]
        gsub => ["rest", "False", "false"]
    }

    json {
        source => "rest"
    }

    mutate {
        remove_field => ["rest", "message"]
    }
}


output {
  elasticsearch {
    host => ["localhost:9200"]
    protocol => http
    manage_template => false
    index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
    document_type => "%{[@metadata][type]}"
  }
    stdout { codec => rubydebug }
}
```

Data Analysis and Results

The experimental investigation on the honeypots was performed for 10 days. Figure 10 shows the total hits on the honeypots. During this period, almost 500,000 attacks were captured. The experiment mirrors real world implementation of honeynet system because a Webserver, File Transfer Protocol, and Simple Mail Tranfer Protocol and MYSQL servers were installed on the honeypots. The attack data was analyzed by calculating the frequencies, percentages, Chi square and the P-value for each of the honeypots. The data analysis investigated the top source IP addresses, top five attacks, top five countries, daily attacks, top ten passwords, top ten usernames and top ten source ports.

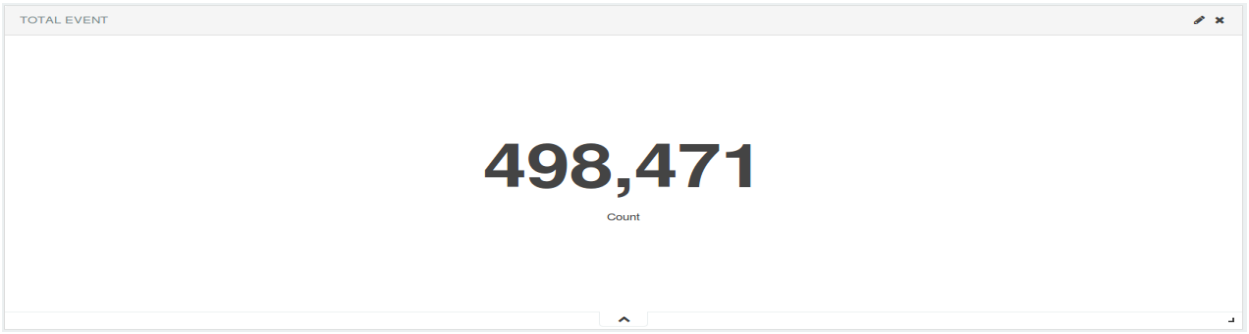

Figure 10: Total Event of Honeypots

Table 1: Top 5 Source IP

ATTACKS ON HONEYPOT 1		
Top Source IP	**Frequency**	**Percentage %**
218.25.208.124	831	50.49%
222.186.21.211	368	22.36%
121.12.127.94	274	16.65%
59.49.5.235	143	8.69%
119.146.221.68	30	1.82%
TOTAL	1646	100.00%

X^2 = 1155.974484
P –Value = < 0.0001

Table 1 shows the frequencies and percentages of the attacks on honeypot 1 from the top five source IP addresses. The Chi square value of 1155 for investigating the equality of the proportions of that attacks from these IP addresses is significant at the 0.0001 level. There tends to be a high incidence of attacks from IP address 218.25.208.124.

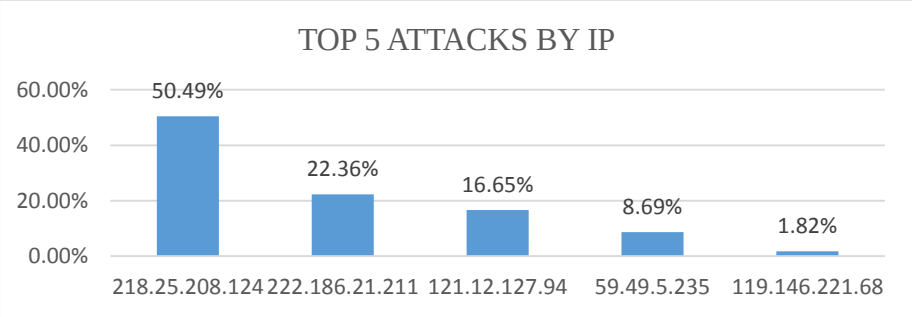

Figure 11: The Graph of Top 5 Source IP

Figure 11 shows the graph of the attacks on honeypot 1 from the top five source IP addresses. Clearly, the IP address 218.25.208.124 produced the most attacks (50.49%) and the IP address 119.146.221.68 generated the least attacks (1.82%). Consequently, the IP address 218.25.208.124 had the high incidence of attacks on honeypot 1.

Table 2: Top 5 Countries Attacks

Countries	Frequency	Percentage %
China	1745	90.84%
United States	65	3.38%
Viet Nam	58	3.02%
Bulgaria	27	1.41%
Ukraine	26	1.35%
TOTAL	1921	100.0%

X^2 = 6028.034357
P –Value = < 0.0001

Table 2 shows the frequencies and percentages of the attacks on honeypot 1 from the top five countries. The Chi square value of 6028 for investigating the equality of the proportions of that attacks from these countries is significant at the 0.0001 level. There tends to be a high incidence of attacks from China.

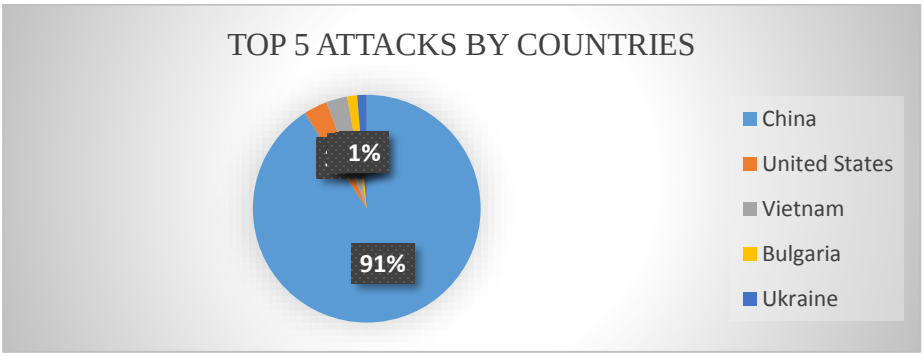

Figure 12: The Graph of Top 5 Countries

Figure 12 shows the graph of the attacks on honeypot 1 from the top five countries. Clearly, China produced the most attacks (90.84%) and Ukraine generated the least attacks (1.35%). Consequently, China had the high incidence of attacks on honeypot 1.

Table 3: Daily Attacks

Timestamp	Frequency	Percentage %
4/4/2016	4	.20%
4/5/2016	118	6.03%
4/6/2016	30	1.53%
4/7/2016	5	0.26%
4/8/2016	99	5.06%
4/9/2016	88	4.50%
4/10/2016	480	24.54%
4/11/2016	25	1.28%
4/12/2016	182	9.30%
4/13/2016	925	47.29%
TOTAL	1956	100.0%

$X^2 = 1438.203476$
P –Value = < 0.0001

Table 3 shows the frequencies and percentages of top ten daily attacks on honeypot 1. The Chi square value of 1438 for investigating the equality of the proportions of that attacks on these timestamp is significant at the 0.0001 level. There tends to be a high incidence of attacks on 4/13/2016.

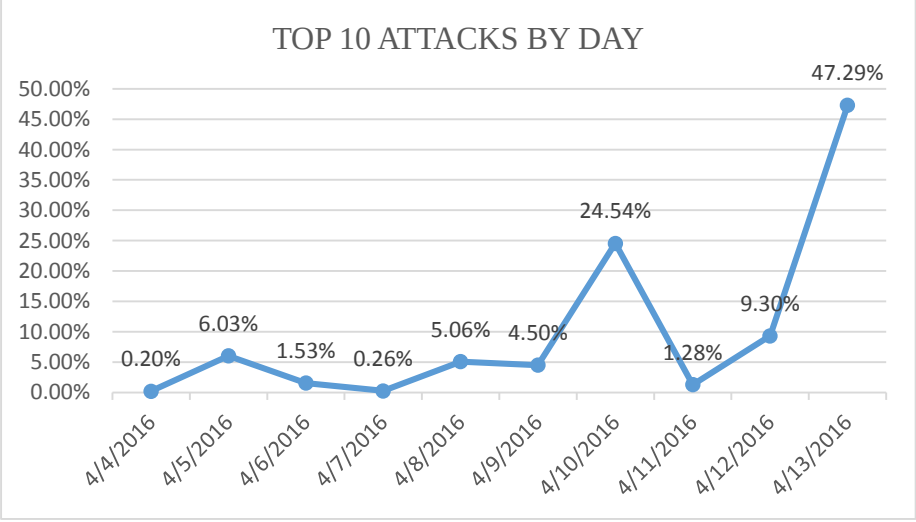

Figure 13: The Graph of Daily Attacks

Figure 13 shows the graph of the attacks on honeypot 1 base on the top ten timestamps. Clearly, 4/13/2016 produced the most attacks (47.29%) and 4/4/2016 generated the least attacks (0.20%). Consequently, 4/13/2016 had the high incidence of attacks on honeypot 1.

Table 4: Top 5 Source IP Attacks

ATTACKS ON HONEYPOT 2		
Top Source IP	**Frequency**	**Percentage %**
218.25.208.124	859	51.72%
222.186.21.211	391	23.54%
121.12.127.94	273	16.44%
59.47.5.235	100	6.02%
119.146.221.68	38	2.29%
TOTAL	1661	100.0%

$X^2 = 1279.201686$
P –Value = < 0.0001

Table 4 shows the frequencies and percentages of the attacks on honeypot 2 from the top five source IP addresses. The Chi square value of 1279 for investigating the equality of the proportions of that attacks from these IP addresses is significant at the 0.0001 level. There tends to be a high incidence of attacks from IP address 218.25.208.124.

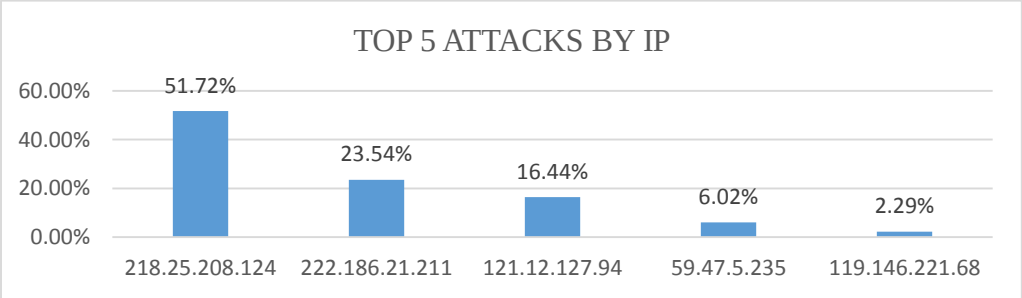

Figure 14: The Graph of Top 5 Source IP

Figure 14 shows the graph of the attacks on honeypot 2 from the top five source IP addresses. Clearly, the IP address 218.25.208.124 produced the most attacks (51.72%) and the IP address 119.146.221.68 generated the least attacks (2.29%). Consequently, the IP address 218.25.208.124 had the high incidence of attacks on honeypot 2.

Table 5: Top 5 Countries Attacks

Countries	Frequency	Percentage %
China	1749	90.81%
United States	61	3.17%
Viet Nam	60	3.12%
Bulgaria	38	1.97%
Ukraine	18	0.93%
TOTAL	1926	100.0%

$X^2 = 6038.92731$
P –Value = < 0.0001

Table 5 shows the frequencies and percentages of the attacks on honeypot 2 from the top five countries. The Chi square value of 6038 for investigating the equality of the proportions of that attacks from these countries is significant at the 0.0001 level. There tends to be a high incidence of attacks from China.

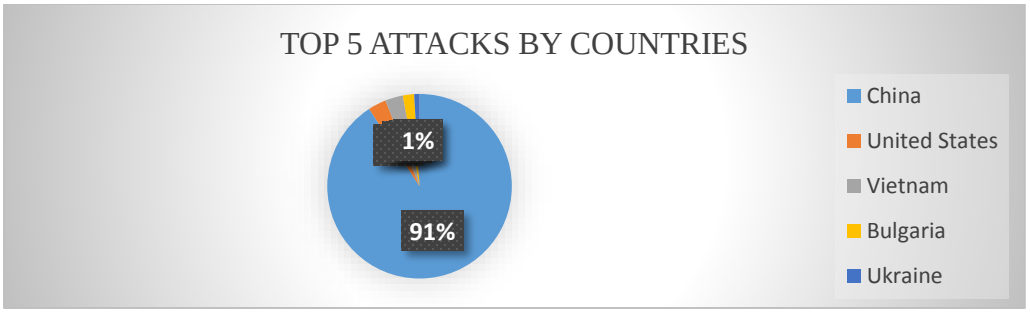

Figure 15: The Graph of Top 5 Countries Attacks

Figure 15 shows the graph of the attacks on honeypot 1 from the top five countries. Clearly, China produced the most attacks (90.81%) and Ukraine generated the least attacks (0.93%). Consequently, China had the high incidence of attacks on honeypot 2.

Table 6: Attacks Per Day

Timestamp	Frequency	Percentage %
4/4/2016	2	.10%
4/5/2016	105	5.36%
4/6/2016	15	0.77%
4/7/2016	5	0.26%
4/8/2016	107	5.46%
4/9/2016	96	4.90%
4/10/2016	493	25.15%
4/11/2016	22	1.12%
4/12/2016	177	9.03%
4/13/2016	938	47.86%
TOTAL	1956	100.0%

$X^2 = 1477.188776$
P –Value = < 0.0001

Table 6 shows the frequencies and percentages of top ten daily attacks on honeypot 1. The Chi square value of 1477 for investigating the equality of the proportions of that attacks on these timestamps is significant at the 0.0001 level. There tends to be a high incidence of attacks on 4/13/2016.

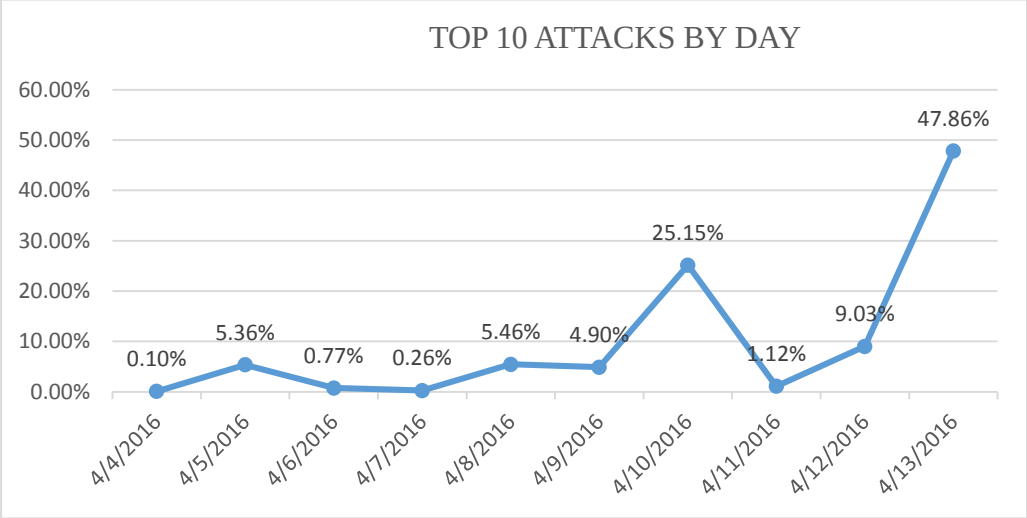

Figure 16: The Graph of Daily Attacks

Figure 16 shows the graph of the attacks on honeypot 2 base on the top ten timestamps. Clearly, 4/13/2016 produced the most attacks (47.86%) and 4/4/2016 generated the least attacks (0.10%). Consequently, 4/13/2016 had the high incidence of attacks on honeypot 2.

Table 7: Top 5 Source IP Attacks

ATTACKS ON HONEYPOT 3		
Top Source IP	**Frequency**	**Percentage %**
218.25.208.124	843	49.73%
222.186.21.211	378	22.30%
121.12.127.94	314	18.53%
59.47.5.235	127	7.49%
119.146.221.68	33	1.95 %
TOTAL	1695	100.0%

$X^2 = 1164.430678$
P –Value = < 0.0001

Table 7 shows the frequencies and percentages of the attacks on honeypot 3 from the top five source IP addresses. The Chi square value of 1164 for investigating the equality of the proportions of that attacks from these IP addresses is significant at the 0.0001 level. There tends to be a high incidence of attacks from IP address 218.25.208.124.

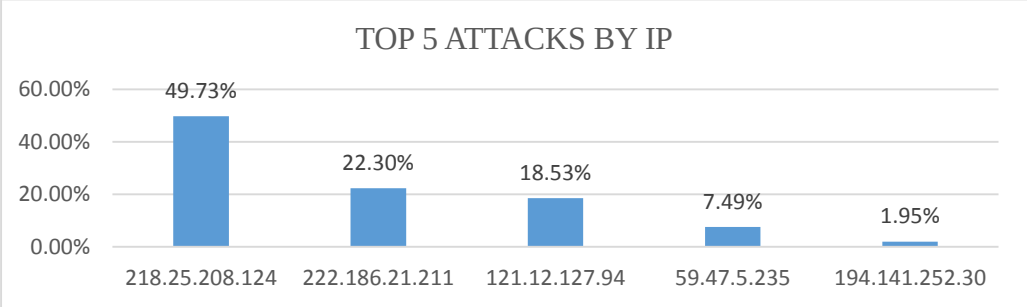

Figure 17: The Graph of Top Source IP Attacks

Figure 17 shows the graph of the attacks on honeypot 3 from the top five source IP addresses. Clearly, the IP address 218.25.208.124 produced the most attacks (49.73%) and the IP address 194.141.252.30 generated the least attacks (1.95%). Consequently, the IP address 218.25.208.124 had the high incidence of attacks on honeypot 3.

Table 8: Top 5 Countries Attacks

Countries	Frequency	Percentage %
China	1789	89.90%
United States	79	3.97%
Viet Nam	65	3.27%
Bulgaria	33	1.66%
Ukraine	24	1.21%
TOTAL	1990	100.0%

$X^2 = 6081.98995$
P –Value = < 0.0001

Table 8 shows the frequencies and percentages of the attacks on honeypot 2 from the top five countries. The Chi square value of 6081 for investigating the equality of the proportions of that attacks from these countries is significant at the 0.0001 level. There tends to be a high incidence of attacks from China

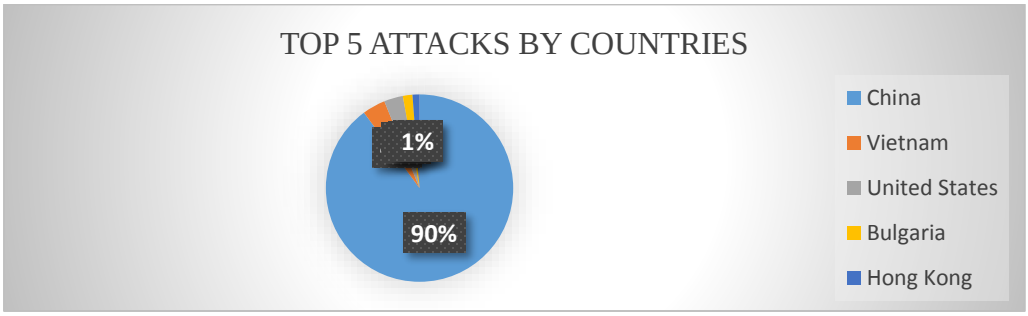

Figure 18: The Graph of Top 5 Countries Attacks

Figure 18 shows the graph of the attacks on honeypot 3 from the top five countries. Clearly, China produced the most attacks (89.90%) and Hong Kong generated the least attacks (1.21%). Consequently, China had the high incidence of attacks on honeypot 3.

Table 9: Attacks Per Day

Timestamp	Frequency	Percentage %
4/4/2016	2	0.10%
4/5/2016	114	5.52%
4/6/2016	40	1.94%
4/7/2016	7	0.34%
4/8/2016	100	4.84%
4/9/2016	93	4.50%
4/10/2016	494	23.92%
4/11/2016	30	1.45%
4/12/2016	199	9.64%
4/13/2016	986	47.75%
TOTAL	2065	100.0%

$X^2 = 1524.878935$
P –Value = < 0.0001

Table 9 shows the frequencies and percentages of top ten daily attacks on honeypot 1. The Chi square value of 1524 for investigating the equality of the proportions of that attacks on these timestamps is significant at the 0.0001 level. There tends to be a high incidence of attacks on 4/13/2016.

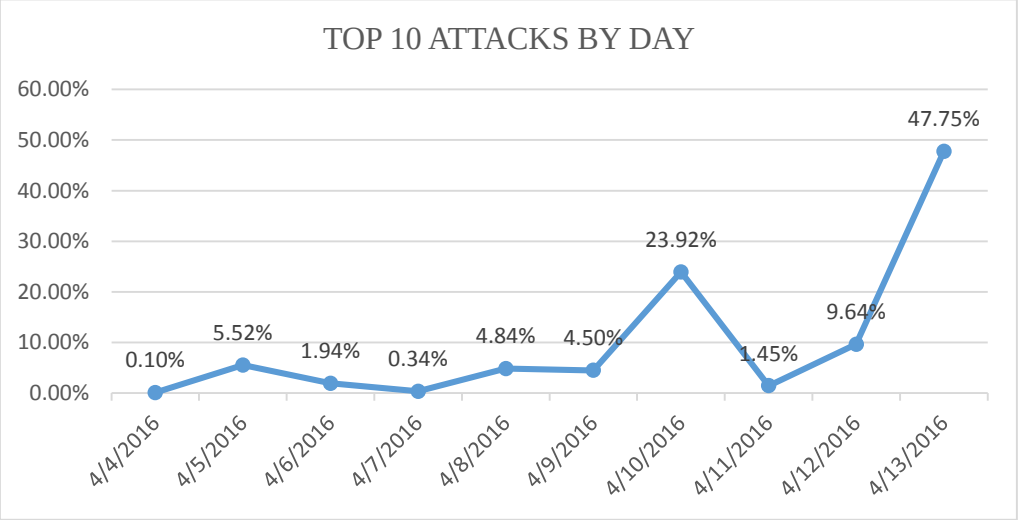

Figure 19: The Graph of Daily Attacks

Figure 19 shows the graph of the attacks on honeypot 3 base on the top ten timestamps. Clearly, 4/13/2016 produced the most attacks (47.75%) and 4/4/2016 generated the least attacks (0.10%). Consequently, 4/13/2016 had the high incidence of attacks on honeypot 3.

Table 10: Top 5 Source IP Attacks

ATTACKS ON HONEYPOT 4		
Top Source IP	**Frequency**	**Percentage %**
218.25.208.124	848	52.35%
222.186.21.211	322	19.88%
121.12.127.94	316	19.51%
59.47.5.235	100	6.17%
119.146.221.68	34	2.10 %
TOTAL	1620	100.0%

$X^2 = 1262.098765$
P –Value = < 0.0001

Table 10 shows the frequencies and percentages of the attacks on honeypot 4 from the top five source IP addresses. The Chi square value of 1262 for investigating the equality of the proportions of that attacks from these IP addresses is significant at the 0.0001 level. There tends to be a high incidence of attacks from IP address 218.25.208.124.

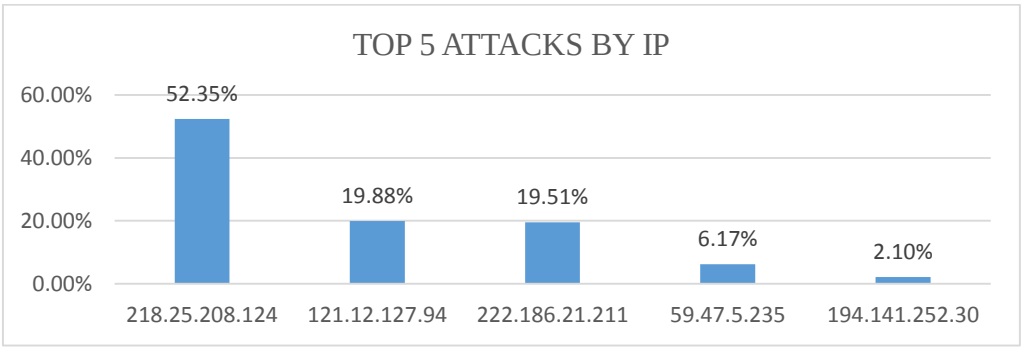

Figure 20: The Graph of Top 5 Source IP

Figure 20 shows the graph of the attacks on honeypot 4 from the top five source IP addresses. Clearly, the IP address 218.25.208.124 produced the most attacks (52.35%) and the IP address

194.141.252.30 generated the least attacks (2.10%). Consequently, the IP address 218.25.208.124 had the high incidence of attacks on honeypot 4.

Table 11: Top 5 Countries Attacks

Countries	Frequency	Percentage %
China	1711	89.86%
United States	85	4.46%
Viet Nam	51	2.68%
Bulgaria	33	1.79%
Ukraine	23	1.21%
TOTAL	1904	100.0%

$X^2 = 5814.046218$
P –Value = < 0.0001

Table 11 shows the frequencies and percentages of the attacks on honeypot 4 from the top five countries. The Chi square value of 5814 for investigating the equality of the proportions of that attacks from these countries is significant at the 0.0001 level. There tends to be a high incidence of attacks from China.

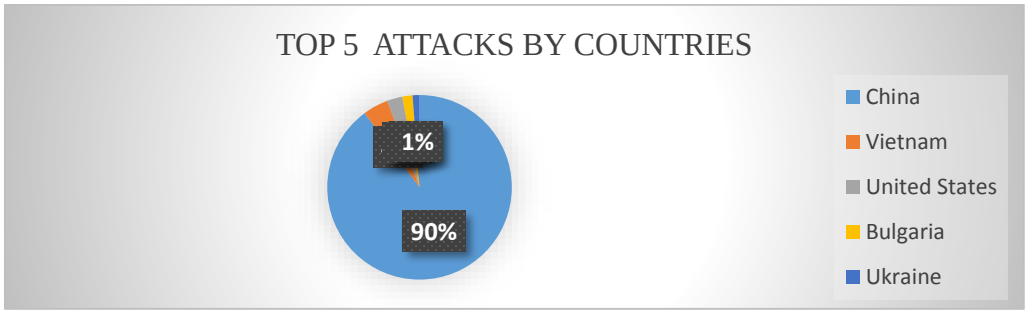

Figure 21: The Graph of Top 5 Countries Attacks

Figure 21 shows the graph of the attacks on honeypot 4 from the top five countries. Clearly, China produced the most attacks (89.86%) and Ukraine generated the least attacks (1.21%). Consequently, China had the high incidence of attacks on honeypot 4.

Table 12: Attacks Per Day

Timestamp	Frequency	Percentage %
4/4/2016	4	0.21%
4/5/2016	96	4.93%
4/6/2016	26	1.33%
4/7/2016	9	0.46%
4/8/2016	98	5.03%
4/9/2016	75	3.85%
4/10/2016	487	24.99%
4/11/2016	19	0.97%
4/12/2016	209	10.72%
4/13/2016	926	47.51%
TOTAL	1949	100.0%

$X^2 = 1452.638789$
P –Value = < 0.0001

Table 12 shows the frequencies and percentages of top ten daily attacks on honeypot 4. The Chi square value of 1452 for investigating the equality of the proportions of that attacks on these timestamps is significant at the 0.0001 level. There tends to be a high incidence of attacks on 4/13/2016.

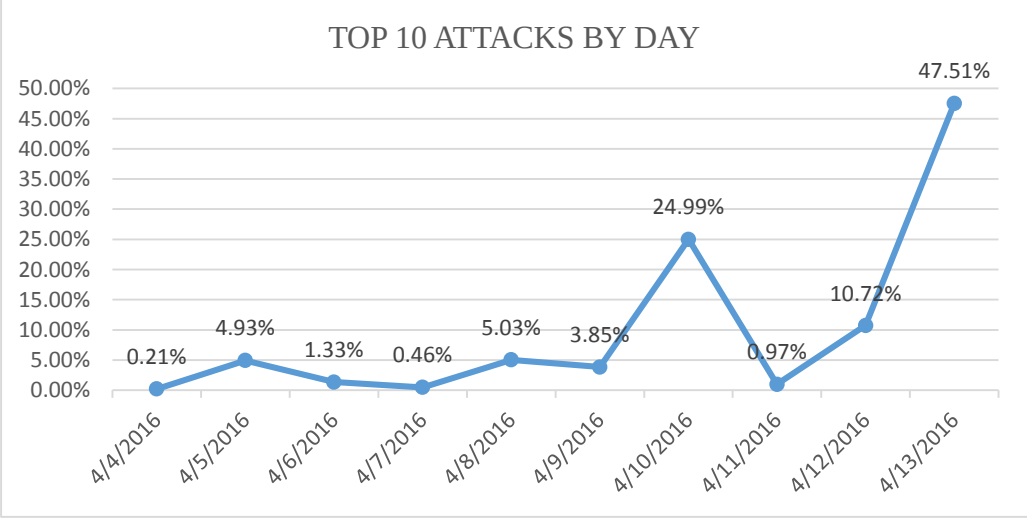

Figure 22: The Graph of Daily Attacks

Figure 22 shows the graph of the attacks on honeypot 4 base on the top ten timestamps. Clearly, 4/13/2016 produced the most attacks (47.51%) and 4/4/2016 generated the least attacks (0.21%). Consequently, 4/13/2016 had the high incidence of attacks on honeypot 4.

Table 13: Top 10 Password

Password	Frequency	Percentage %
admin	80	15.90%
support	73	14.51%
123456	65	12.92%
password	53	10.54%
1234	51	10.14%
root	43	8.55%
user	40	7.95%
ubnt	38	7.55%
default	32	6.36%
test	28	5.57%
TOTAL	503	100.0%

$X^2 = 54.95228628$
P –Value = < 0.0001

Table 13 shows the frequencies and percentages of the attacks on four honeypots from the top ten passwords. The Chi square value of 54 for investigating the equality of the proportions of passwords used in the attacks is significant at the 0.0001 level. There tends to be a high incidence of attacks using root.

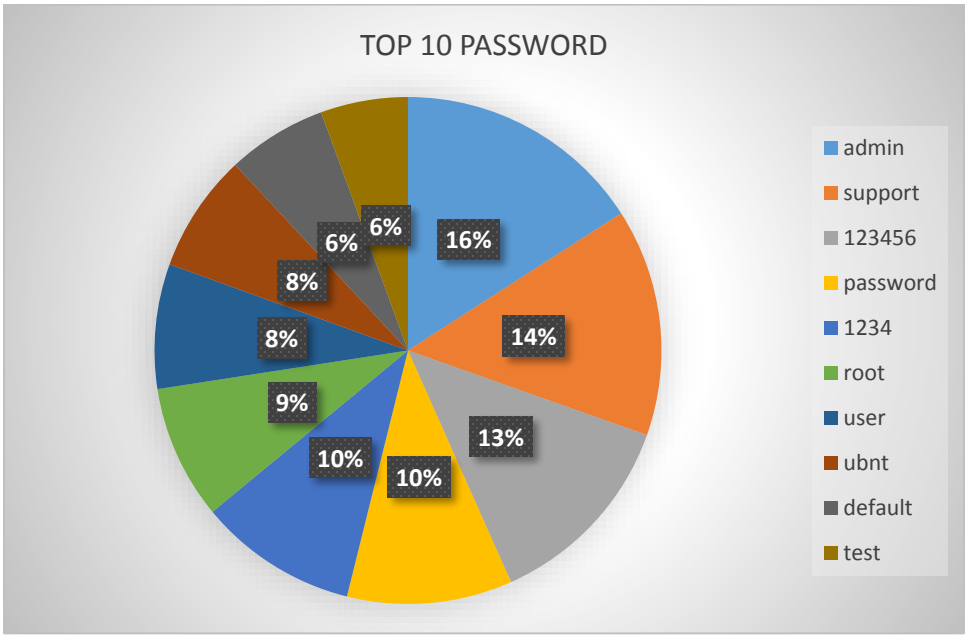

Figure 23: The Graph of Top Password Combined

Figure 23 shows the graph of the attacks on the honeypots from the top ten password. Clearly, admin was used the most (15.90%) and test was used the least (5.57%). Consequently, admin had the high incidence of attacks on honeypots.

Table 14: The Top 10 Username Combined

Username	Frequency	Percentage %
root	2867	83.13%
admin	340	9.86%
support	56	1.62%
user	48	1.39%
ubnt	46	1.33%
test	30	0.87%
1234	22	0.64%
guest	15	0.43%
ubuntu	13	0.38%
oracle	12	0.35%
TOTAL	3449	100.0%

$X^2 = 20745.7434$
P –Value = < 0.0001

Table 14 shows the frequencies and percentages of the attacks on four honeypots from the top ten usernanes. The Chi square value of 20745 for investigating the equality of the proportions of usernanes used in the attacks is significant at the 0.0001 level. There tends to be a high incidence of attacks using root.

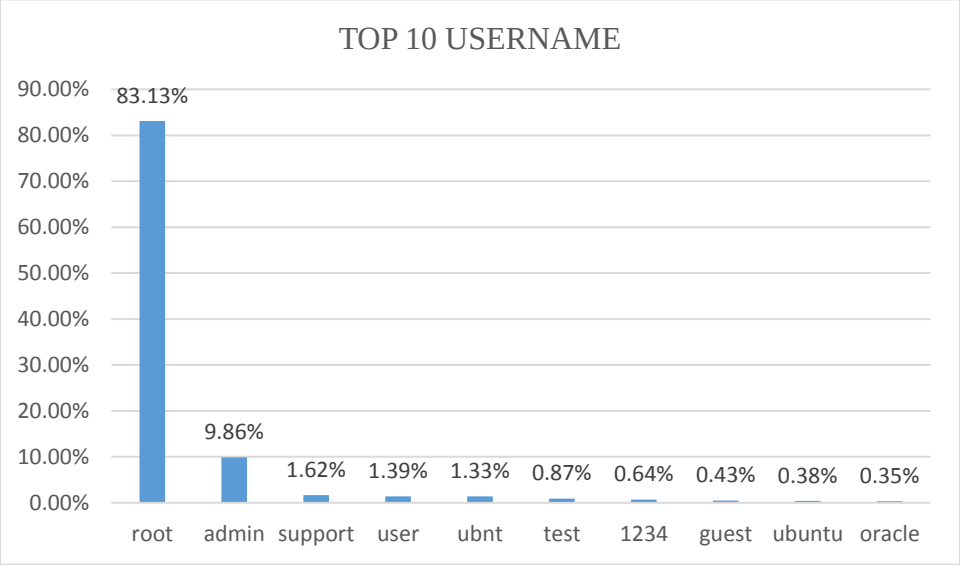

Figure 24: The Graph of Top 10 Username Combined

Figure 24 shows the graph of the attacks on the honeypots from the top ten usernames. Clearly, root was used the most (83.13%) and oracle was used the least (0.35%). Consequently, root had the high incidence of attacks on honeypots.

Table 15: Top 10 Source Port

	ALL	
Source Port	**Frequency**	**Percentage%**
20974	23	24.47%
1779	13	13.83%
1782	10	10.64%
1780	9	9.57%
1781	8	8.51%
1688	7	7.45%
1783	7	7.45%
1516	6	6.38%
2100	6	6.38%
1450	5	5.32%
TOTAL	94	100.0%

$X^2 = 27.06382979$
P –Value = < 0.0014

Table 14 shows the top 10 source port used by attackers on all the honeypots, with corresponding frequencies, percentages. Table 14 shows the frequencies and percentages of the attacks on four honeypots from the top 10 sources port. The Chi square value of 27 for investigating the equality of the proportions of sources port used in the attacks is significant at the 0.0014 level. There tends to be a high incidence of attacks using port 20974.

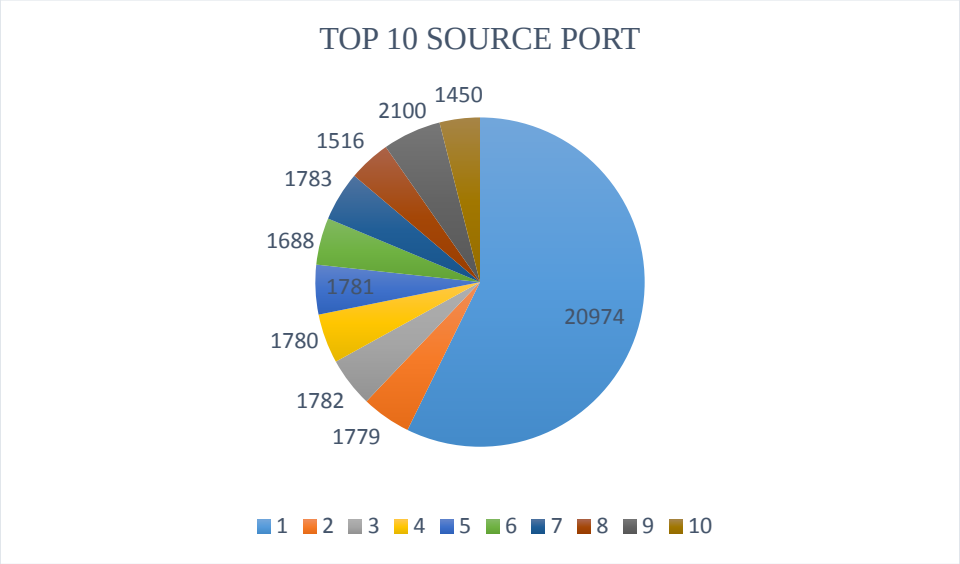

Figure 25: The Graph of Top 10 Source Port

Figure 25 shows the graph of the attacks on honeypots from the top ten source port. Clearly, port 20974 was used for most attacks (24.47%) and port 1450 was used for least attacks (5.32%). Consequently, port 20974 had the high incidence of attacks on the honeypots.

Attack Summary

The results of this study the shows the top five IP addresses from the world, the top five countries of origin of the attacks, top 10 attacks by day, top 10 passwords and usernames used by the intruders and the top ten sources port used for the attacks. The protocol used by attackers was SSH. The study encountered two problems. Firstly, shipping the Honssh logs to the logstash server for processing and indexing. It was resolved by installing Filebeat on the Bifrozt Server. Secondly, setting up a grok patterns for logstash was challenging at the beginning but was overcome by learning how to write the patterns based on the logs captured.

Chapter 5: Conclusions and Future Work

Introduction

This chapter summarizes the methodology of the experiments used in the thesis. The implications of the research results from the use of a real-time honeynet system to detect and prevent attacks are discussed.

Conclusions

The research designed and implemented a real-time Honeynet system for detecting and preventing system attacks. System services on Apache Webserver, MYSQL, FTP and SMTP were used to lure attackers.

The experiment was conducted for 10 days. During this period, almost 500,000 attacks were captured. The results of this study shows the top five IP addresses from the world; the top five countries of origin of the attacks, top ten attacks by day, top ten passwords and usernames used by the intruders, and the top ten sources port used for the attacks.

The question naturally arises on how to prevent high incidence of attacks. For example, the results show high incidence of attacks from China with an IP address of 218.25.208.124. Essentially a bash script such as the one shown below is required to fetch the IP address of China or from any country with high incidence of attack and block it.

```
#!/bin/bash
###PUT HERE COMA SEPARATED LIST OF COUNTRY CODE###
COUNTRIES="AK,AR,BR,CN"
WORKDIR="/root"
************************************************************
cd $WORKDIR
wget -c --output-document=iptables-blocklist.txt http://blogama.org/country_query.php?country=$COUNTRIES
if [ -f iptables-blocklist.txt ]; then
  iptables -F
```

```
  BLOCKDB="iptables-blocklist.txt"
  IPS=$(grep -Ev "^#" $BLOCKDB)
  for i in $IPS
  do
    iptables -A INPUT -s $i -j DROP
    iptables -A OUTPUT -d $i -j DROP
  done
fi
rm $WORKDIR/iptables-blocklist.txt
```

The research results are useful for addressing two main questions:

Q1. How should open source technologies be used to dynamically add or modify hacking incidences in a high-interaction honeynet system?

A1. Open-source technologies and methods were used to reduce the amount of manual labor required to add to or modify a high-interaction honeynet system.

Q2. How should honeypots be made more attractive for hackers to spend more time to provide hacking evidences?

A2. The honeypots attracted voluminous attacks and evidences of passwords, usernames, and source ports from several countries. However, the File Transfer Protocol server on Honeypot 3 appeared more lucrative for attackers, it attracted the most attacks of 2065 events.

Future Work

There is more research work that can be conducted to reduce the amount of manual intervention required to add to or modify new honeypots to the honeynet system. In particular, further study can pursue the use of a centralized system management called Ansible to capture and record all activities performed on the honeypots. The design of countermeasure algorithms

for redirecting all real attacks to an isolated honeypot for visual inspection by network administrator is an intesting future areas for research. Further studies are required to establish threshold boundary values of each type of network attacks.

References

Akkaya, D., & Thalgott, F. (2010). *Honeypots in network security.* Retrieved from http://www.diva-portal.org/smash/get/diva2:327476/fulltext01.

Anicas, M. (2015). *How to install elasticsearch, logstash, and kibana (ELK Stack) on Ubuntu 14.04*. Retrieved from Digital Ocean: https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elk-stack-on-ubuntu-14-04.

Dittrich, D. (2004). Cr*eating and managing distributed honeynets using honeywalls.* Draft. University of Washington.

Döring, C. (2005). *Improving network security with honeypots.* Darmstadt: University of Applied Sciences.

Hoque, M. S., & Bikas, M. A. (2012). An implementation of intrusion dectection system using genetic algorithm. *International Journal of Network Security & Its Applications (IJNSA)*.

Jaiganesh, V., Sumathi, D. P., & A.Vinitha. (2013). *Classification algorithms in intrusion detection system: A survey.* A Vinitha et al. Int. J. Computer Technology & Applications.

Kaur, T., Malhotra, V., & Singh, D. D. (2014). *Comparison of network security tools.* Firewall intrusion detection system and honeypot, 202.

Krawetz, N. (2004). Anti-honeypot technology. *IEEE Security & Privacy*, pp. 76-79.

Liston, T. (2002, February 12). *Tom Liston talks about LaBrea*. Retrieved from http://labrea.sourceforge.net/Intro-History.html.

Sahu, N., & Richhariya, V. (2012). Honeypot: A survey. *International Journal of Computer Science and Technology*.

Sobesto, B., Cukier, M., Hiltunen, M., Kormann, D., & Vesonder, G. (2011). DarkNOC: Dashboard for honeypot management. *USENIX Association,* 16, 16 .

Spitzner, L. (2002). *Tracking hackers.* Boston, MA: Addison-Wesley.

Spitzner, L. (2003). The honeynet project: Trapping the hackers. *IEEE Security & Privacy, 1*(2), *15-23.*

Stiawan, D., Abdullah, A. H., & Idris, M. Y. (2011). Characterizing network intrusion prevention system. *International Journal of Computer Applications (0975–8887).*

Stockman, M., Rein, R., & Heile, A. (2015). An open-source honeynet system to study system banner message effects on hackers. *Journal of Computing Sciences in Colleges*, pp. 282-293.

Virvilis, N., Serrano, O. S., & Vanautgaerden, B. (2014). Changing the game: The art of deceiving sophisticated attackers. *NATO CCD COE Publications*.

Weiler, N. (2002). Honeypots for distributed denial of service attacks. *IEEE Computer Society,* 109-114.

Wilson, T., Maimon, D., Sobesto, B., & Cukier, M. (2015). The effect of a surveillance banner in an attacked computer system: Additional evidence for the relevance of restrictive deterrence in cyberspace. *Journal of Research in Crime and Delinquency*.

Appendix

Configuration file for SSH Server

```
class ssh (
  $hiera_merge                  = false,
  $packages                     = 'USE_DEFAULTS',
  $permit_root_login            = 'yes',
  $purge_keys                   = true,
  $manage_firewall              = false,
  $ssh_package_source           = 'USE_DEFAULTS',
  $ssh_package_adminfile        = 'USE_DEFAULTS',
  $ssh_config_hash_known_hosts  = 'USE_DEFAULTS',
  $ssh_config_path              = '/etc/ssh/ssh_config',
  $ssh_config_owner             = 'root',
  $ssh_config_group             = 'root',
  $ssh_config_mode              = '0644',
  $ssh_config_forward_x11       = undef,
  $ssh_config_forward_x11_trusted = 'USE_DEFAULTS',
  $ssh_config_forward_agent     = undef,
  $ssh_config_server_alive_interval = undef,
  $ssh_config_sendenv_xmodifiers = false,
  $ssh_config_ciphers           = undef,
  $ssh_config_macs              = undef,
  $ssh_config_use_roaming       = 'USE_DEFAULTS',
  $ssh_config_template          = 'ssh/ssh_config.erb',
  $ssh_sendenv                  = 'USE_DEFAULTS',
  $ssh_gssapiauthentication     = 'yes',
  $ssh_gssapidelegatecredentials = undef,
  $sshd_config_path             = '/etc/ssh/sshd_config',
  $sshd_config_owner            = 'root',
```

```
$sshd_config_group              = 'root',
$sshd_config_loglevel           = 'INFO',
$sshd_config_mode               = 'USE_DEFAULTS',
$sshd_config_port               = '22',
$sshd_config_syslog_facility    = 'AUTH',
$sshd_config_template           = 'ssh/sshd_config.erb',
$sshd_config_login_grace_time   = '120',
$sshd_config_challenge_resp_auth = 'yes',
$sshd_config_print_motd         = 'yes',
$sshd_config_use_dns            = 'USE_DEFAULTS',
$sshd_config_authkey_location   = undef,
$sshd_config_strictmodes        = undef,
$sshd_config_serverkeybits      = 'USE_DEFAULTS',
$sshd_config_banner             = 'none',
$sshd_config_ciphers            = undef,
$sshd_config_macs               = undef,
$sshd_config_allowgroups        = [],
$sshd_config_allowusers         = [],
$sshd_config_denygroups         = [],
$sshd_config_denyusers          = [],
$sshd_config_maxstartups        = undef,
$sshd_config_maxsessions        = undef,
$sshd_config_chrootdirectory    = undef,
$sshd_config_forcecommand       = undef,
$sshd_config_match              = undef,
$sshd_config_forcecommand       = undef,
$sshd_config_match              = undef,
$sshd_authorized_keys_command   = undef,
$sshd_authorized_keys_command_user = undef,
$sshd_banner_content            = undef,
```

$sshd_banner_owner = 'root',
$sshd_banner_group = 'root',
$sshd_banner_mode = '0644',
$sshd_config_xauth_location = 'USE_DEFAULTS',
$sshd_config_subsystem_sftp = 'USE_DEFAULTS',
$sshd_kerberos_authentication = undef,
$sshd_password_authentication = 'yes',
$sshd_allow_tcp_forwarding = 'yes',
$sshd_x11_forwarding = 'yes',
$sshd_use_pam = 'USE_DEFAULTS',
$sshd_client_alive_count_max = '3',
$sshd_client_alive_interval = '0',
$sshd_gssapiauthentication = 'yes',
$sshd_gssapikeyexchange = 'USE_DEFAULTS',
$sshd_pamauthenticationviakbdint = 'USE_DEFAULTS',
$sshd_gssapicleanupcredentials = 'USE_DEFAULTS',
$sshd_acceptenv = 'USE_DEFAULTS',
$sshd_config_hostkey = 'USE_DEFAULTS',
$sshd_listen_address = undef,
$sshd_hostbasedauthentication = 'no',
$sshd_ignoreuserknownhosts = 'no',
$sshd_ignorerhosts = 'yes',
$manage_service = true,
$sshd_addressfamily = 'any',
$service_ensure = 'running',
$service_name = 'USE_DEFAULTS',
$service_enable = true,
$service_hasrestart = true,
$service_hasstatus = 'USE_DEFAULTS',
$ssh_key_ensure = 'present',

```
  $ssh_key_import                 = true,
  $ssh_key_type                   = 'ssh-rsa',
  $ssh_config_global_known_hosts_file  = '/etc/ssh/ssh_known_hosts',
  $ssh_config_global_known_hosts_owner = 'root',
  $ssh_config_global_known_hosts_group = 'root',
  $ssh_config_global_known_hosts_mode  = '0644',
  $keys                           = undef,
  $manage_root_ssh_config         = false,
  $root_ssh_config_content        = "# This file is being maintained by Puppet.\n# DO
NOT $
) {

  case $::osfamily {
    'RedHat': {
      $default_packages                 = ['openssh-server',
                                           'openssh-clients']
      $default_service_name             = 'sshd'
      $default_ssh_config_hash_known_hosts     = 'no'
      $default_ssh_config_forward_x11_trusted  = 'yes'
      $default_ssh_package_source       = undef
      $default_ssh_package_adminfile    = undef
      $default_ssh_sendenv              = true
      $default_ssh_package_adminfile    = undef
      $default_ssh_sendenv              = true
      $default_ssh_config_use_roaming   = 'no'
      $default_sshd_config_subsystem_sftp      = '/usr/libexec/openssh/sftp-server'
      $default_sshd_config_mode         = '0600'
      $default_sshd_config_use_dns      = 'yes'
      $default_sshd_config_xauth_location      = '/usr/bin/xauth'
      $default_sshd_use_pam             = 'yes'
```

```
  $default_sshd_gssapikeyexchange          = undef
  $default_sshd_pamauthenticationviakbdint = undef
  $default_sshd_gssapicleanupcredentials   = 'yes'
  $default_sshd_acceptenv                  = true
  $default_service_hasstatus               = true
  $default_sshd_config_serverkeybits       = '1024'
  $default_sshd_config_hostkey             = [ '/etc/ssh/ssh_host_rsa_key' ]
}
'Suse': {
  $default_packages                    = 'openssh'
  $default_service_name                = 'sshd'
  $default_ssh_config_hash_known_hosts     = 'no'
  $default_ssh_package_source              = undef
  $default_ssh_package_adminfile           = undef
  $default_ssh_sendenv                 = true
  $default_ssh_config_use_roaming          = 'no'
  $default_ssh_config_forward_x11_trusted  = 'yes'
  $default_sshd_config_mode                = '0600'
  $default_sshd_config_use_dns             = 'yes'
  $default_sshd_config_xauth_location      = '/usr/bin/xauth'
  $default_sshd_use_pam                = 'yes'
  $default_sshd_gssapikeyexchange          = undef
  $default_sshd_pamauthenticationviakbdint = undef
  $default_sshd_gssapicleanupcredentials   = 'yes'
  $default_sshd_acceptenv                  = true
  $default_service_hasstatus               = true
  $default_sshd_config_serverkeybits       = '1024'
  $default_sshd_config_hostkey             = [ '/etc/ssh/ssh_host_rsa_key' ]
  case $::architecture {
    'x86_64': {
```

```
      if ($::operatingsystem == 'SLES') and ($::operatingsystemrelease =~ /^12\./) {
        $default_sshd_config_subsystem_sftp = '/usr/lib/ssh/sftp-server'
      } else {
        $default_sshd_config_subsystem_sftp = '/usr/lib64/ssh/sftp-server'
      }
    }
    'i386' : {
      $default_sshd_config_subsystem_sftp = '/usr/lib/ssh/sftp-server'
   }
    default: {
      fail("ssh supports architectures x86_64 and i386 for Suse. Detected architecture is
<$
    }
   }
  }
  'Debian': {
   $default_packages                = ['openssh-server',
                                       'openssh-client']
   $default_service_name            = 'ssh'
                                       'openssh-client']
   $default_service_name            = 'ssh'
   $default_ssh_config_forward_x11_trusted  = 'yes'
   $default_ssh_config_hash_known_hosts     = 'no'
   $default_ssh_package_source          = undef
   $default_ssh_package_adminfile       = undef
   $default_ssh_sendenv                 = true
   $default_ssh_config_use_roaming      = 'no'
   $default_sshd_config_subsystem_sftp  = '/usr/lib/openssh/sftp-server'
   $default_sshd_config_mode            = '0600'
   $default_sshd_config_use_dns         = 'yes'
```

```
  $default_sshd_config_xauth_location      = '/usr/bin/xauth'
  $default_sshd_use_pam                    = 'yes'
  $default_sshd_gssapikeyexchange          = undef
  $default_sshd_pamauthenticationviakbdint = undef
  $default_sshd_gssapicleanupcredentials   = 'yes'
  $default_sshd_acceptenv                  = true
  $default_service_hasstatus               = true
  $default_sshd_config_serverkeybits       = '1024'
  $default_sshd_config_hostkey             = [ '/etc/ssh/ssh_host_rsa_key' ]
 }
 'Solaris': {
  $default_ssh_config_hash_known_hosts     = undef
  $default_ssh_sendenv                     = false
  $default_ssh_config_forward_x11_trusted  = undef
  $default_ssh_config_use_roaming          = 'unset'
  $default_sshd_config_subsystem_sftp      = '/usr/lib/ssh/sftp-server'
  $default_sshd_config_mode                = '0644'
  $default_sshd_config_use_dns             = undef
  $default_sshd_config_xauth_location      = '/usr/openwin/bin/xauth'
  $default_sshd_use_pam                    = undef
  $default_sshd_gssapikeyexchange          = 'yes'
  $default_sshd_pamauthenticationviakbdint = 'yes'
  $default_sshd_gssapicleanupcredentials   = undef
  $default_sshd_acceptenv                  = false
  $default_sshd_config_serverkeybits       = '768'
  $default_ssh_package_adminfile           = undef
  $default_sshd_config_hostkey             = [ '/etc/ssh/ssh_host_rsa_key' ]
  case $::kernelrelease {
    '5.11': {
      $default_packages                    = ['network/ssh',
```

```
                                'network/ssh/ssh-key',
                                'service/network/ssh']
  $default_service_name             = 'ssh'
  $default_service_hasstatus         = true
  $default_ssh_package_source          = undef
}
'5.10': {
  $default_packages                 = ['SUNWsshcu',
                                'SUNWsshdr',
                                'SUNWsshdu',
                                'SUNWsshr',
                                'SUNWsshu']
  $default_service_name             = 'ssh'
  $default_service_hasstatus         = true
  $default_ssh_package_source          = '/var/spool/pkg'
  $default_service_hasstatus         = true
  $default_ssh_package_source          = '/var/spool/pkg'
}
'5.9' : {
  $default_packages                 = ['SUNWsshcu',
                                'SUNWsshdr',
                                'SUNWsshdu',
                                'SUNWsshr',
                                'SUNWsshu']
  $default_service_name             = 'sshd'
  $default_service_hasstatus         = false
  $default_ssh_package_source          = '/var/spool/pkg'
}
default: {
  fail('ssh module supports Solaris kernel release 5.9, 5.10 and 5.11.')
```

```
      }
    }
  }
  default: {
    fail("ssh supports osfamilies RedHat, Suse, Debian and Solaris. Detected osfamily is
<${:$
  }
}

if $packages == 'USE_DEFAULTS' {
  $packages_real = $default_packages
} else {
  $packages_real = $packages
}

if $ssh_config_hash_known_hosts == 'USE_DEFAULTS' {
  $ssh_config_hash_known_hosts_real = $default_ssh_config_hash_known_hosts
} else {
  $ssh_config_hash_known_hosts_real = $ssh_config_hash_known_hosts
}

if $service_name == 'USE_DEFAULTS' {
  $service_name_real = $default_service_name
} else {
  $service_name_real = $service_name
}

if $sshd_config_subsystem_sftp == 'USE_DEFAULTS' {
  $sshd_config_subsystem_sftp_real = $default_sshd_config_subsystem_sftp
} else {
```

```
  $sshd_config_subsystem_sftp_real = $sshd_config_subsystem_sftp
}

if $sshd_config_mode    == 'USE_DEFAULTS' {
  $sshd_config_mode_real = $default_sshd_config_mode
} else {
  $sshd_config_mode_real = $sshd_config_mode
}

if $sshd_config_xauth_location == 'USE_DEFAULTS' {
  $sshd_config_xauth_location_real = $default_sshd_config_xauth_location
} else {
  $sshd_config_xauth_location_real = $default_sshd_config_xauth_location
} else {
  $sshd_config_xauth_location_real = $sshd_config_xauth_location
}

if $sshd_config_xauth_location_real != undef {
  validate_absolute_path($sshd_config_xauth_location_real)
}

if $ssh_package_source == 'USE_DEFAULTS' {
  $ssh_package_source_real = $default_ssh_package_source
} else {
  $ssh_package_source_real = $ssh_package_source
}

if $ssh_package_source_real != undef {
  validate_absolute_path($ssh_package_source_real)
}
```

```
if $ssh_package_adminfile == 'USE_DEFAULTS' {
  $ssh_package_adminfile_real = $default_ssh_package_adminfile
} else {
  $ssh_package_adminfile_real = $ssh_package_adminfile
}

if $ssh_package_adminfile_real != undef {
  validate_absolute_path($ssh_package_adminfile_real)
}

if $sshd_config_use_dns == 'USE_DEFAULTS' {
  $sshd_config_use_dns_real = $default_sshd_config_use_dns
} else {
  $sshd_config_use_dns_real = $sshd_config_use_dns
}

if $sshd_use_pam == 'USE_DEFAULTS' {
  $sshd_use_pam_real = $default_sshd_use_pam
} else {
  $sshd_use_pam_real = $sshd_use_pam
}

if $sshd_config_serverkeybits == 'USE_DEFAULTS' {
  $sshd_config_serverkeybits_real = $default_sshd_config_serverkeybits
} else {
  $sshd_config_serverkeybits_real = $sshd_config_serverkeybits
}

if $ssh_config_forward_x11_trusted == 'USE_DEFAULTS' {
```

```
    $ssh_config_forward_x11_trusted_real = $default_ssh_config_forward_x11_trusted
  } else {
    $ssh_config_forward_x11_trusted_real = $ssh_config_forward_x11_trusted
  }
  if $ssh_config_forward_x11_trusted_real != undef {
    validate_re($ssh_config_forward_x11_trusted_real, '^(yes|no)$',
"ssh::ssh_config_forward_x1$
  }
    $sshd_config_xauth_location_real = $default_sshd_config_xauth_location
  } else {
    $sshd_config_xauth_location_real = $sshd_config_xauth_location
  }

  if $sshd_config_xauth_location_real != undef {
    validate_absolute_path($sshd_config_xauth_location_real)
  }

  if $ssh_package_source == 'USE_DEFAULTS' {
    $ssh_package_source_real = $default_ssh_package_source
  } else {
    $ssh_package_source_real = $ssh_package_source
  }

  if $ssh_package_source_real != undef {
    validate_absolute_path($ssh_package_source_real)
  }

  if $ssh_package_adminfile == 'USE_DEFAULTS' {
    $ssh_package_adminfile_real = $default_ssh_package_adminfile
  } else {
```

```
  $ssh_package_adminfile_real = $ssh_package_adminfile
}

if $ssh_package_adminfile_real != undef {
  validate_absolute_path($ssh_package_adminfile_real)
}

if $sshd_config_use_dns == 'USE_DEFAULTS' {
  $sshd_config_use_dns_real = $default_sshd_config_use_dns
} else {
  $sshd_config_use_dns_real = $sshd_config_use_dns
}

if $sshd_use_pam == 'USE_DEFAULTS' {
  $sshd_use_pam_real = $default_sshd_use_pam
} else {
  $sshd_use_pam_real = $sshd_use_pam
}

if $sshd_config_serverkeybits == 'USE_DEFAULTS' {
  $sshd_config_serverkeybits_real = $default_sshd_config_serverkeybits
} else {
  $sshd_config_serverkeybits_real = $sshd_config_serverkeybits
}

if $ssh_config_forward_x11_trusted == 'USE_DEFAULTS' {
  $ssh_config_forward_x11_trusted_real = $default_ssh_config_forward_x11_trusted
} else {
  $ssh_config_forward_x11_trusted_real = $ssh_config_forward_x11_trusted
}
```

```
    if $ssh_config_forward_x11_trusted_real != undef {
      validate_re($ssh_config_forward_x11_trusted_real, '^(yes|no)$',
"ssh::ssh_config_forward_x1$
     }

    if $sshd_gssapikeyexchange == 'USE_DEFAULTS' {
      $sshd_gssapikeyexchange_real = $default_sshd_gssapikeyexchange
    } else {
      $sshd_gssapikeyexchange_real = $sshd_gssapikeyexchange
    }

    if $sshd_pamauthenticationviakbdint == 'USE_DEFAULTS' {
      $sshd_pamauthenticationviakbdint_real = $default_sshd_pamauthenticationviakbdint
    } else {
      $sshd_pamauthenticationviakbdint_real = $sshd_pamauthenticationviakbdint
    }

    if $sshd_gssapicleanupcredentials == 'USE_DEFAULTS' {
      $sshd_gssapicleanupcredentials_real = $default_sshd_gssapicleanupcredentials
    } else {
      $sshd_gssapicleanupcredentials_real = $sshd_gssapicleanupcredentials
    }

    if $ssh_config_use_roaming == 'USE_DEFAULTS' {
      $ssh_config_use_roaming_real = $default_ssh_config_use_roaming
    } else {
      $ssh_config_use_roaming_real = $ssh_config_use_roaming
    }

    if $ssh_sendenv == 'USE_DEFAULTS' {
```

```
    $ssh_sendenv_real = $default_ssh_sendenv
  } else {
    case type3x($ssh_sendenv) {
      'string': {
        validate_re($ssh_sendenv, '^(true|false)$', "ssh::ssh_sendenv may be either 'true' or
'$
        $ssh_sendenv_real = str2bool($ssh_sendenv)
      }
      'boolean': {
        $ssh_sendenv_real = $ssh_sendenv
      }
      default: {
        fail('ssh::ssh_sendenv type must be true or false.')
      }
    }
  }

  if $sshd_acceptenv == 'USE_DEFAULTS' {
    $sshd_acceptenv_real = $default_sshd_acceptenv
  } else {
    case type3x($sshd_acceptenv) {
      'string': {
        validate_re($sshd_acceptenv, '^(true|false)$', "ssh::sshd_acceptenv may be either
'true$
        $sshd_acceptenv_real = str2bool($sshd_acceptenv)
      }
      'boolean': {
        $sshd_acceptenv_real = $sshd_acceptenv
      }
      default: {
```

```
      fail('ssh::sshd_acceptenv type must be true or false.')
    default: {
      fail('ssh::sshd_acceptenv type must be true or false.')
     }
   }
  }

  if $sshd_config_hostkey == 'USE_DEFAULTS' {
   $sshd_config_hostkey_real = $default_sshd_config_hostkey
  } else {
   validate_array($sshd_config_hostkey)
   validate_absolute_path(join($sshd_config_hostkey))
   $sshd_config_hostkey_real = $sshd_config_hostkey
  }

  if $sshd_listen_address {
   validate_array($sshd_listen_address)
  }

  if $service_hasstatus == 'USE_DEFAULTS' {
   $service_hasstatus_real = $default_service_hasstatus
  } else {
   case type3x($service_hasstatus) {
     'string': {
      validate_re($service_hasstatus, '^(true|false)$', "ssh::service_hasstatus must be
'true$
      $service_hasstatus_real = str2bool($service_hasstatus)
     }
     'boolean': {
      $service_hasstatus_real = $service_hasstatus
```

```
    }
    default: {
      fail('ssh::service_hasstatus must be true or false.')
    }
  }
}

# validate params
if $ssh_config_ciphers != undef {
  validate_array($ssh_config_ciphers)
}

if $sshd_config_ciphers != undef {
  validate_array($sshd_config_ciphers)
}

if $ssh_config_macs != undef {
  validate_array($ssh_config_macs)
}

if $sshd_config_macs != undef {
  validate_array($sshd_config_macs)
}

if $ssh_config_hash_known_hosts_real != undef {
  validate_re($ssh_config_hash_known_hosts_real, '^(yes|no)$',
"ssh::ssh_config_hash_known_ho$
}
validate_re($sshd_config_port, '^\d+$', "ssh::sshd_config_port must be a valid number
and is $
```

```
    }
      validate_re($sshd_config_port, '^\d+$', "ssh::sshd_config_port must be a valid number
and is $
      if $sshd_kerberos_authentication != undef {
        validate_re($sshd_kerberos_authentication, '^(yes|no)$',
"ssh::sshd_kerberos_authentication$
      }
      validate_re($sshd_password_authentication, '^(yes|no)$',
"ssh::sshd_password_authentication m$
      validate_re($sshd_allow_tcp_forwarding, '^(yes|no)$',
"ssh::sshd_allow_tcp_forwarding may be $
      validate_re($sshd_x11_forwarding, '^(yes|no)$', "ssh::sshd_x11_forwarding may be
either 'yes'$
      if $sshd_use_pam_real != undef {
        validate_re($sshd_use_pam_real, '^(yes|no)$', "ssh::sshd_use_pam may be either 'yes'
or 'no$
      }
      if $sshd_config_serverkeybits_real != undef {
        if is_integer($sshd_config_serverkeybits_real) == false {
fail("ssh::sshd_config_serverkeyb$
      }
      if $ssh_config_use_roaming_real != undef {
        validate_re($ssh_config_use_roaming_real, '^(yes|no|unset)$',
"ssh::ssh_config_use_roaming $
      }
      if is_integer($sshd_client_alive_interval) == false {
fail("ssh::sshd_client_alive_interval m$
      if is_integer($sshd_client_alive_count_max) == false {
fail("ssh::sshd_client_alive_count_max$
```

```
    if $sshd_config_banner != 'none' {
      validate_absolute_path($sshd_config_banner)
    }
    if $sshd_banner_content != undef and $sshd_config_banner == 'none' {
      fail('ssh::sshd_config_banner must be set to be able to use sshd_banner_content.')
    }

    validate_re($ssh_gssapiauthentication, '^(yes|no)$', "ssh::ssh_gssapiauthentication may
be ei$

    if $ssh_gssapidelegatecredentials != undef {
      validate_re($ssh_gssapidelegatecredentials, '^(yes|no)$',
"ssh::ssh_gssapidelegatecredentia$
    }

    validate_re($sshd_gssapiauthentication, '^(yes|no)$', "ssh::sshd_gssapiauthentication
may be $

    if $sshd_gssapikeyexchange_real != undef {
      validate_re($sshd_gssapikeyexchange_real, '^(yes|no)$',
"ssh::sshd_gssapikeyexchange may be$
    }

    if $sshd_pamauthenticationviakbdint_real != undef {
      validate_re($sshd_pamauthenticationviakbdint_real, '^(yes|no)$',
"ssh::sshd_pamauthenticati$
    }

    if $sshd_gssapicleanupcredentials_real != undef {
```

```
    validate_re($sshd_gssapicleanupcredentials_real, '^(yes|no)$',
"ssh::sshd_gssapicleanupcred$
  }

  if $sshd_config_authkey_location != undef {
    validate_string($sshd_config_authkey_location)
  }

  if $sshd_config_maxstartups != undef {
    validate_re($sshd_config_maxstartups,'^(\d+)+(\d+?:\d+?:\d+)?$',
      "ssh::sshd_config_maxstartups may be either an integer or three integers separated
with c$
  }
  if $sshd_config_maxsessions != undef {
    $is_int_sshd_config_maxsessions = is_integer($sshd_config_maxsessions)
    if $is_int_sshd_config_maxsessions == false {
      fail("sshd_config_maxsessions must be an integer. Detected value is
${sshd_config_maxsess$
    }
  }

  if $sshd_config_chrootdirectory != undef {
    validate_absolute_path($sshd_config_chrootdirectory)
  }

  if $sshd_config_forcecommand != undef {
    validate_string($sshd_config_forcecommand)
  }

  if $sshd_authorized_keys_command != undef {
```

```
  validate_absolute_path($sshd_authorized_keys_command)
 }

 if $sshd_authorized_keys_command_user != undef {
  validate_string($sshd_authorized_keys_command_user)
 }

 if $sshd_config_match != undef {
  validate_hash($sshd_config_match)
 }

 if $sshd_config_strictmodes != undef {
  validate_re($sshd_config_strictmodes, '^(yes|no)$', "ssh::sshd_config_strictmodes
may be ei$
 }

 validate_re($sshd_hostbasedauthentication, '^(yes|no)$',
"ssh::sshd_hostbasedauthentication m$

 validate_re($sshd_ignoreuserknownhosts, '^(yes|no)$',
"ssh::sshd_ignoreuserknownhosts may be $

 validate_re($sshd_ignorerhosts, '^(yes|no)$', "ssh::sshd_ignorerhosts may be either
'yes' or $

 case type3x($hiera_merge) {
  'string': {
   validate_re($hiera_merge, '^(true|false)$', "ssh::hiera_merge may be either 'true' or
'fa$
   $hiera_merge_real = str2bool($hiera_merge)
```

```
    }
    'boolean': {
      $hiera_merge_real = $hiera_merge
    }
    default: {
      fail('ssh::hiera_merge type must be true or false.')
    }
  }

  case type3x($ssh_key_import) {
    'string': {
      validate_re($ssh_key_import, '^(true|false)$', "ssh::ssh_key_import may be either
'true' $
      $ssh_key_import_real = str2bool($ssh_key_import)
  validate_re($ssh_key_import, '^(true|false)$', "ssh::ssh_key_import may be either 'true'
$
      $ssh_key_import_real = str2bool($ssh_key_import)
    }
    'boolean': {
      $ssh_key_import_real = $ssh_key_import
    }
    default: {
      fail('ssh::ssh_key_import type must be true or false.')
    }
  }
  validate_bool($ssh_key_import_real)

  case type3x($ssh_config_sendenv_xmodifiers) {
    'string': {
      $ssh_config_sendenv_xmodifiers_real = str2bool($ssh_config_sendenv_xmodifiers)
```

```
    }
    'boolean': {
      $ssh_config_sendenv_xmodifiers_real = $ssh_config_sendenv_xmodifiers
    }
    default: {
      fail('ssh::ssh_config_sendenv_xmodifiers type must be true or false.')
    }
  }

  case $permit_root_login {
    'no', 'yes', 'without-password', 'forced-commands-only': {
      # noop
    }
    default: {
      fail("ssh::permit_root_login may be either 'yes', 'without-password', 'forced-
commands-on$
    }
  }

  case $ssh_key_type {
    'ssh-rsa','rsa': {
      $key = $::sshrsakey
    }
    'ssh-dsa','dsa': {
      $key = $::sshdsakey
    }
    default: {
      fail("ssh::ssh_key_type must be 'ssh-rsa', 'rsa', 'ssh-dsa', or 'dsa' and is
<${ssh_key_t$
    }
```

```
}

validate_absolute_path($ssh_config_global_known_hosts_file)
validate_string($ssh_config_global_known_hosts_owner)
validate_string($ssh_config_global_known_hosts_group)
validate_re($ssh_config_global_known_hosts_mode, '^[0-7]{4}$',
  "ssh::ssh_config_global_known_hosts_mode must be a valid 4 digit mode in octal
notation. De$

if type3x($purge_keys) == 'string' {
  $purge_keys_real = str2bool($purge_keys)
} else {
  $purge_keys_real = $purge_keys
}
$purge_keys_real = $purge_keys
}
validate_bool($purge_keys_real)

if type3x($manage_service) == 'string' {
  $manage_service_real = str2bool($manage_service)
} else {
  $manage_service_real = $manage_service
}
validate_bool($manage_service_real)

if type3x($service_enable) == 'string' {
  $service_enable_real = str2bool($service_enable)
} else {
  $service_enable_real = $service_enable
}
```

```
validate_bool($service_enable_real)

if type3x($service_hasrestart) == 'string' {
  $service_hasrestart_real = str2bool($service_hasrestart)
} else {
  $service_hasrestart_real = $service_hasrestart
}
validate_bool($service_hasrestart_real)

if type3x($manage_root_ssh_config) == 'string' {
  $manage_root_ssh_config_real = str2bool($manage_root_ssh_config)
} else {
  $manage_root_ssh_config_real = $manage_root_ssh_config
}
validate_bool($manage_root_ssh_config_real)

#ssh_config template
validate_string($ssh_config_template)

#sshd_config template
validate_string($sshd_config_template)

#loglevel
$supported_loglevel_vals=['QUIET', 'FATAL', 'ERROR', 'INFO', 'VERBOSE']
validate_re($sshd_config_loglevel, $supported_loglevel_vals)

#enable hiera merging for groups and users
if $hiera_merge_real == true {
  $sshd_config_allowgroups_real = hiera_array('ssh::sshd_config_allowgroups',[])
  $sshd_config_allowusers_real  = hiera_array('ssh::sshd_config_allowusers',[])
```

```
  $sshd_config_denygroups_real  = hiera_array('ssh::sshd_config_denygroups',[])
  $sshd_config_denyusers_real   = hiera_array('ssh::sshd_config_denyusers',[])
 } else {
  $sshd_config_allowgroups_real = $sshd_config_allowgroups
  $sshd_config_allowusers_real  = $sshd_config_allowusers
  $sshd_config_denygroups_real  = $sshd_config_denygroups
  $sshd_config_denyusers_real   = $sshd_config_denyusers
 }

 if $sshd_config_denyusers_real != [] {
if $sshd_config_denyusers_real != [] {
  validate_array($sshd_config_denyusers_real)
 }

 if $sshd_config_denygroups_real != [] {
  validate_array($sshd_config_denygroups_real)
 }

 if $sshd_config_allowusers_real != [] {
  validate_array($sshd_config_allowusers_real)
 }

 if $sshd_config_allowgroups_real != [] {
  validate_array($sshd_config_allowgroups_real)
 }

 package { $packages_real:
  ensure    => installed,
  source    => $ssh_package_source_real,
  adminfile => $ssh_package_adminfile_real,
```

```
}

file { 'ssh_config' :
  ensure  => file,
  path    => $ssh_config_path,
  owner   => $ssh_config_owner,
  group   => $ssh_config_group,
  mode    => $ssh_config_mode,
  content => template($ssh_config_template),
  require => Package[$packages_real],
}

file { 'sshd_config' :
  ensure  => file,
  path    => $sshd_config_path,
  mode    => $sshd_config_mode_real,
  owner   => $sshd_config_owner,
  group   => $sshd_config_group,
  content => template($sshd_config_template),
  require => Package[$packages_real],
}

if $sshd_config_banner != 'none' and $sshd_banner_content != undef {
  file { 'sshd_banner' :
    ensure  => file,
    path    => $sshd_config_banner,
    owner   => $sshd_banner_owner,
    group   => $sshd_banner_group,
    mode    => $sshd_banner_mode,
    content => $sshd_banner_content,
```

```
    require => Package[$packages_real],
  }
}

if $manage_root_ssh_config_real == true {

if $sshd_config_denyusers_real != [] {
  validate_array($sshd_config_denyusers_real)
}

if $sshd_config_denygroups_real != [] {
  validate_array($sshd_config_denygroups_real)
}

if $sshd_config_allowusers_real != [] {
  validate_array($sshd_config_allowusers_real)
}

if $sshd_config_allowgroups_real != [] {
  validate_array($sshd_config_allowgroups_real)
}

package { $packages_real:
  ensure    => installed,
  source    => $ssh_package_source_real,
  adminfile => $ssh_package_adminfile_real,
}

file  { 'ssh_config' :
  ensure  => file,
```

```
  path    => $ssh_config_path,
  owner   => $ssh_config_owner,
  group   => $ssh_config_group,
  mode    => $ssh_config_mode,
  content => template($ssh_config_template),
  require => Package[$packages_real],
}

file  { 'sshd_config' :
  ensure  => file,
  path    => $sshd_config_path,
  mode    => $sshd_config_mode_real,
  owner   => $sshd_config_owner,
  group   => $sshd_config_group,
  content => template($sshd_config_template),
  require => Package[$packages_real],
}

if $sshd_config_banner != 'none' and $sshd_banner_content != undef {
  file { 'sshd_banner' :
    ensure  => file,
    path    => $sshd_config_banner,
    owner   => $sshd_banner_owner,
    group   => $sshd_banner_group,
    mode    => $sshd_banner_mode,
    content => $sshd_banner_content,
    require => Package[$packages_real],
  }
}
```

```
 if $manage_root_ssh_config_real == true {
if $manage_root_ssh_config_real == true {

  include ::common

  common::mkdir_p { "${::root_home}/.ssh": }

  file { 'root_ssh_dir':
    ensure  => directory,
    path    => "${::root_home}/.ssh",
    owner   => 'root',
    group   => 'root',
    mode    => '0700',
    require => Common::Mkdir_p["${::root_home}/.ssh"],
  }

  file { 'root_ssh_config':
    ensure  => file,
    path    => "${::root_home}/.ssh/config",
    content => $root_ssh_config_content,
    owner   => 'root',
    group   => 'root',
    mode    => '0600',
  }
 }

 if $manage_service_real {
  service { 'sshd_service' :
    ensure     => $service_ensure,
    name       => $service_name_real,
```

```
    enable     => $service_enable_real,
    hasrestart => $service_hasrestart_real,
    hasstatus  => $service_hasstatus_real,
    subscribe  => File['sshd_config'],
  }
}

if $manage_firewall == true {
  firewall { '22 open port 22 for SSH':
    action => 'accept',
    dport  => 22,
    proto  => 'tcp',
  }
}

# export each node's ssh key
@@sshkey { $::fqdn :
  ensure => $ssh_key_ensure,
  type   => $ssh_key_type,
  key    => $key,
}

file { 'ssh_known_hosts':
  ensure => file,
  path   => $ssh_config_global_known_hosts_file,
  owner  => $ssh_config_global_known_hosts_owner,
  path   => $ssh_config_global_known_hosts_file,
  owner  => $ssh_config_global_known_hosts_owner,
  group  => $ssh_config_global_known_hosts_group,
  mode   => $ssh_config_global_known_hosts_mode,
```

```
}

# import all nodes' ssh keys
if $ssh_key_import_real == true {
  Sshkey <<||>> {
    target => $ssh_config_global_known_hosts_file,
  }
}

# remove ssh key's not managed by puppet
resources  { 'sshkey':
  purge => $purge_keys_real,
}

# manage users' ssh authorized keys if present
if $keys != undef {
  if $hiera_merge_real == true {
    $keys_real = hiera_hash('ssh::keys')
  } else {
    $keys_real = $keys
    notice('Future versions of the ssh module will default ssh::hiera_merge_real to true')
  }
  validate_hash($keys_real)
  create_resources('ssh_authorized_key', $keys_real)
}

if $sshd_addressfamily != undef {
  validate_re($sshd_addressfamily, '^(any|inet|inet6)$',
    "ssh::sshd_addressfamily can be undef, 'any', 'inet' or 'inet6' and is set to
${sshd_addr$
```

```
  }
}
```

Configuration file for NTP Server

```
class ntp (
  $autoupdate       = $ntp::params::autoupdate,
  $broadcastclient  = $ntp::params::broadcastclient,
  $config           = $ntp::params::config,
  $config_template  = $ntp::params::config_template,
  $disable_auth     = $ntp::params::disable_auth,
  $disable_monitor  = $ntp::params::disable_monitor,
  $fudge            = $ntp::params::fudge,
  $driftfile        = $ntp::params::driftfile,
  $leapfile         = $ntp::params::leapfile,
  $logfile          = $ntp::params::logfile,
  $iburst_enable    = $ntp::params::iburst_enable,
  $keys_enable      = $ntp::params::keys_enable,
  $keys_file        = $ntp::params::keys_file,
  $keys_controlkey  = $ntp::params::keys_controlkey,
  $keys_requestkey  = $ntp::params::keys_requestkey,
  $keys_trusted     = $ntp::params::keys_trusted,
  $minpoll          = $ntp::params::minpoll,
  $maxpoll          = $ntp::params::maxpoll,
  $package_ensure   = $ntp::params::package_ensure,
  $package_manage   = $ntp::params::package_manage,
  $package_name     = $ntp::params::package_name,
  $panic            = $ntp::params::panic,
  $peers            = $ntp::params::peers,
  $preferred_servers = $ntp::params::preferred_servers,
  $restrict         = $ntp::params::restrict,
  $interfaces       = $ntp::params::interfaces,
  $servers          = $ntp::params::servers,
  $service_enable   = $ntp::params::service_enable,
```

```
  $service_ensure    = $ntp::params::service_ensure,
  $service_manage    = $ntp::params::service_manage,
  $service_name      = $ntp::params::service_name,
  $stepout           = $ntp::params::stepout,
  $tinker            = $ntp::params::tinker,
  $udlc              = $ntp::params::udlc,
  $udlc_stratum      = $ntp::params::udlc_stratum,
) inherits ntp::params {

  validate_bool($broadcastclient)
  validate_absolute_path($config)
  validate_string($config_template)
  validate_bool($disable_auth)
  validate_bool($disable_monitor)
  validate_absolute_path($driftfile)
  if $logfile { validate_absolute_path($logfile) }
  if $leapfile { validate_absolute_path($leapfile) }
  validate_bool($iburst_enable)
  validate_bool($keys_enable)
  validate_re($keys_controlkey, ['^\d+$', ''])
  validate_re($keys_requestkey, ['^\d+$', ''])
  validate_array($keys_trusted)
  if $minpoll { validate_numeric($minpoll, 16, 3) }
  if $maxpoll { validate_numeric($maxpoll, 16, 3) }
  validate_string($package_ensure)
  validate_bool($package_manage)
  validate_array($package_name)
 validate_bool($package_manage)
  validate_array($package_name)
  if $panic { validate_numeric($panic, 65535, 0) }
  validate_array($preferred_servers)
  validate_array($restrict)
  validate_array($interfaces)
```

```
  validate_array($servers)
  validate_array($fudge)
  validate_bool($service_enable)
  validate_string($service_ensure)
  validate_bool($service_manage)
  validate_string($service_name)
  if $stepout { validate_numeric($stepout, 65535, 0) }
  validate_bool($tinker)
  validate_bool($udlc)
  validate_array($peers)

  if $autoupdate {
    notice('autoupdate parameter has been deprecated and replaced with package_ensure.  Set
thi$
  }

  # Anchor this as per #8040 - this ensures that classes won't float off and
  # mess everything up.  You can read about this at:
  # http://docs.puppetlabs.com/puppet/2.7/reference/lang_containment.html#known-issues
  anchor { 'ntp::begin': } ->
  class { '::ntp::install': } ->
  class { '::ntp::config': } ~>
  class { '::ntp::service': } ->
  anchor { 'ntp::end': }

}
```

Configuration file for Apache Server

```
class apache (
  $apache_name          = $::apache::params::apache_name,
  $service_name         = $::apache::params::service_name,
  $default_mods         = true,
  $default_vhost        = true,
```

$default_charset = undef,
$default_confd_files = true,
$default_ssl_vhost = false,
$default_ssl_cert = $::apache::params::default_ssl_cert,
$default_ssl_key = $::apache::params::default_ssl_key,
$default_ssl_chain = undef,
$default_ssl_ca = undef,
$default_ssl_crl_path = undef,
$default_ssl_crl = undef,
$default_ssl_crl_check = undef,
$default_type = 'none',
$dev_packages = $::apache::params::dev_packages,
$ip = undef,
$service_enable = true,
$service_manage = true,
$service_ensure = 'running',
$service_restart = undef,
$purge_configs = true,
$purge_vhost_dir = undef,
$purge_vdir = false,
$serveradmin = 'root@localhost',
$sendfile = 'On',
$error_documents = false,
$timeout = '120',
$httpd_dir = $::apache::params::httpd_dir,
$server_root = $::apache::params::server_root,
$conf_dir = $::apache::params::conf_dir,
$confd_dir = $::apache::params::confd_dir,
$vhost_dir = $::apache::params::vhost_dir,
$vhost_enable_dir = $::apache::params::vhost_enable_dir,
$vhost_include_pattern = $::apache::params::vhost_include_pattern,
$mod_dir = $::apache::params::mod_dir,
$mod_enable_dir = $::apache::params::mod_enable_dir,

```
  $mpm_module           = $::apache::params::mpm_module,
  $lib_path             = $::apache::params::lib_path,
  $conf_template        = $::apache::params::conf_template,
  $servername           = $::apache::params::servername,
 $conf_template         = $::apache::params::conf_template,
  $servername           = $::apache::params::servername,
  $pidfile              = $::apache::params::pidfile,
  $rewrite_lock         = undef,
  $manage_user          = true,
  $manage_group         = true,
  $user                 = $::apache::params::user,
  $group                = $::apache::params::group,
  $keepalive            = $::apache::params::keepalive,
  $keepalive_timeout    = $::apache::params::keepalive_timeout,
  $max_keepalive_requests = $::apache::params::max_keepalive_requests,
  $limitreqfieldsize    = '8190',
  $logroot              = $::apache::params::logroot,
  $logroot_mode         = $::apache::params::logroot_mode,
  $log_level            = $::apache::params::log_level,
  $log_formats          = {},
  $ports_file           = $::apache::params::ports_file,
  $docroot              = $::apache::params::docroot,
  $apache_version       = $::apache::version::default,
  $server_tokens        = 'OS',
  $server_signature     = 'On',
  $trace_enable         = 'On',
  $allow_encoded_slashes  = undef,
  $package_ensure       = 'installed',
  $use_optional_includes  = $::apache::params::use_optional_includes,
  $use_systemd          = $::apache::params::use_systemd,
  $mime_types_additional  = $::apache::params::mime_types_additional,
  $file_mode            = $::apache::params::file_mode,
) inherits ::apache::params {
```

```
validate_bool($default_vhost)
validate_bool($default_ssl_vhost)
validate_bool($default_confd_files)
# true/false is sufficient for both ensure and enable
validate_bool($service_enable)
validate_bool($service_manage)
validate_bool($use_optional_includes)

$valid_mpms_re = $apache_version ? {
  '2.4'   => '(event|itk|peruser|prefork|worker)',
  default => '(event|itk|prefork|worker)'
}

if $mpm_module and $mpm_module != 'false' { # lint:ignore:quoted_booleans
  validate_re($mpm_module, $valid_mpms_re)
}

if $allow_encoded_slashes {
  validate_re($allow_encoded_slashes, '(^on$|^off$|^nodecode$)', "${allow_encoded_slashes}
is$
}

# NOTE: on FreeBSD it's mpm module's responsibility to install httpd package.
# NOTE: the same strategy may be introduced for other OSes. For this, you
# should delete the 'if' block below and modify all MPM modules' manifests
# such that they include apache::package class (currently event.pp, itk.pp,
# peruser.pp, prefork.pp, worker.pp).
if $::osfamily != 'FreeBSD' {
# peruser.pp, prefork.pp, worker.pp).
if $::osfamily != 'FreeBSD' {
  package { 'httpd':
    ensure => $package_ensure,
    name   => $apache_name,
```

```
    notify => Class['Apache::Service'],
  }
}
validate_re($sendfile, [ '^[oO]n$' , '^[oO]ff$' ])

# declare the web server user and group
# Note: requiring the package means the package ought to create them and not puppet
validate_bool($manage_user)
if $manage_user {
  user { $user:
    ensure  => present,
    gid     => $group,
    require => Package['httpd'],
  }
}
validate_bool($manage_group)
if $manage_group {
  group { $group:
    ensure  => present,
    require => Package['httpd']
  }
}

validate_apache_log_level($log_level)

class { '::apache::service':
  service_name    => $service_name,
  service_enable  => $service_enable,
  service_manage  => $service_manage,
  service_ensure  => $service_ensure,
  service_restart => $service_restart,
}
```

```
# Deprecated backwards-compatibility
if $purge_vdir {
  warning('Class[\'apache\'] parameter purge_vdir is deprecated in favor of purge_configs')
  $purge_confd = $purge_vdir
} else {
  $purge_confd = $purge_configs
}

# Set purge vhostd appropriately
if $purge_vhost_dir == undef {
  $purge_vhostd = $purge_confd
} else {
  $purge_vhostd = $purge_vhost_dir
}

Exec {
  path => '/bin:/sbin:/usr/bin:/usr/sbin',
}
path => '/bin:/sbin:/usr/bin:/usr/sbin',
}

exec { "mkdir ${confd_dir}":
  creates => $confd_dir,
  require => Package['httpd'],
}
file { $confd_dir:
  ensure  => directory,
  recurse => true,
  purge   => $purge_confd,
  notify  => Class['Apache::Service'],
  require => Package['httpd'],
}
```

```
if ! defined(File[$mod_dir]) {
  exec { "mkdir ${mod_dir}":
    creates => $mod_dir,
    require => Package['httpd'],
  }
  # Don't purge available modules if an enable dir is used
  $purge_mod_dir = $purge_configs and !$mod_enable_dir
  file { $mod_dir:
    ensure  => directory,
    recurse => true,
    purge   => $purge_mod_dir,
    notify  => Class['Apache::Service'],
    require => Package['httpd'],
  }
}

if $mod_enable_dir and ! defined(File[$mod_enable_dir]) {
  $mod_load_dir = $mod_enable_dir
  exec { "mkdir ${mod_enable_dir}":
    creates => $mod_enable_dir,
    require => Package['httpd'],
  }
  file { $mod_enable_dir:
    ensure  => directory,
    recurse => true,
    purge   => $purge_configs,
    notify  => Class['Apache::Service'],
    require => Package['httpd'],
  }
} else {
  $mod_load_dir = $mod_dir
}
```

```
 if ! defined(File[$vhost_dir]) {
   exec { "mkdir ${vhost_dir}":
     creates => $vhost_dir,
     require => Package['httpd'],
   }
   file { $vhost_dir:
     ensure  => directory,
     recurse => true,
ensure  => directory,
     recurse => true,
     purge   => $purge_vhostd,
     notify  => Class['Apache::Service'],
     require => Package['httpd'],
   }
 }

 if $vhost_enable_dir and ! defined(File[$vhost_enable_dir]) {
   $vhost_load_dir = $vhost_enable_dir
   exec { "mkdir ${vhost_load_dir}":
     creates => $vhost_load_dir,
     require => Package['httpd'],
   }
   file { $vhost_enable_dir:
     ensure  => directory,
     recurse => true,
     purge   => $purge_vhostd,
     notify  => Class['Apache::Service'],
     require => Package['httpd'],
   }
 } else {
   $vhost_load_dir = $vhost_dir
 }
```

```
concat { $ports_file:
  owner   => 'root',
  group   => $::apache::params::root_group,
  mode    => $::apache::file_mode,
  notify  => Class['Apache::Service'],
  require => Package['httpd'],
}
concat::fragment { 'Apache ports header':
  ensure  => present,
  target  => $ports_file,
  content => template('apache/ports_header.erb')
}

if $::apache::conf_dir and $::apache::params::conf_file {
  case $::osfamily {
    'debian': {
      $error_log          = 'error.log'
      $scriptalias        = '/usr/lib/cgi-bin'
      $access_log_file    = 'access.log'
    }
    'redhat': {
      $error_log          = 'error_log'
      $scriptalias        = '/var/www/cgi-bin'
      $access_log_file    = 'access_log'
    }
    'freebsd': {
      $error_log          = 'httpd-error.log'
      $scriptalias        = '/usr/local/www/apache24/cgi-bin'
      $access_log_file    = 'httpd-access.log'
    } 'gentoo': {
      $error_log          = 'error.log'
} 'gentoo': {
      $error_log          = 'error.log'
```

```
  $error_documents_path = '/usr/share/apache2/error'
  $scriptalias          = '/var/www/localhost/cgi-bin'
  $access_log_file      = 'access.log'

  if is_array($default_mods) {
    if versioncmp($apache_version, '2.4') >= 0 {
      if defined('apache::mod::ssl') {
        ::portage::makeconf { 'apache2_modules':
          content => concat($default_mods, [ 'authz_core', 'socache_shmcb' ]),
        }
      } else {
        ::portage::makeconf { 'apache2_modules':
          content => concat($default_mods, 'authz_core'),
        }
      }
    } else {
      ::portage::makeconf { 'apache2_modules':
        content => $default_mods,
      }
    }
  }

  file { [
    '/etc/apache2/modules.d/.keep_www-servers_apache-2',
    '/etc/apache2/vhosts.d/.keep_www-servers_apache-2'
  ]:
    ensure  => absent,
    require => Package['httpd'],
  }
}
'Suse': {
  $error_log          = 'error.log'
  $scriptalias        = '/usr/lib/cgi-bin'
```

```
    $access_log_file      = 'access.log'
  }
  default: {
    fail("Unsupported osfamily ${::osfamily}")
  }
}

$apxs_workaround = $::osfamily ? {
  'freebsd' => true,
  default   => false
}

if $rewrite_lock {
  validate_absolute_path($rewrite_lock)
}

# Template uses:
# - $pidfile
# - $user
# - $group
# - $logroot
# - $group
# - $logroot
# - $error_log
# - $sendfile
# - $mod_dir
# - $ports_file
# - $confd_dir
# - $vhost_dir
# - $error_documents
# - $error_documents_path
# - $apxs_workaround
# - $keepalive
```

```
# - $keepalive_timeout
# - $max_keepalive_requests
# - $server_root
# - $server_tokens
# - $server_signature
# - $trace_enable
# - $rewrite_lock
file { "${::apache::conf_dir}/${::apache::params::conf_file}":
  ensure  => file,
  content => template($conf_template),
  notify  => Class['Apache::Service'],
  require => [Package['httpd'], Concat[$ports_file]],
}

# preserve back-wards compatibility to the times when default_mods was
# only a boolean value. Now it can be an array (too)
if is_array($default_mods) {
  class { '::apache::default_mods':
    all  => false,
    mods => $default_mods,
  }
} else {
  class { '::apache::default_mods':
    all => $default_mods,
  }
}
class { '::apache::default_confd_files':
  all => $default_confd_files
}
if $mpm_module and $mpm_module != 'false' { # lint:ignore:quoted_booleans
  class { "::apache::mod::${mpm_module}": }
}
```

```
$default_vhost_ensure = $default_vhost ? {
  true  => 'present',
  false => 'absent'
}
$default_ssl_vhost_ensure = $default_ssl_vhost ? {
  true  => 'present',
  false => 'absent'
}

::apache::vhost { 'node1':
  ensure          => $default_vhost_ensure,
  port            => 80,
  docroot         => '/var/www/html',
  scriptalias     => $scriptalias,
  serveradmin     => $serveradmin,
  access_log_file => $access_log_file,
  priority        => '15',
  ip              => '192.168.1.12',
  logroot_mode    => $logroot_mode,
  manage_docroot  => $default_vhost,
}
$ssl_access_log_file = $::osfamily ? {
  'freebsd' => $access_log_file,
  default   => "ssl_${access_log_file}",
}
::apache::vhost { 'default-ssl':
  ensure          => $default_ssl_vhost_ensure,
  port            => 443,
  ssl             => true,
  docroot         => $docroot,
  scriptalias     => $scriptalias,
  serveradmin     => $serveradmin,
  access_log_file => $ssl_access_log_file,
```

```
    priority        => '15',
    ip              => $ip,
    logroot_mode    => $logroot_mode,
    manage_docroot  => $default_ssl_vhost,
   }
  }

  # This anchor can be used as a reference point for things that need to happen *after*
  # all modules have been put in place.
  anchor { '::apache::modules_set_up': }
}
```

Configuration file for MYSQL Server

```
# Class: mysql::server:  See README.md for documentation.
class mysql::server (
  $config_file            = $mysql::params::config_file,
  $includedir             = $mysql::params::includedir,
  $install_options        = undef,
  $install_secret_file    = $mysql::params::install_secret_file,
  $manage_config_file     = $mysql::params::manage_config_file,
  $override_options       = {},
  $package_ensure         = $mysql::params::server_package_ensure,
  $package_manage         = $mysql::params::server_package_manage,
  $package_name           = $mysql::params::server_package_name,
```

```
  $purge_conf_dir         = $mysql::params::purge_conf_dir,
  $remove_default_accounts = false,
  $restart                = $mysql::params::restart,
  $root_group             = $mysql::params::root_group,
  $mysql_group            = $mysql::params::mysql_group,
  $root_password          = $mysql::params::root_password,
  $service_enabled        = $mysql::params::server_service_enabled,
  $service_manage         = $mysql::params::server_service_manage,
  $service_name           = $mysql::params::server_service_name,
  $service_provider       = $mysql::params::server_service_provider,
  $create_root_user       = $mysql::params::create_root_user,
  $create_root_my_cnf     = $mysql::params::create_root_my_cnf,
  $users                  = {},
  $grants                 = {},
  $databases              = {},

  # Deprecated parameters
  $enabled                = undef,
  $manage_service         = undef,
  $old_root_password      = undef
) inherits mysql::params {


  # Deprecated parameters.
  if $enabled {
    crit('This parameter has been renamed to service_enabled.')
    $real_service_enabled = $enabled
  } else {
    $real_service_enabled = $service_enabled
  }
  if $manage_service {
    crit('This parameter has been renamed to service_manage.')
    $real_service_manage = $manage_service
```

```
} else {
  $real_service_manage = $service_manage
}
if $old_root_password {
  warning('old_root_password is no longer used and will be removed in a future release')
}

# Create a merged together set of options.  Rightmost hashes win over left.
$options = mysql_deepmerge($mysql::params::default_options, $override_options)

Class['mysql::server::root_password'] -> Mysql::Db <| |>

include '::mysql::server::install'
include '::mysql::server::config'
include '::mysql::server::installdb'
include '::mysql::server::service'
include '::mysql::server::root_password'
include '::mysql::server::providers'

if $remove_default_accounts {
  class { '::mysql::server::account_security':
    require => Anchor['mysql::server::end'],
  }
}

anchor { 'mysql::server::start': }
anchor { 'mysql::server::end': }

if $restart {
  Class['mysql::server::config'] ~>
  Class['mysql::server::service']
}
```

```
  Anchor['mysql::server::start'] ->
  Class['mysql::server::install'] ->
  Class['mysql::server::config'] ->
  Class['mysql::server::installdb'] ->
  Class['mysql::server::service'] ->
  Class['mysql::server::root_password'] ->
  Class['mysql::server::providers'] ->
  Anchor['mysql::server::end']
}
```

Configuration file for SSMTP Server

```
class ssmtp (
  $default_mta       = $ssmtp::params::default_mta,
  $root_email        = $ssmtp::params::root_email,
  $mail_hub          = $ssmtp::params::mail_hub,
  $revaliases        = $ssmtp::params::revaliases,
  $from_line_override = $ssmtp::params::from_line_override,
  $hostname          = undef,
  $rewritedomain     = undef,
  $authuser          = undef,
  $authpass          = undef,
  $authmethod        = undef,
  $usetls            = undef,
  $usestarttls       = undef,
  $tlscert           = undef,
  $tlskey            = undef,
  $tls_ca_file       = undef,
  $tls_ca_dir        = undef,
) inherits ssmtp::params {

  # Start workflow
  if $ssmtp::params::linux {
```

```
  # Containment
  contain ssmtp::package
  contain ssmtp::config
  contain ssmtp::service
  Class['ssmtp::package'] ->
  Class['ssmtp::config'] ->
  Class['ssmtp::service']
 }
}
 else {
  warning('The current operating system is not supported!')
 }
}
```

Configuration file for FTP Server

```
class vsftpd (
 $confdir              = $::vsftpd::params::confdir,
 $package_name         = $::vsftpd::params::package_name,
 $service_name         = $::vsftpd::params::service_name,
 $template             = 'vsftpd/vsftpd.conf.erb',
 # vsftpd.conf options
 $anonymous_enable     = 'NO',
 $local_enable         = 'YES',
 $write_enable         = 'YES',
 $local_umask          = '022',
 $anon_upload_enable   = 'NO',
 $anon_mkdir_write_enable = 'NO',
 $dirmessage_enable    = 'YES',
 $xferlog_enable       = 'YES',
 $connect_from_port_20 = 'YES',
 $chown_uploads        = 'NO',
 $chown_username       = undef,
 $xferlog_file         = '/var/log/vsftpd.log',
```

```
  $xferlog_std_format      = 'YES',
  $idle_session_timeout    = '600',
  $data_connection_timeout = '120',
  $nopriv_user             = undef,
  $async_abor_enable       = 'NO',
  $ascii_upload_enable     = 'NO',
  $ascii_download_enable   = 'NO',
  $ftpd_banner             = 'FTP SERVER',
  $chroot_local_user       = 'YES',
  $chroot_list_enable      = 'NO',
  $chroot_list_file        = '/etc/vsftpd/chroot_list',
  $ls_recurse_enable       = 'NO',
  $listen                  = 'YES',
  $listen_port             = undef,
  $pam_service_name        = 'vsftpd',
  $userlist_enable         = 'YES',
  $userlist_deny           = undef,
  $tcp_wrappers            = 'YES',
  $hide_file               = undef,
  $hide_ids                = 'NO',
  $setproctitle_enable     = 'NO',
  $text_userdb_names       = 'NO',
  $max_clients             = undef,
  $max_per_ip              = undef,
  $pasv_min_port           = undef,
  $pasv_max_port           = undef,
  $ftp_username            = undef,
  $banner_file             = undef,
  $allow_writeable_chroot  = undef,
  $directives              = {},
) inherits ::vsftpd::params {

  package { $package_name: ensure => installed }
```

```
    service { $service_name:
      require   => Package[$package_name],
      enable    => true,
      ensure    => running,
      hasstatus => true,
     }

    file { "${confdir}/vsftpd.conf":
      require => Package[$package_name],
      content => template($template),
      notify  => Service[$service_name],
     }
   }
```

Honssh Configuration

```
#
# HonSSH configuration file (honssh.cfg)
#

#-----------------------------------------------#
#               GENERAL SETUP                   #
#-----------------------------------------------#

#----------------------#
#      HONEYPOT        #
#----------------------#
[honeypot]

# IP addresses to listen for incoming SSH connections.
#
# input: IP Address
# required: YES
ssh_addr = 192.168.1.5

# Port to listen for incoming SSH connections.
#
# input: Number
# required: YES
# default: 2222
ssh_port = 22
```

```
# IP addresses to send outgoing SSH connections.
# 0.0.0.0 for all interfaces
#
# input: IP Address
# required: YES
client_addr = 0.0.0.0

# Public and private SSH key files.
#
# input: Text
# required: YES
# default: id_rsa.pub
# default: id_rsa
# default: id_dsa.pub
# default: id_dsa
public_key = id_rsa.pub
private_key = id_rsa
public_key_dsa = id_dsa.pub
private_key_dsa = id_dsa

# SSH banner to send to clients
# If not specified, HonSSH will try and obtain it by connecting to
# honey_addr:honey_port
#
# input: text
# required: No
# default:
ssh_banner =

#----------------------#
#   HONEYPOT STATIC    #
#----------------------#
[honeypot-static]
# Documentation to come, stick with these options and ignore honeypot-* unless you know what you are
doing or fancy a challenge
enabled = false

# Should HonSSH use this plugin to get the honeypot details (before authentication)
pre-auth = false

# Should HonSSH use this plugin to get the honeypot details (after authentication)
post-auth = false

# This name will be used when logging to any of the output mechanisms.
# Please ensure it is meaningful.
#
# input: Text
# required: YES
```

```
sensor_name =

# IP addresses of the honeypot.
#
# input: IP Address
# required: YES
honey_ip = 192.168.1.12

# SSH port of the honeypot.
#
# input: Number
# required: YES
# default: 22
honey_port = 22

#----------------------#
#   HONEYPOT SCRIPT    #
#----------------------#
[honeypot-script]
# Documentation to come
enabled = true

# Should HonSSH use this plugin to get the honeypot details (before authentication)
pre-auth = true

# Should HonSSH use this plugin to get the honeypot details (after authentication)
post-auth = true

# ./script IP LOCALIP PORT LOCALPORT
pre-auth-script =/usr/local/sbin/pot.py

# ./script IP LOCALIP PORT LOCALPORT USERNAME PASSWORD
post-auth-script =/usr/local/sbin/pot.py

#----------------------#
#   HONEYPOT DOCKER    #
#----------------------#
[honeypot-docker]
# Documentation to come
enabled = false

# Should HonSSH use this plugin to get the honeypot details (before authentication)
pre-auth = false

# Should HonSSH use this plugin to get the honeypot details (after authentication)
post-auth = false

# image: image id/name to use for honeypot container
# required: if enabled = true
```

```
image =

# uri: socket to interact with container daemon
# required: if enabled = true
# default: unix://var/run/docker.sock
uri = unix://var/run/docker.sock

# honey_hostname: the hostname for the container
# required: if enabled = true
hostname = test-box

# launch_cmd: command to run when container is first launched
# required: if enabled = true
# default = service ssh start
launch_cmd = service ssh start

# SSH port of the honeypot.
#
# input: Number
# required: YES
# default: 22
honey_port =

#----------------------#
# HONEYPOT RESTRICTIONS #
#----------------------#
[hp-restrict]

# When enabled, HonSSH will restrict connections to password only and decline any public keys.
# HonSSH will not work with public keys - this should always be true.
#
# input: true/false
# required: YES
# default: true
disable_publicKey = true

# When enabled, HonSSH will block any attempts to start an X11 session.
# You can allow X11 but HonSSH will not log the session.
#
# input: true/false
# required: YES
# default: true
disable_x11 = true

# When enabled, HonSSH will block any attempts to start an SFTP session.
# HonSSH will log SFTP traffic and capture downloaded files.
#
# input: true/false
# required: YES
```

```
# default: false
disable_sftp = false

# When enabled, HonSSH will block any attempts to start an EXEC session.
# HonSSH will log all EXEC sessions, including SCP transfers.
#
# input: true/false
# required: YES
# default: false
disable_exec = false

# When enabled, HonSSH will block any attempts to start running port forwarding over SSH.
# You can allow port forwarding but HonSSH will not log the session - Yet! (log to PCAP?)
#
# input: true/false
# required: YES
# default: true
disable_port_forwarding = true

#----------------------#
#   OUTPUT DIRECTORIES  #
#----------------------#
[folders]

# Directory where log files will be saved in.
#
# input: Text
# required: YES
# default: logs
log_path = logs

# Directory where session files will be saved in.
#
# input: Text
# required: YES
# default: sessions
session_path = sessions

#----------------------#
#  ADVANCED NETWORKING  #
#----------------------#
[advNet]

# To enable this HonSSH must be ran as root or an account allowed to run
# iptables and ip link/addr commands.
#
# With this disabled, the honeypot will always see connections coming from
# honey_addr. With this enabled, connections will look as if the connections
# are coming from the attacker.
```

```
# See the Wiki page for more details.
# https://github.com/tnich/honssh/wiki/Advanced-Networking
#
# input: true/false
# required: YES
# default: false
enabled = false

#----------------------#
#    LIVE INTERACTION   #
#----------------------#
[interact]

# Session management interface.
#
# This is a TCP based service that can be used to interact with active
# sessions. Disabled by default.
#
# Use honsshInteraction.py to interact with this interface.
#
# input: true/false
# required: YES
# default: false
enabled = false

# Interface to create the interaction on - 0.0.0.0 for all.
#
# input: IP Address
# required: if interact_enabled = true
# default: 127.0.0.1
interface = 127.0.0.1

# Port to create the interaction on
#
# input: Number
# required: if interact_enabled = true
# default: 5123
port = 5123

#----------------------#
#   PASSWORD SPOOFING   #
#----------------------#
[spoof]

# Enabling this will allow HonSSH to spoof an incorrect password with the real password.
# A list of users and passwords must be defined in the users.cfg file.
#
# Passwords to spoof can either be a fixed list or a random chance.
#
```

```
# See the Wiki page for more details.
# https://github.com/tnich/honssh/wiki/Password-Spoofing
#
# input: true/false
# required: YES
# default: false
enabled = true

# Location of the users.cfg file
#
# input: text
# required: if enabled is true
# default: users.cfg
users_conf = users.cfg

#---------------------------------------------#
#            LOGGING AND OUTPUTS              #
#---------------------------------------------#

#----------------------#
#   FILE DOWNLOADING   #
#----------------------#
[download]

# File Download
#
# HonSSH will attempt to download all scp and sftp files to a local store if this is true
#
# input: true/false
# required: YES
# default: false
passive = true

# HonSSH wil attempt to download all wget files to a local store.
#
# I believe another tool should be used to passively capture all http(s) connections on all ports - maybe
the next project?
# Until then HonSSH will use a 'best effort' approach to capture files when the wget commands is
detected.
# It will not be able to capture commands such as:
# url=www.test.url; wget $url
#
# input: true/false
# required: YES
# default: false
active = true

#----------------------#
#     TEXT LOGGING     #
```

```
#----------------------#
[output-txtlog]

# All activity will be logged to text files
# A log of entry attempts will be kept in log_path/
# A log of session activity will be kept in session_path/
#
# input: true/false
# required: YES
# default: true
enabled = true

#----------------------#
#     MYSQL LOGGING    #
#----------------------#
[output-mysql]

# All activity will be logged to a MYSQL Database
# Database structure for this module is supplied in utils/honssh.sql
#
# input: true/false
# required: yes
# default: false
enabled = false

# IP address of the database
#
# input: IP Address
# required: if enabled = true
# default: localhost
host =

# Port to connect to the database on
#
# input: Number
# required: NO
# default: 3306
port = 3306

# Name of the database
#
# input: Text
# required: if enabled = true
database =

# Username to authenticate with the database
#
# input: Text
# required: if enabled = true
```

```
username =

# Password to authenticate with the database
#
# input: Text
# required: if enabled = true
password =

#----------------------#
#     EMAIL LOGGING    #
#----------------------#
[output-email]

# Enable email output plugin
#
# dependency: txtlog MUST be enabled
# input: true/false
# required: YES
# default: false
enabled = false

# Send an email upon hacker connect
#
# dependency: txtlog MUST be enabled
# input: true/false
# required: YES
# default: false
login = false

# Send an email upon hacker disconnect - Will attach the tty log file
#
# dependency: txtlog MUST be enabled
# input: true/false
# required: YES
# default: false
attack = false

# Your SMTP Host
#
# input: Text
# required: if login or attack = true
host =

# Your SMTP Port
#
# input: Number
# required: if login or attack = true
port =
```

```
# Use SSL/TLS to connect to the SMTP provider?
#
# input: true/false
# required: if login or attack = true
# default: true
use_tls = true

# Does your SMTP provider require a login?
#
# input: true/false
# required: if login or attack = true
# default: true
use_smtpauth = true

# Your SMTP login username
#
# input: Text
# required: if use_smtpauth = true
username =

# Your SMTP login password
#
# input: Text
# required: if use_smtpauth = true
password =

# The address the email is sent from
#
# input: Email Address
# required: if login or attack = true
from =

# The address(es) the email is sent to
#
# input: Email Addresses in a comma seperated list spaces without
# required: if login or attack = true
to =

#----------------------#
#       HP FEEDS       #
#----------------------#
[output-hpfeeds]

# All activity will be logged to a hpfeeds broker for dissemination
# between the honeypot community.
# Authentication attempts will be logged to honssh.auth
# Sessions will be logged to honssh.sessions
#
# input: true/false
```

```
# required: yes
# default: false
enabled = false

# The server address of the hpfeeds broker
#
# input: Text
# required: if enabled = true
server =

# The server port of the hpfeeds broker
#
# input: Number
# required: if enabled = true
port =

# Your hpfeed authe key identifier
#
# input: Text
# required: if enabled = true
identifier =

# Your hpfeed authe key secret
#
# input: Text
# required: if enabled = true
secret =

#----------------------#
#   APPLICATION HOOKS   #
#----------------------#
[output-app_hooks]

# Enable app_hooks output plugin
#
# input: true/false
# required: YES
# default: false
enabled = false

# If you want any other application hooks or arguments passing, raise an issue
# on the HonSSH code page.

# Calls the script when a connection is made with the following arguments
# ./script CONNECTION_MADE DATETIME IP PORT HONEYIP HONEYPORT SESSION_ID
#
# input: path of script to run
# required: NO
connection_made =
```

```
# Calls the script when a connection is lost with the following arguments
# ./script CONNECTION_LOST DATETIME IP PORT HONEYIP HONEYPORT SESSION_ID
#
# input: path of script to run
# required: NO
connection_lost =

# Calls the script when a login is successful with the following arguments
# ./script LOGIN_SUCCESSFUL DATETIME IP USERNAME PASSWORD
#
# input: path of script to run
# required: NO
login_successful =

# Calls the script when a login has failed with the following arguments
# ./script LOGIN_FAILED DATETIME IP USERNAME PASSWORD
#
# input: path of script to run
# required: NO
login_failed =

# Calls the script when a channel is opened with the following arguments
# ./script CHANNEL_OPENED DATETIME NAME CHANNEL_ID
#
# input: path of script to run
# required: NO
channel_opened =

# Calls the script when a channel is closed with the following arguments
# ./script CHANNEL_CLOSED DATETIME NAME CHANNEL_ID
#
# input: path of script to run
# required: NO
channel_closed =

# Calls the script when a command is entered with the following arguments
# ./script COMMAND_ENTERED DATETIME CHANNEL_ID COMMAND
#
# input: path of script to run
# required: NO
command_entered =

# Calls the script when a file download is started with the following arguments
# ./script DOWNLOAD_STARTED DATETIME CHANNEL_ID LINK FILE_PATH
#
# input: path of script to run
# required: NO
download_started =
```

 # Calls the script when a file download is finished with the following arguments
 # ./script DOWNLOAD_FININSHED DATETIME CHANNEL_ID LINK FILE_PATH
#
 # input: path of script to run
 # required: NO
download_finished =

#----------------------#
 # PACKET LOGGING #
#----------------------#
[packet_logging]

 # Set to true to enable plugins to use the packet_logged function
#
 # input: true/false
 # required: YES
 # default: false
enabled = true

[output-packets]

 # Log all SSH Packets to text file (.log-adv)
#
 # dependency: packet_logging MUST be enabled
 # input: true/false
 # required: YES
 # default: false
enabled = true

Summary of the data (attacks)

016-04-12 14:35:18-0500 [-] Starting factory <honssh.client.HonsshClientFactory instance at 0x7fe42c703830>

2016-04-12 14:35:18-0500 [Uninitialized] [CLIENT] - New client connection

2016-04-12 14:35:18-0500 [HonsshClientTransport,client] kex alg, key alg: diffie-hellman-group-exchange-sha1 ssh-rsa

2016-04-12 14:35:18-0500 [HonsshClientTransport,client] outgoing: aes256-ctr hmac-sha1 none

2016-04-12 14:35:18-0500 [HonsshClientTransport,client] incoming: aes256-ctr hmac-sha1 none

2016-04-12 14:35:18-0500 [HonsshClientTransport,client] REVERSE

2016-04-12 14:35:18-0500 [HonsshClientTransport,client] NEW KEYS

2016-04-12 14:35:18-0500 [HonsshClientTransport,client] [CLIENT] - Client Connection Secured

2016-04-12 14:35:18-0500 [HonsshServerTransport,3442,121.12.127.94] kex alg, key alg: diffie-hellman-group1-sha1 ssh-rsa

2016-04-12 14:35:18-0500 [HonsshServerTransport,3442,121.12.127.94] outgoing: aes128-ctr hmac-sha1 none

2016-04-12 14:35:18-0500 [HonsshServerTransport,3442,121.12.127.94] incoming: aes128-ctr hmac-sha1 none

2016-04-12 14:35:18-0500 [-] [PLUGIN][EXAMPLE] - SET_SERVER

2016-04-12 14:35:18-0500 [-] [PLUGIN][EXAMPLE] - SET SERVER

2016-04-12 14:35:18-0500 [-] [PLUGIN][OUTPUT-TXTLOG] - CONNECTION_MADE

2016-04-12 14:35:18-0500 [-] [PLUGIN][EXAMPLE] - CONNECTION_MADE

2016-04-12 14:35:18-0500 [-] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [], 'country': 'China', 'start_time': '20160412_143518_928962', 'log_location': 'sessions/n$

2016-04-12 14:35:18-0500 [-] [PLUGIN][EXAMPLE] - SET_CLIENT

2016-04-12 14:35:18-0500 [-] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [], 'country': 'China', 'start_time': '20160412_143518_928962', 'log_location': 'sessions/n$

2016-04-12 14:35:18-0500 [-] [PRE_AUTH] - CLIENT CONNECTED, REPLAYING BUFFERED PACKETS

2016-04-12 14:35:19-0500 [HonsshServerTransport,3442,121.12.127.94] NEW KEYS

2016-04-12 14:35:19-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [], 'country': 'China', 'start_time': '20160412_1435$

2016-04-12 14:35:19-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [], 'country': 'China', 'start_time': '20160412_143518_928962', $

2016-04-12 14:35:20-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [], 'country': 'China', 'start_time': '20160412_1435$

2016-04-12 14:35:20-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][HONEYPOT-SCRIPT] - GET_POST_AUTH_DETAILS

2016-04-12 14:35:20-0500 [-] [POST_AUTH] - SUCCESS = FALSE, NOT POST-AUTHING

2016-04-12 14:35:20-0500 [-] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [], 'country': 'China', 'start_time': '20160412_143518_928962', 'log_location': 'sessions/n$

2016-04-12 14:35:22-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [], 'country': 'China', 'start_time': '20160412_143518_928962', $

2016-04-12 14:35:22-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [], 'country': 'China', 'start_time': '20160412_143518_928962', $

2016-04-12 14:35:22-0500 [HonsshClientTransport,client] [SSH] - Detected Public Key Auth - Disabling!

2016-04-12 14:35:22-0500 [HonsshClientTransport,client] [PLUGIN][OUTPUT-TXTLOG] - LOGIN_FAILED

2016-04-12 14:35:22-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - LOGIN_FAILED

2016-04-12 14:35:22-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'country': 'China', 'start_time': '20160412_143518_928962', 'log_location$

2016-04-12 14:35:22-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [{'username': 'root', 'date_time': '20160412_143522_$

2016-04-12 14:35:22-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][HONEYPOT-SCRIPT] - GET_POST_AUTH_DETAILS

2016-04-12 14:35:22-0500 [-] [POST_AUTH] - SUCCESS = FALSE, NOT POST-AUTHING

2016-04-12 14:35:22-0500 [-] [POST_AUTH] - Spoofing Password

2016-04-12 14:35:22-0500 [-] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [{'username': 'root', 'date_time': '20160412_143522_496430', 'spoofed': False, 'password': $

2016-04-12 14:35:24-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [{'username': 'root', 'date_time': '20160412_143522_496430', 'sp$

2016-04-12 14:35:24-0500 [HonsshClientTransport,client] [SSH] - Detected Public Key Auth - Disabling!

2016-04-12 14:35:24-0500 [HonsshClientTransport,client] [PLUGIN][OUTPUT-TXTLOG] - LOGIN_FAILED

2016-04-12 14:35:24-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - LOGIN_FAILED

2016-04-12 14:35:24-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'country': 'China', 'start_time': '20160412_143518_928962', 'log_location$

2016-04-12 14:35:25-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [{'username': 'root', 'date_time': '20160412_143522_$

2016-04-12 14:35:25-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][HONEYPOT-SCRIPT] - GET_POST_AUTH_DETAILS

2016-04-12 14:35:25-0500 [-] [POST_AUTH] - SUCCESS = FALSE, NOT POST-AUTHING

2016-04-12 14:35:25-0500 [-] [POST_AUTH] - Spoofing Password

2016-04-12 14:35:25-0500 [-] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [{'username': 'root', 'date_time': '20160412_143522_496430', 'spoofed': False, 'password': $

2016-04-12 14:35:27-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [{'username': 'root', 'date_time': '20160412_143522_496430', 'sp$

2016-04-12 14:35:27-0500 [HonsshClientTransport,client] [SSH] - Detected Public Key Auth - Disabling!

2016-04-12 14:35:27-0500 [HonsshClientTransport,client] [PLUGIN][OUTPUT-TXTLOG] - LOGIN_FAILED

2016-04-12 14:35:27-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - LOGIN_FAILED

2016-04-12 14:35:27-0500 [HonsshClientTransport,client] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'country': 'China', 'start_time': '20160412_143518_928962', 'log_location$

2016-04-12 14:35:27-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][EXAMPLE] - {'honey_port': '22', 'sensor_name': 'node2', 'session': {'auths': [{'username': 'root', 'date_time': '20160412_143522_$

2016-04-12 14:35:27-0500 [HonsshServerTransport,3442,121.12.127.94] [PLUGIN][HONEYPOT-SCRIPT] - GET_POST_AUTH_DETAILS

2016-04-12 14:35:27-0500 [-] [POST_AUTH] - SUCCESS = FALSE, NOT POST-AUTHING

2016-04-12 14:35:27-0500 [-] [POST_AUTH] - Spoofing Password