ARP Spoofing & Abnormality Detection

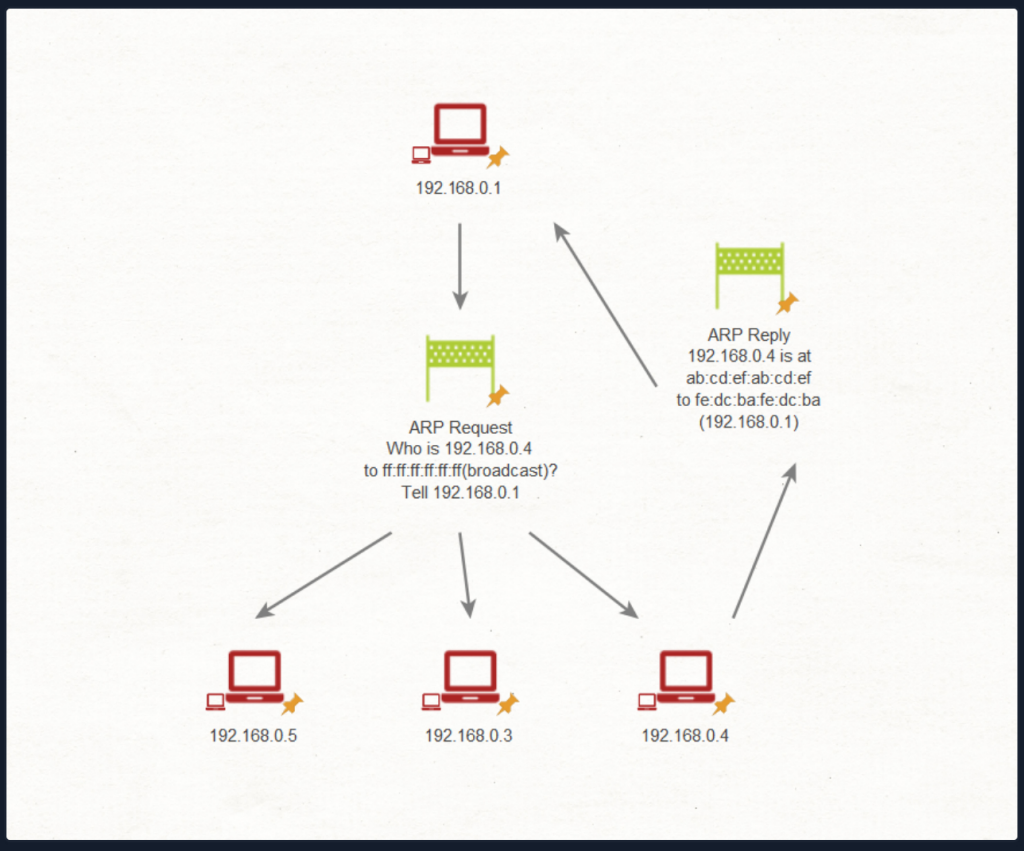
Related PCAP File(s):

- `ARP_Spoof.pcapng`

The `Address Resolution Protocol (ARP)` has been a longstanding utility exploited by attackers to launch man-in-the-middle and denial-of-service attacks, among others. Given this prevalence, ARP forms a focal point when we undertake traffic analysis, often being the first protocol we scrutinize. Many ARP-based attacks are broadcasted, not directed specifically at hosts, making them more readily detectable through our packet sniffing techniques.

How Address Resolution Protocol Works

Before identifying ARP anomalies, we need to first comprehend how this protocol functions in its standard, or 'vanilla', operation.



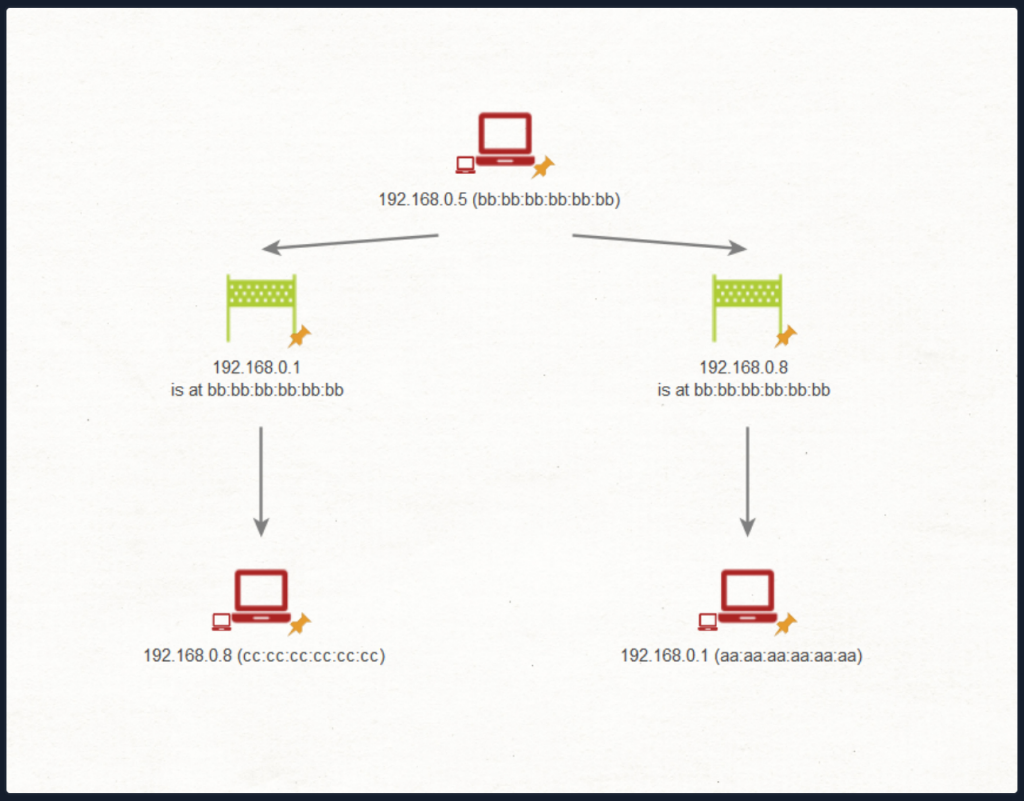
In our network, hosts must know the physical address (MAC address) to which they must send their data. This need gave birth to ARP. Let's elucidate this with a step-by-step process.

Step	Description
1	Imagine our first computer, or Host A, needs to send data to our second computer, Host B. To achieve successful transmission, Host A must ascertain the physical address of Host B.
2	Host A begins by consulting its list of known addresses, the ARP cache, to check if it already possesses this physical address.
3	In the event the address corresponding to the desired IP isn't in the ARP cache, Host A broadcasts an ARP request to all machines in the subnet, inquiring, "Who holds the IP x.x.x.x?"

4	Host B responds to this message with an ARP reply, "Hello, Host A, my IP is x.x.x.x and is mapped to MAC address aa:aa:aa:aa:aa:aa."
5	On receiving this response, Host A updates its ARP cache with the new IP-to-MAC mapping.
6	Occasionally, a host might install a new interface, or the IP address previously allocated to the host might expire, necessitating an update and remapping of the ARP cache. Such instances could introduce complications when we analyze our network traffic.

ARP Poisoning & Spoofing

In an ideal scenario, robust controls would be in place to thwart these attacks, but in reality, this isn't always feasible. To comprehend our Indicators of Compromise (IOCs) more effectively, let's delve into the behavior of ARP Poisoning and Spoofing attacks.



Detecting these attacks can be challenging, as they mimic the communication structure of standard ARP traffic. Yet, certain ARP requests and replies can reveal their nefarious nature. Let's illustrate how these attacks function, enabling us to better identify them during our traffic analysis.

Step	Description
1	Consider a network with three machines: the victim's computer, the router, and the attacker's machine.
2	The attacker initiates their ARP cache poisoning scheme by dispatching counterfeit ARP messages to both the victim's computer and the router.
3	The message to the victim's computer asserts that the gateway's (router's) IP address corresponds to the physical address of the attacker's machine.
4	Conversely, the message to the router claims that the IP address of the victim's machine maps to the physical address of the attacker's machine.
5	On successfully executing these requests, the attacker may manage to corrupt the ARP cache on both the victim's machine and the router, causing all data to be misdirected to the attacker's machine.
6	If the attacker configures traffic forwarding, they can escalate the situation from a denial-of-service to a man-in-the-middle attack.
7	By examining other layers of our network model, we might discover additional attacks. The attacker could conduct DNS spoofing to redirect web requests to a bogus site or perform SSL stripping to attempt the interception of sensitive data in transit.

Detecting these attacks is one aspect, but averting them is a whole different challenge. We could potentially fend off these attacks with controls such as:

OFFLINE

Start Instance

∞ / 1 spawns left

1. `Static ARP Entries`: By disallowing easy rewrites and poisoning of the ARP cache, we can stymie these attacks. This, however, necessitates increased maintenance and oversight in our network environment.
2. `Switch and Router Port Security`: Implementing network profile controls and other measures can ensure that only authorized devices can connect to specific ports on our network devices, effectively blocking machines attempting ARP spoofing/poisoning.

Installing & Starting TCPDump

To effectively capture this traffic, especially in the absence of configured network monitoring software, we can employ tools like `tcpdump` and `Wireshark`, or simply `Wireshark` for Windows hosts.

We can typically find `tcpdump` located in `/usr/sbin/tcpdump`. However, if the tool isn't installed, it can be installed using the appropriate command, which will be provided based on the specific system requirements.

TCPDump

ARP Spoofing & Abnormality Detection

```
MisaelMacias@htb[/htb]$ sudo apt install tcpdump -y
```

To initiate the traffic capture, we can employ the command-line tool `tcpdump`, specifying our network interface with the `-i` switch, and dictating the name of the output capture file using the `-w` switch.

ARP Spoofing & Abnormality Detection

```
MisaelMacias@htb[/htb]$ sudo tcpdump -i eth0 -w filename.pcapng
```

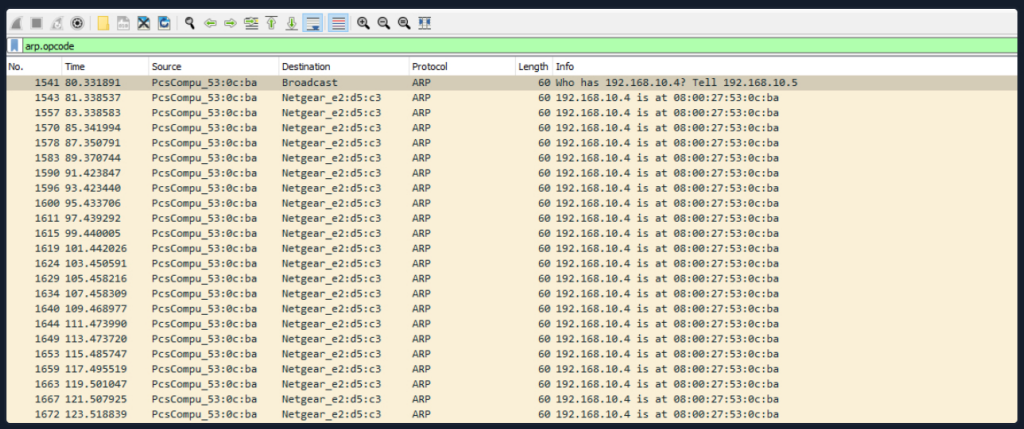
Finding ARP Spoofing

For detecting ARP Spoofing attacks, we'll need to open the related traffic capture file (`ARP_Spoof.pcapng`) from this module's resources using Wireshark.

ARP Spoofing & Abnormality Detection

```
MisaelMacias@htb[/htb]$ wireshark ARP_Spoof.pcapng
```

Once we've navigated to Wireshark, we can streamline our view to focus solely on ARP requests and replies by employing the filter `arp.opcode`.



A key red flag we need to monitor is any anomaly in traffic emanating from a specific host. For instance, one host incessantly broadcasting ARP requests and replies to another host could be a telltale sign of ARP spoofing.

In such a scenario, we might identify that the MAC address `08:00:27:53:0C:BA is behaving suspiciously`.

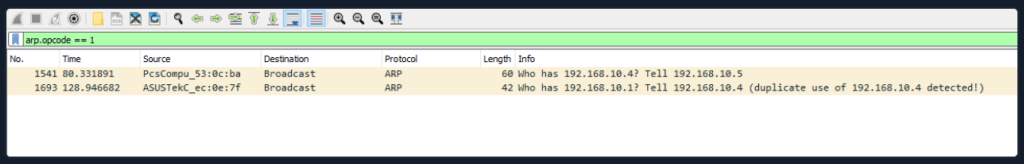
To ascertain this, we can fine-tune our analysis to inspect solely the interactions—both requests and replies—among

the attacker's machine, the victim's machine, and the router. The opcode functionality in `Wireshark` can simplify this process.

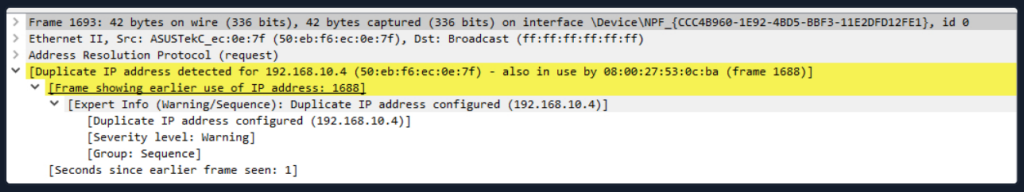
1. `Opcode == 1`: This represents all types of ARP Requests
2. `Opcode == 2`: This signifies all types of ARP Replies

As a preliminary step, we could scrutinize the requests dispatched using the following filter.

- `arp.opcode == 1`



Almost instantly, we should notice a red flag - an address duplication, accompanied by a warning message. If we delve into the details of the error message within Wireshark, we should be able to extract additional information.



Upon immediate inspection, we might discern that one IP address is mapped to two different MAC addresses. We can validate this on a Linux system by executing the appropriate commands.

ARP

ARP Spoofing & Abnormality Detection

```
MisaelMacias@htb[/htb]$ arp -a | grep 50:eb:f6:ec:0e:7f

? (192.168.10.4) at 50:eb:f6:ec:0e:7f [ether] on eth0
```

ARP Spoofing & Abnormality Detection

```
MisaelMacias@htb[/htb]$ arp -a | grep 08:00:27:53:0c:ba

? (192.168.10.4) at 08:00:27:53:0c:ba [ether] on eth0
```

In this situation, we might identify that our ARP cache, in fact, contains both MAC addresses allocated to the same IP address - an anomaly that warrants our immediate attention.

To sift through more duplicate records, we can utilize the subsequent Wireshark filter.

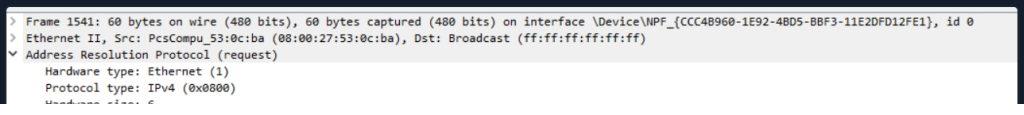
- `arp.duplicate-address-detected && arp.opcode == 2`

Identifying The Original IP Addresses

A crucial question we need to pose is, what were the initial IP addresses of these devices? Understanding this aids us in determining which device altered its IP address through MAC spoofing. After all, if this attack was exclusively performed via ARP, the victim machine's IP address should remain consistent. Conversely, the attacker's machine might possess a different historical IP address.

We can unearth this information within an ARP request and expedite the discovery process using this Wireshark filter.

- `(arp.opcode) && ((eth.src == 08:00:27:53:0c:ba) || (eth.dst == 08:00:27:53:0c:ba))`



```
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: PcsCompu_53:0c:ba (08:00:27:53:0c:ba)
Sender IP address: 192.168.10.5
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.10.4
```

In this case, we might instantly note that the MAC address `08:00:27:53:0c:ba` was initially linked to the IP address `192.168.10.5`, but this was recently switched to `192.168.10.4`. This transition is indicative of a deliberate attempt at ARP spoofing or cache poisoning.

Additionally, examining the traffic from these MAC addresses with the following Wireshark filter can prove insightful:

- `eth.addr == 50:eb:f6:ec:0e:7f or eth.addr == 08:00:27:53:0c:ba`

36 64.016999	204.79.197.254	192.168.10.4	TCP	60 443 → 50985 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
42 66.278704	52.51.147.233	192.168.10.4	TCP	66 443 → 51046 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM WS=256
37 64.307719	52.212.57.111	192.168.10.4	TLSv1.2	85 Encrypted Alert
38 64.441145	52.212.57.111	192.168.10.4	TLSv1.2	85 Encrypted Alert
26 48.614791	ASUSTekC_ec:0e:7f	Broadcast	ARP	42 Who has 192.168.10.1? Tell 192.168.10.4 (duplicate use of 192.168.10.4 detected!)
1 0.000000	PcsCompu_53:0c:ba	Broadcast	ARP	60 Who has 192.168.10.4? Tell 192.168.10.5
39 65.043801	52.212.57.111	192.168.10.4	TCP	85 [TCP Retransmission] 443 → 51811 [FIN, PSH, ACK] Seq=1 Ack=1 Win=784 Len=31
40 65.177343	52.212.57.111	192.168.10.4	TCP	85 [TCP Retransmission] 443 → 64208 [FIN, PSH, ACK] Seq=1 Ack=1 Win=771 Len=31

Right off the bat, we might notice some inconsistencies with TCP connections. If TCP connections are consistently dropping, it's an indication that the attacker is not forwarding traffic between the victim and the router.

If the attacker is, in fact, forwarding the traffic and is operating as a man-in-the-middle, we might observe identical or nearly symmetrical transmissions from the victim to the attacker and from the attacker to the router.

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 140ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

+ 1 Inspect the ARP_Poison.pcapng file, part of this module's resources, and submit the total count of ARP requests (opcode 1) that originated from the address 08:00:27:53:0c:ba as your answer.

507

Submit

Previous Next

Mark Complete & Next

Powered by HACKTHEBOX