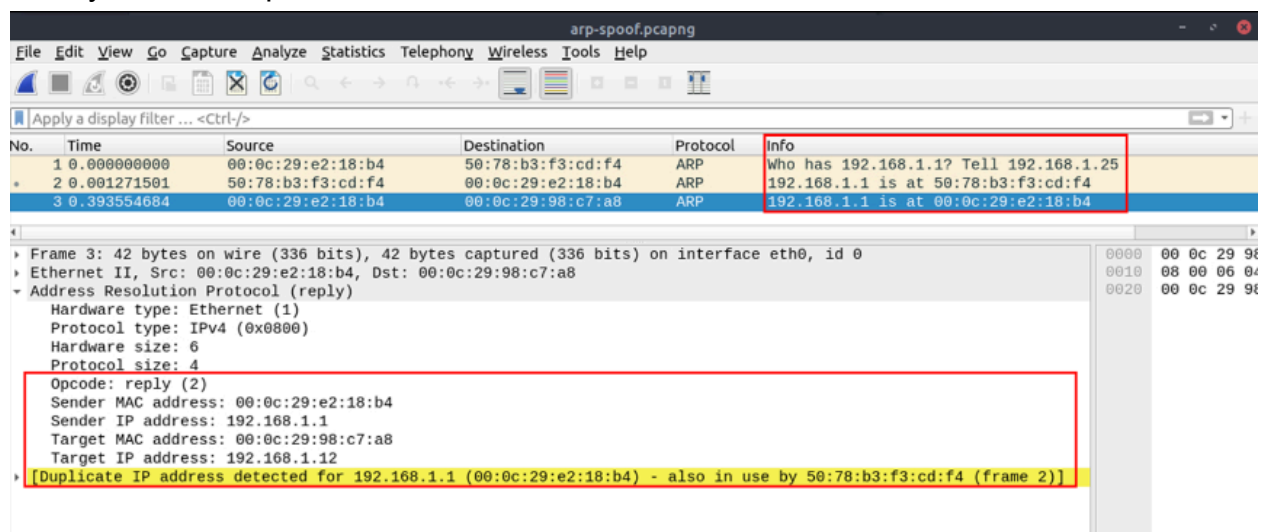


Detecting Man In The Middle attacks

Today I am going to learn how to detect ARP poisoning through Wireshark's packet analysis features. I will be using TryHackMe's learning room to walk through questions and answers. My goal is to be able to teach someone how to detect this attack by reading this writeup as well as enhance my own learning.

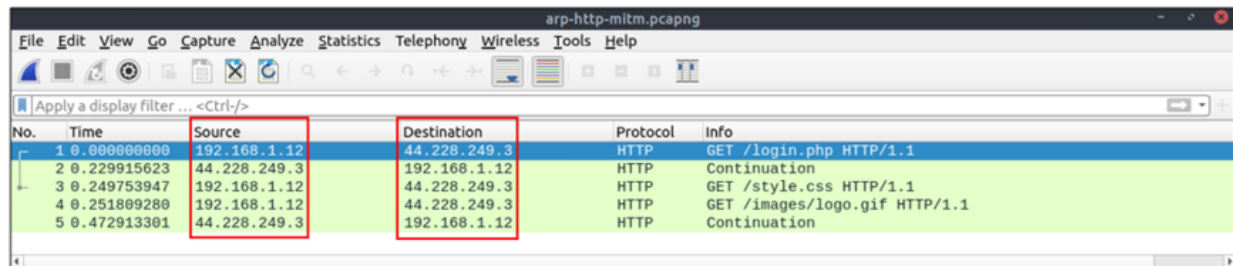
ARP poisoning is a type of network manipulation by sending malicious ARP packets to the default gateway. The goal is to alter the IP to MAC table and sniff traffic. It can prove to be a powerful tool in an attacker's arsenal but can also be easily detected given the right knowledge.

Suspicion should occur when two MAC addresses claim the same IP address. Luckily Wireshark provides a notice for when this occurs, here's what it looks like.



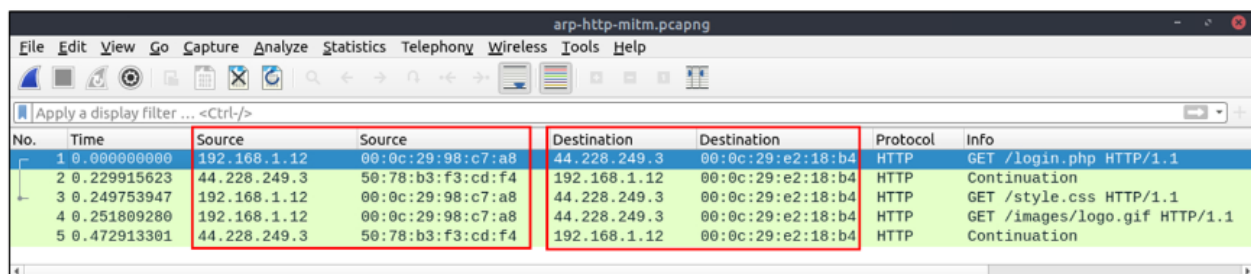
After seeing this we can still not be totally clear an attack is occurring, misconfigurations do occur. But just in case, it is best practice to look further. The first consideration should be finding out which one of the MAC addresses could be fraudulent. There is one ending in "b4" and another in "f4".

To continue the investigation, I'm going to change over to view the HTTP traffic file.



No.	Time	Source	Destination	Protocol	Info
1	0.000000000	192.168.1.12	44.228.249.3	HTTP	GET /login.php HTTP/1.1
2	0.229915623	44.228.249.3	192.168.1.12	HTTP	Continuation
3	0.249753947	192.168.1.12	44.228.249.3	HTTP	GET /style.css HTTP/1.1
4	0.251809280	192.168.1.12	44.228.249.3	HTTP	GET /images/logo.gif HTTP/1.1
5	0.472913301	44.228.249.3	192.168.1.12	HTTP	Continuation

At a first glance, it looks like normal HTTP traffic. By right clicking on any of the columns, we can go to preferences and add a column to view the MAC addresses associated with the traffic.



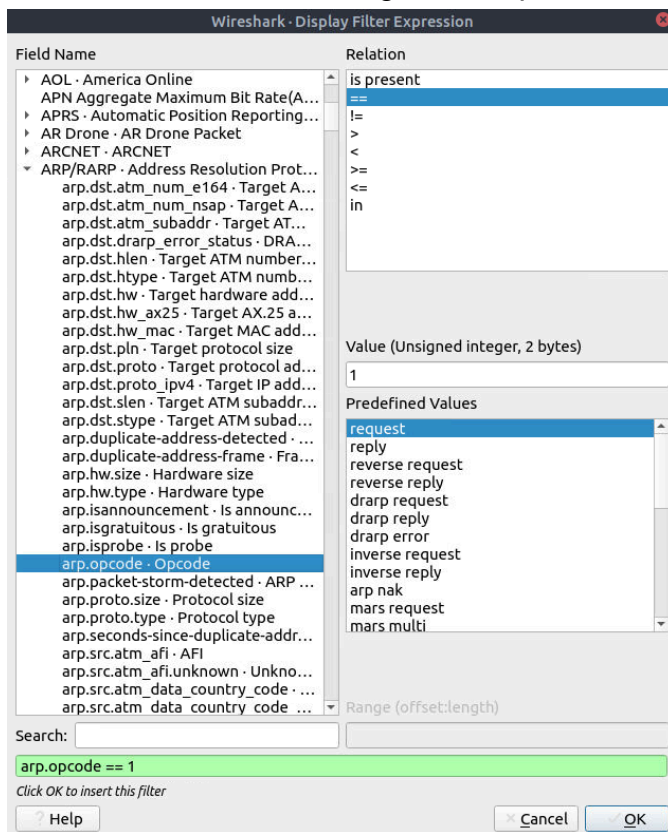
No.	Time	Source	Source	Destination	Destination	Protocol	Info
1	0.000000000	192.168.1.12	00:0c:29:98:c7:a8	44.228.249.3	00:0c:29:e2:18:b4	HTTP	GET /login.php HTTP/1.1
2	0.229915623	44.228.249.3	50:78:b3:f3:cd:f4	192.168.1.12	00:0c:29:e2:18:b4	HTTP	Continuation
3	0.249753947	192.168.1.12	00:0c:29:98:c7:a8	44.228.249.3	00:0c:29:e2:18:b4	HTTP	GET /style.css HTTP/1.1
4	0.251809280	192.168.1.12	00:0c:29:98:c7:a8	44.228.249.3	00:0c:29:e2:18:b4	HTTP	GET /images/logo.gif HTTP/1.1
5	0.472913301	44.228.249.3	50:78:b3:f3:cd:f4	192.168.1.12	00:0c:29:e2:18:b4	HTTP	Continuation

A clear anomaly! All the traffic associated with multiple different IP addresses is all being sent the same MAC address ending in “b4”, the one we had suspicion for earlier. This is an example of ARP spoofing or a “man in the middle” attack. Someone is spoofing IP addresses to intercept traffic.

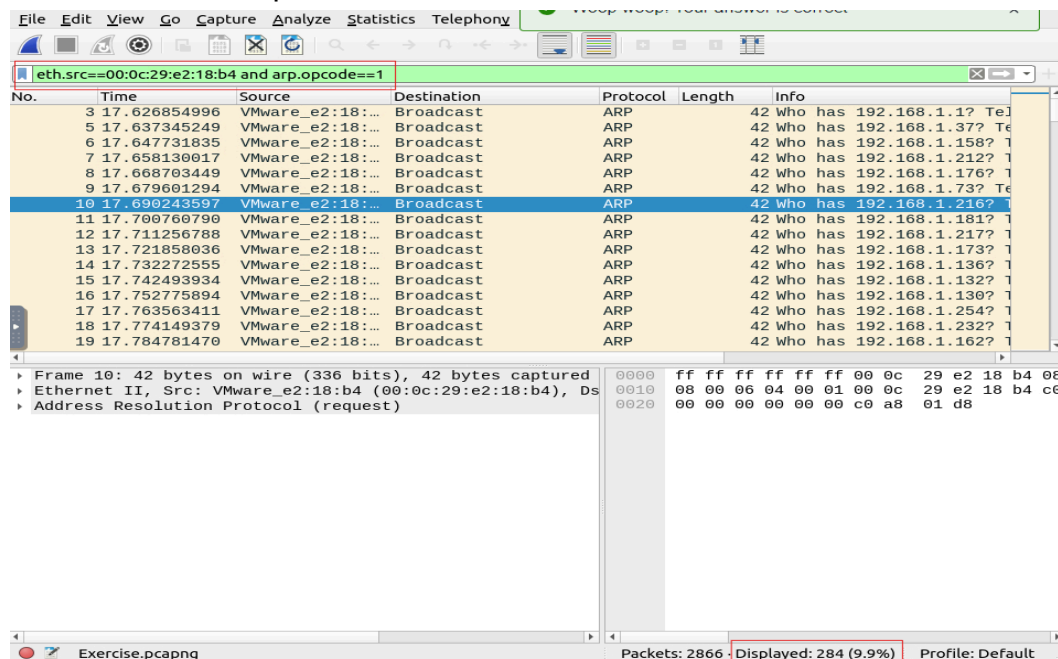
From a defensive standpoint, it would be up to another platform to defend against this attack, Wireshark is only used for analysis. So i’m going to continue onto more examples to detect anomalies like this.

The first question asks how many ARP packets the attacker sent. Now that we know the attackers MAC address, all that’s needed to be done is alter the display filters so we can find what’s needed.

Wireshark has a function that provides expressions needed to filter traffic, which can be found in the analyze tab at the top of the window. After scrolling through the tab, we can find the ARP section and get the expressions to filter for ARP requests.



Using that and the expression to filter ethernet traffic, “eth.src==”, we can find the answer to our questions.



The last question asks “How many usernames and passwords were sniffed by the attacker?” To find this, I'm going to start by filtering all the HTTP traffic sent to the attacker. I manually looked through the traffic messages to find an instance of a username and password being communicated.

eth.dst==00:0c:29:e2:18:b4 and http			
No.	Time	Source	Destination
1143	107.768836318	44.228.249.3	192.168.1.12
1167	107.856353223	192.168.1.12	44.228.249.3
1217	108.088537576	44.228.249.3	192.168.1.12
1226	112.878779336	192.168.1.12	44.228.249.3
1232	113.105809063	44.228.249.3	192.168.1.12
1272	128.722212965	192.168.1.12	44.228.249.3
1275	128.953703403	44.228.249.3	192.168.1.12
1280	130.293866830	192.168.1.12	44.228.249.3
1283	130.519826760	44.228.249.3	192.168.1.12
1288	134.047286876	192.168.1.12	44.228.249.3
1291	134.271756279	44.228.249.3	192.168.1.12
1298	136.609621027	192.168.1.12	44.228.249.3
1301	136.829973665	44.228.249.3	192.168.1.12
1306	137.855933795	192.168.1.12	44.228.249.3
1309	138.076887087	44.228.249.3	192.168.1.12
1381	187.285689649	192.168.1.12	44.228.249.3

▶	Frame 1381: 849 bytes on wire (6792 bits), 849 bytes captured (6792 bits) on interface 0
▶	Ethernet II, Src: VMWare_98:c7:a8 (00:0c:29:98:c7:a8), Dst: 08:00:27:1d:33:00 (08:00:27:1d:33:00)
▶	Internet Protocol Version 4, Src: 192.168.1.12, Destination: 44.228.249.3
▶	Transmission Control Protocol, Src Port: 49915, Dst Port: 80
▶	Hypertext Transfer Protocol
▼	HTML Form URL Encoded: application/x-www-form-urlencoded
▶	Form item: "uname" = "test_THM_test"
▶	Form item: "upass" = "insecurepw"
▶	Form item: "upass2" = "insecurepw"
▶	Form item: "urname" = "MITM_TEST"
▶	Form item: "ucc" = "1234567890"
▶	Form item: "uemail" = ""
▶	Form item: "uphone" = "09876543211"
▶	Form item: "uaddress" = "THM_THM_THM"
▶	Form item: "signup" = "signup"

After that, I found what I was looking for, I added the filter "*urlencoded-form contains 'uname'*" along to what I already had and got my answer.

The screenshot shows the Wireshark interface with a filter applied: `eth.dst==00:0c:29:e2:18:b4 and http and urlencoded-form contains "uname"`. The packet list shows several HTTP POST requests to `44.228.249.3`. The selected packet (1381) is expanded to show the following details:

- Frame 1381: 849 bytes on wire (6792 bits), 849 bytes captured
- Ethernet II, Src: VMware_98:c7:a8 (00:0c:29:98:c7:a8), Dst: 44.228.249.3
- Internet Protocol Version 4, Src: 192.168.1.12, Dst: 44.228.249.3
- Transmission Control Protocol, Src Port: 49915, Dst Port: 80
- Hypertext Transfer Protocol
- HTML Form URL Encoded: `application/x-www-form-urlencoded`
 - Form item: `"uname" = "test_THM_test"`
 - Form item: `"upass" = "insecurepw"`
 - Form item: `"upass2" = "insecurepw"`
 - Form item: `"urname" = "MITM_TEST"`
 - Form item: `"ucc" = "1234567890"`
 - Form item: `"uemail" = ""`
 - Form item: `"uphone" = "09876543211"`
 - Form item: `"uaddress" = "THM_THM_THM"`
 - Form item: `"signup" = "signup"`

The packet bytes pane shows the raw data for the selected packet, starting with `0250 3a 20 68 74 74 70 3a 2f` and ending with `0350 70`.

At the bottom, the status bar indicates: `Text item (text), 21 byte(s)` and `Packets: 2866 - Displayed: 7 (0.2%)`.