**Cybersecurity Final Project - ARP spoofing detector**

Introduction

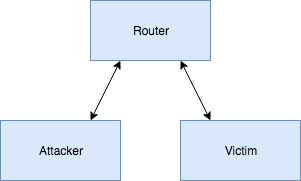
ARP Spoofing is a malicious technique in which an attacker sends a spoofed ARP request on a local area network. The goal of the attacker is to embed their MAC address with the IP address of another host located on the LAN. A correctly spoofed ARP request allows the attacker to intercept the traffic intended for the correct target host IP from the target. ARP Spoofing can be used to perform attacks such as Denial of Service (DOS), Man-in-the-Middle (MITM), or Session Hijacking. ARP is a communications protocol in which the internet layer IP addresses are resolved into link layer MAC addresses. The means of attack in ARP spoofing comes from the lack of host authentication of the packet that was received by the victim.

The goal of the project is to detect ARP spoofing in a local area network. The program created will be used to notify the victim when a possible address hijacking has occurred.

Method

Before any ARP spoofing detection can take place, it is first required to simulate an attack to determine what is normal behavior and what is not normal. A few different configurations were tested, one which involved all machines being virtual and belonging to subnet 192.168.10.0/24. Another configuration involved a WiFi setup with the attacker being in a VM and the victim being a real computer. In any scenario, three (virtual or real) devices are required to be on the same network, since ARP requests only work on layer 2 of the OSI model.

Normal State of a Local Area Network



ARP spoofing attacks are primarily done on the router since most traffic on any LAN passes through there.

The structure of the packet the attacker needs to forge is relatively simple. The packet shown below tells the target "I have ip 192.168.10.128".

ARP Packet Structure

00 50 56 f6 c7 3a 00 0c 29 be af 9d 08 06 00 01

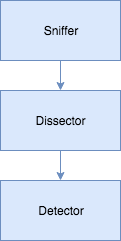
08 00 06 04 00 02 00 0c 29 be af 9d c0 a8 0a 80

00 50 56 f6 c7 3a c0 a8 0a 02 .

kKey: target, attacker, spoofed ip, target ip, protocol information

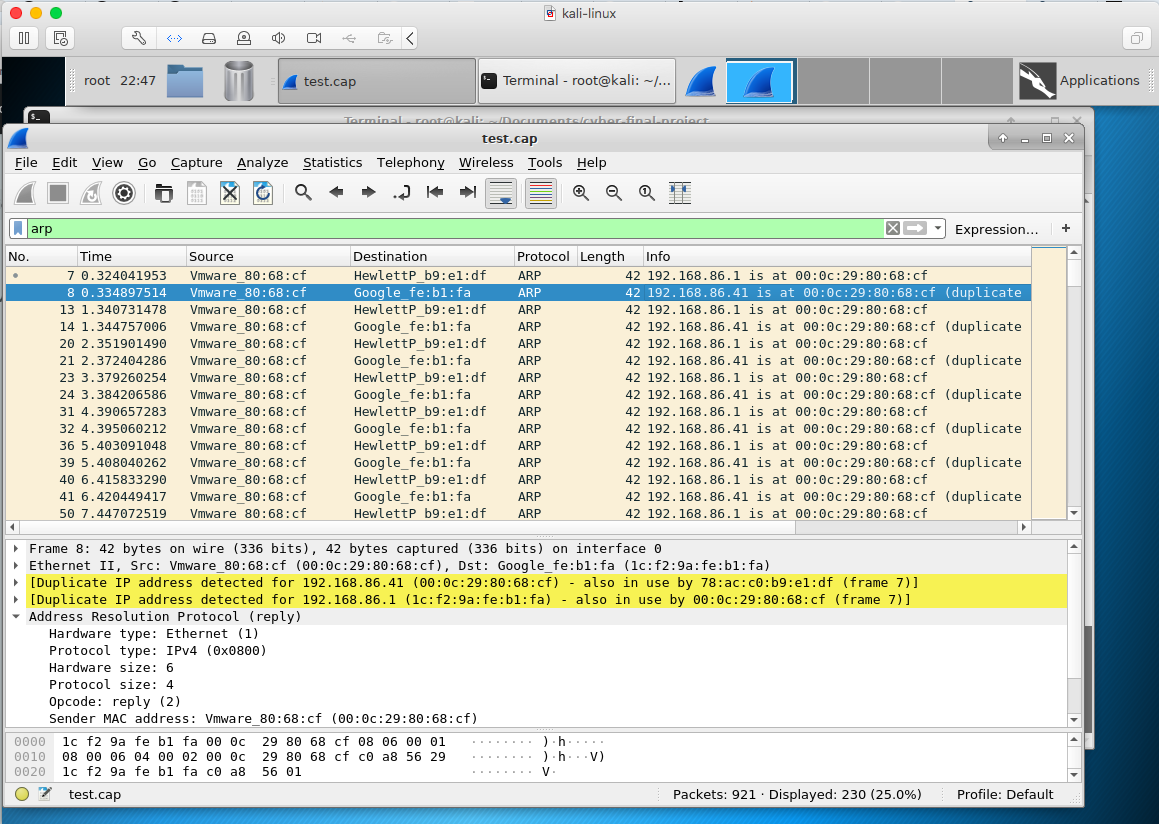
In a typical case ARP packets are only sent after a request is made, however the ARP protocol also allows for gratuidous updates where no request is sent out. If an ARP table entry is not renewed, it is also dropped from the cache. A common method of attack involves spamming ARP replies so that the cache is always poisoned. If the attacker waits until a an ARP request occurs, there may be contention with the actual valid device.

From the perspective of the detector, an anomoly-based detection method is fairly straightforward. If a suspicious number of ARP replys are given when no actual request has been sent out, we may presume correctly that an attack is taking place. In the program created, we built a simple multithreaded pipline for data to flow through. The first part is a sniffer, which passes data to a dissector, then the detector analyses the packets with more scrutiny.



So that all traffic is recieved, the sniffer must constantly be listening on the wire for incoming packets. There is no time to dissect before a new packet is recieved, so every packet is immediately put onto a queue to be processed. The detector employs the logic behind what constitutes the integrity of the address resolution protocol for IP addresses on the network. This is where we determine if duplicates are found in the IP/Mac table and determine a fan-out rate for a threshold set number of ARP requests per unit time.

Results Part 1  
To demonstrate the severity of a sucessful ARP spoof, we hoped to simulate a MITM attack. Both the router and the main victim were sent ARP packets, and ip forwarding was set to 'true' on the attacker host.



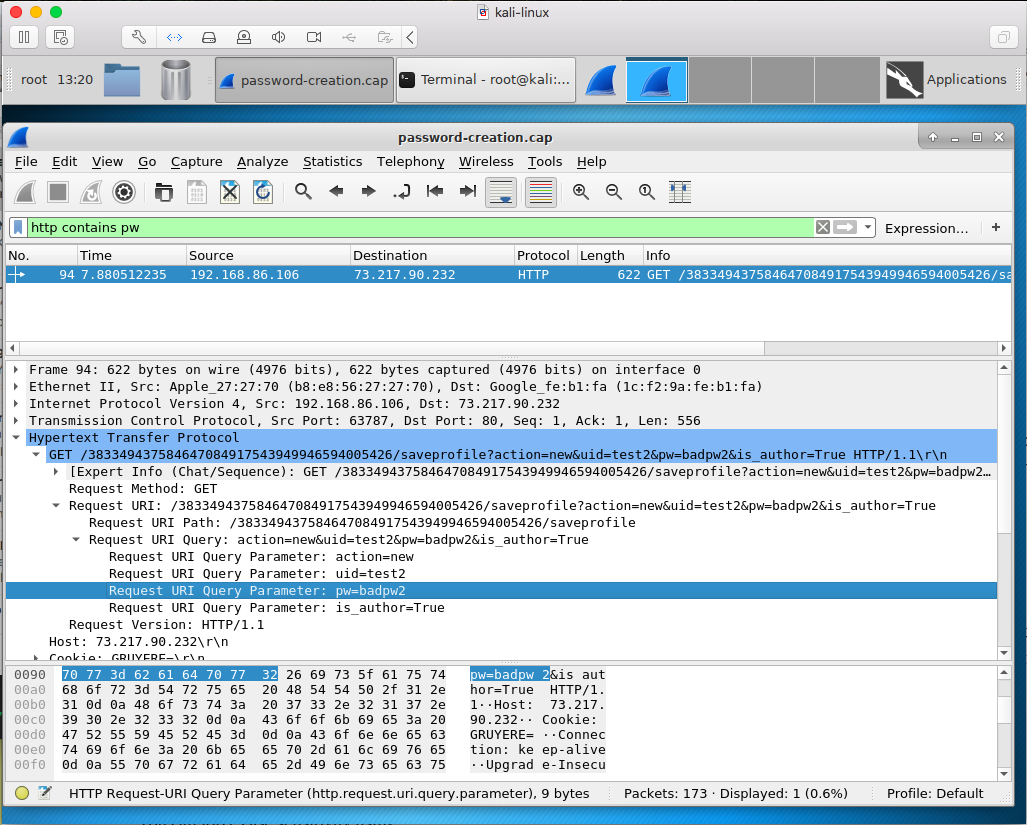
Despite what seemed like a good configuration however, ip packets were not forwarded and we only ended up with a DoS. The DoS we were able to apply by only targeting a single device and is the type of ARP spoofing attack used when we run the spoofer against our detector.

Man in the Middle Attack

Macintosh HD:Users:Phoenixo:Downloads:MITM.png

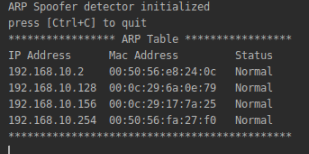
Man in the middle attacks allow the attacker to read and modify packets sent to it. During our testing we found out that as long as we only want to read packet information, listening promiscuously should give the same result. That being said there are a few theoretical differences in the behavior of an ARP spoof listener vs a promiscuous listen on an ethernet connection instead of a WiFi one. The WiFi cannot conrol promiscuous listening as well as the ethernet type connection. The ethernet connection seems to be easier to control because a savvy defender may be listening on the line itself. Any tool the attacker can use, the defender can use as well.

Even though our MITM attack didn't exactly work out, we thought it was interesting to show how an intercepted packet could be used. Below is the result of listening for an account creation from the vulnurable grueyre program. The password given is in plain text and can be read quite easily. Session hijacking is done in a similar way by inspecting the packet for information.

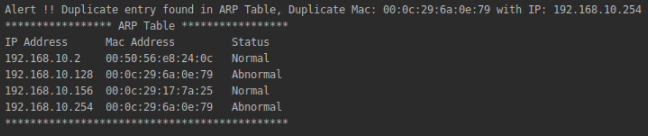


Results Part 2

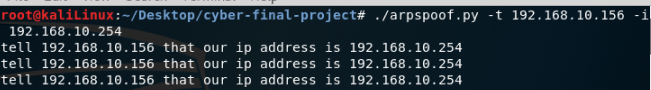
Below is an example of what the custom ARP table looks like when no spoofing is detected in the network. The three main portions to the table are the 1. IP address column, 2. Mac address column and Status Column. The first two columns are self-explanatory, but the status column is a Boolean condition for if the IP/Mac pairing is considered abnormal or abnormal. For an abnormal condition to arise there would have to be a duplicate entry in the table or a fan-out rate that has exceeded the threshold we placed upon it.



The photo below shows an IP/Mac pairing that was deemed abnormal. If you analyze the photo a little further, you will see that the status value for ip address 192.168.10.128 and 192.168.10.254 was triggered as abnormal since these two ip addresses have the same mac address and produce a duplicate in the table.



What’s going on behind the scene is that on the separate ARP spoofer program running on a separate node in the network, we are sending the ICMP message to tell 192.168.10.156 which is the machine that the spoofer detector is running on that our ip address associated with this mac address is 192.168.10.254. This makes an entry in the victims ARP table with the duplicate entry. Our program then parses the table checking for this duplicate.



Also, from the photo above you can get an idea of why we need to check for fan-out rate. In order for our spoofer to work, we would need to send the same message across the network saying, “hey you this is me” or “hey Ip 192.168.10.156 I am IP 192.168.10.254”. Or else the table would revert back to whatever mac address entry actual belongs to that IP-address or nothing if it is not on the LAN.

Fan out rate was then deemed useful since this could be a potential sign of ARP spoofing. This made sense for the research we did into the assignment and also for a countermeasure to what we had to reproduce in the spoofer itself. The Fan out rate calculating function simply steps through every unique IP to IP ARP message intercepted and compares each timestamp associated with the message to determine the threshold. So, for example say a threshold across a time span of 5 minutes with 10 occurrences. So, if we intercept the same IP – IP ARP message 10 times within a 5-minute period than we raise an alert.

Discussion

During the development of this project it was easy to see why cybersecurity is a difficult task. After the detector was created and results were gathered, we decided to continue development of the attacker. One of the changes made to the attacker was a simple arptable rule that prevents real ARP requests from exiting the attacker to reveal its true identity.

arptables -A OUTPUT --source-ip 192.168.86.108 -j DROP

This simple rule allows the attacker to avoid detection on the basis of having a duplicate address, however the rate limiting is still a challenge the attacker faces. In addition, a MITM attack will still yield a duplicate address because the attacker simultaneously spoofs two IPs at once. Perhaps if we were to develop this further, we would find more effective methods of either attacking or defedning, and the most effective strategy could evolve and depend on the strategy of the other.