Tool for ARP and DNS Poisoning with Spoofing Capabilities

2IC80, Group 3, 2021-2022

Ruxandra Hristache, Thanos Papamichail, Manka Huszar

1 INTRODUCTION

Millions, if not billions of people connect to a wireless or wired network every single day, to have access to web apps and communicate with servers. However, the large majority of people are not aware of the vulnerabilities protocols have while connecting to a network, such as having to trust devices inside a network to establish a connection. This facilitates malicious individuals with the ability to fake and mask their identity inside a network, and the possibility of stealing private data from users.

Our group took the opportunity to delve into how such attacks work, and worked to create a software tool which enables a user to automatically ARP or DNS poison IP addresses in a local area network. The software will have options and configurations available to the user, catering to their needs, and it also comes with SSL stripping capabilities. In section 5 we will present the specific way in which the attack is performed by our tool. In Section 8, we provide the link to a video showcasing our tool.

2 ATTACK DESCRIPTION

This section aims to be an introduction into the two protocols on which our tool performs the attacks, Address Resolution Protocol (ARP) and Domain Name System (DNS). After the base has been established for both of them, we proceed to explain the various scenarios in which ill-intended individuals might poison the ARP and DNS records.

2.1 ARP Spoofing and Poisoning

The **Address Resolution Protocol (ARP)** is a Layer 2 protocol that maps the IP address of a device to its MAC address. With this protocol, a device in a local area network (LAN) can learn the MAC address of another device in the same network. An ARP table is maintained by each device which contains the mappings between the IP and the corresponding MAC addresses within a subnet.

The ARP table is the basis of the ARP cache poisoning attack. ARP poisoning refers to the corruption of the ARP table in such way that the attacker's MAC address gets mapped to the IP address of other devices in the LAN. This procedure of "impersonating another machine's MAC address" [2] is ARP spoofing. By pretending to be another device, the attacker is able to receive and modify data packets.

2.1.1 Man-in-the-Middle (MitM) attack

The most popular usage of APR poisoning is performing a Man-in-the-Middle (MiTM) attack. In this scenario, the attacker tries to intercept data flow between two devices, usually the server and a victim machine. For this to work, the ARP tables of both the server and the victim must be poisoned. Hence, all data that is being transmitted between victim and server would instead be sent to the attacker's machine. Moreover, if the attacker forwards the data packets to their original destination, the victim might not notice anything abnormal.

To achieve a consistent MitM attack with ARP poisoning, the spoofed ARP packages need to be resent frequently. This is because if an ARP entry is not used for a certain amount of time (around 1-2 minutes), it gets removed from the ARP table.

2.1.2 SSL Stripping

SSL stripping is a way to convert a website from HTTPS to HTTP, making the website less secure. The three main methods in order to have access to execute SSL stripping are proxy servers, ARP spoofing and network access. In our tool we use ARP spoofing to execute SSL strip.

Since HTTPS encrypts traffic passed through the website via SSL/TLS, information becomes almost impossible to obtain. SSL strip is useful in that manner, because HTTP websites do not use encryption, which enables an attacker to position themselves between a victim and a website, and eavesdrop on all data that is exchanged between them.

However, nowadays many sites are HSTS, which stands for **HTTP Strict Transport Security**. HSTS is a way to defend against SSL stripping, which means that SSL strip does not really work against them. [4]

2.2 DNS Poisoning

The **Domain Name System (DNS)** is an application layer protocol used to translate web addresses from human-readable to IP addresses. It is based on a hierarchical and decentralized naming system and runs on User Datagram Protocol (UDP) port 53 for fast answers.

The DNS servers are the service which facilitate the connection between the domain name and its corresponding IP address, by storing this relationship in a database called the DNS record. This information is then used by web browsers to communicate with the servers of origin, as well as CDN web servers.

To facilitate fast redirecting to the correct domain, many web clients store information in a DNS cache system. When an IP address is retrieved for the first time, this is stored inside the DNS cache, alongside with the corresponding time-to-live (TTL), as it is mentioned in the DNS records. [7]

DNS poisoning is a broad term which refers to the malicious action of spoofing DNS records. There are various means in which an attacker may achieve this. In our tool, we made DNS spoofing via mounting a man-in-the-middle attack [3].

2.2.1 DNS cache poisoning

DNS cache poisoning is an attack in which the DNS cache is injected with incorrect information, so that a query into it will redirect the online traffic to a different web address than originally intended, such as a fraudulent website, which is a copy of the initial destination, which might deploy various malware and steal the user's information.

DNS is built on top of UDP, instead of a TCP, which would facilitate a handshake to open the communication between two hosts. Thus, it

- *Manka Huszar, 1581368, m.huszar@student.tue.nl*
- *Ruxandra Hristache 1537180, r.hristache@student.tue.nl*
- *Thanos Papamichail, 1592378, a.papamichail@student.tue.nl*

whenever a client creates a DNS requests and expects an answer, there is no guarantee that there actually is an established connection between the two hosts or that the receiver is ready to accept the queries.When deploying a DNS poisoning, attackers are impersonating the DNS name servers.

3 Technical Setup

Our technical setup consists of 3 Virtual Machines with all having different IP and MAC addresses, and are in the same Local Area Network (LAN) as each other. This makes ARP poisoning and DNS spoofing feasible.

3.1 User arguments

To facilitate user interaction, the user can start the tool by calling the corresponding command line arguments. The user can access more information about the supported commands via `python3 main.py -h`. All the commands must be called using administrative rights.

- `python3 main.py -A 5` - perform an all-out ARP poisoning which repeats the attack every 5 seconds. The recommended time for this is 10 seconds, due to the fact that ARP caches are restoring every 15 to 45 seconds
- `python3 main.py -A 5 -s` - perform silent ARP poisoning
- `python3 main.py -D 192.168.56.102` - perform DNS attack which will redirect the user to the server IP address specified as a command line argument
- `python3 main.py -D 192.168.56.102 -s` - the program stops since a silent DNS poisoning is not possible

These are all the possible combinations of flags that a user has access to. Any other combination will inform the user that said combination is invalid.

Moreover, considering the ”path” the user chooses, they will be asked for more input throughout the program. During both attacks, the user must choose an interface and hosts to poison. During ARP, the user can choose to perform SSL stripping on the victim. During DNS the user is asked which websites to spoof.

3.2 Interface and host scanning

The tool is capable to scan all the interfaces in the local area network, using the Scapy function `get_if_list()` and it will notify the user if no interfaces were found. Followed by that, the user has to choose an interface on which the tool will perform a host discovery. The host scanning works by sniffing outgoing traffic between the hosts and extracting the source and destination IP addresses. This comes with the limitation that even if a host is up and running, it will not be discoverable by our tool if said host does not produce any traffic. If no traffic is detected, the scan will fail and the program stops.

3.3 ARP poisoning

When the scanning has finished, the user is presented with the IP and MAC addresses of the hosts that were found on the selected interface. If the user chooses to ARP poison, there is a possibility to poison all hosts or manually type a list of hosts. Then, the tool performs a MitM attack for all possible combination of pairs of hosts, one being the victim and the other being the target. Obviously, if only one host is given by the user, then only that host will be poisoned. Because of the ARP cache renewal, it is possible to repeat the attack after every x seconds, where x is specified by the user. When terminating the attack, the ARP caches for all poisoned hosts are restored.

3.3.1 SSL Strip

The tool has SSL Stripping abilities built on top of ARP poisoning. Once ARP poisoning begins, the user has the option to concomitantly perform an SSL stripping attack. In theory, the outcome of this attack is that a HTTPS website has its security protocols removed and all the sensitive information will be transmitted in clear text through an insecure channel. Passwords, usernames, emails are just a few of the possible data that can be intercepted. This data will then be stored in the `sslstrip.log` file on the attacker's machine.

3.3.2 Operational modes - packet forwarding

By default, the tool does not forward the intercepted packets towards their original destination. The argument -s (”silent”) can be used to forward the packets. If this is the case, it is likely that the victims and targets will not notice that an ARP poisoning attack is ongoing, since their communication will not be ”visibly” interrupted. If the -s argument is not added, packets will not be forwarded and then the attack is evident (”noisy”). However, this may still be useful when the intention of the attacker is simply to stop traffic between two hosts.

3.4 DNS poisoning

DNS poisoning can be executed by the `python3 main.py -D SERVER_IP` argument. When DNS spoofing is selected, the tool will sniff for packets and will capture the first packet with a DNS query, and construct a new packet imitating the response of the intended receiver of it.

The string next to -D represents the server IP address to which the traffic on the victim's machine will be redirected to. Followed by that, the user can input inside the program which web addresses they would like to be spoofed. The domain names are transcribed to their corresponding IP addresses through code.

The tool will sniff all packets through the network without discriminating between senders, and will DNS poison the first packet it receives which has a DNS query.

4 Attack Analysis

In this section we present two situations in which our tool could be used.

4.1 Sniffing sensitive data with ARP poisoning and SSL stripping

One scenario in which ARP poisoning can be used in combination with SSL stripping is when an attack wants to gain access to sensitive data, like passwords or user details of a victim. To achieve this, the attacker establishes a MitM attack, which they can do with ARP poisoning. If the victim is trying to send a request to the server, this request will be received first by the attacker who is intercepting the communication. For instance, the victim might try to access their bank account's website. The attacker will forward their request to the server, which will send its response to the attacker. Now the attacker can apply SSL stripping in order to make the bank's web page insecure. After forwarding the insecure web page to the victim, the victim might try to log into their account, unaware of the underlying attack. Since the SSL encryption has been stripped, the attacker can sniff the victim's login details and possibly use them with a malicious intent.

4.2 DNS poisoning for preventing website access.

The DNS spoofing function of the tool can be used to redirect another user on the same network to a website hosted by M2. If a user attempts to access any website, and sends a DNS query, the tool will detect the packet (Since a man-in-the-middle position is established first) and send another packet constantly to the victim which will contain the IP address of M2. That way the user will now be directed to M2's website, meaning that the user now has no access to the website they originally wanted to visit. DNS spoofing can be further used to redirect another user to a malicious website, for identity theft, phishing, or malware infection (which we do not condone).

5 Attack Engineering

5.1 ARP poison setup

ARP poisoning is carried out by Scapy's `sendp()` function to send ARP packets. Once the user has selected the interface and hosts they would like to attack, ARP spoofing packets for every host are constructed. In this ARP packet, the source's MAC address is switched to the attacker's MAC address while keeping the source's IP intact. The destination's MAC address and IP address are also untouched. Looping through every given host allows the attacker to intercept the traffic between all pairs of hosts.

The ARP packets are sent as frequently as specified by the user (in seconds). In the help menu the user can see that the recommended time suggested by the tool is 10 seconds. This is because ARP tables get "renewed" after 15 - 120 seconds [1] [5]. Hence, sending the packets every 10 seconds should be able to keep the attack alive.

5.1.1 SSL Strip

The SSL strip attack is conducted while ARP poisoning the victim. In essence, it is just a simple linux command (`sslstrip`), which requires a specific set-up before starting it. That means that IP port forwarding must be enabled, so that all TCP traffic from port 80 can be routed to SSL strip's port 10000 [4].

5.1.2 ARP cache restoration

Cache restoration is done by sending packets with the true MAC and IP addresses for each source and destination, hence restoring their ARP caches. Since the ARP caches clear unused entries after a while, eventually the same restoration would happen naturally.

5.2 DNS poison setup

In order to achieve DNS poisoning, the attacker already needs to be man-in-the-middle, so that they can be able to monitor communication between the victim and a server. Also, IP forwarding must not be active, because otherwise the packet will keep being transmitted until it reaches its destination IP. This is why using the silent option will not work for this attack.

DNS poisoning starts by sniffing packets on an interface, and waiting for a UDP packet on port 53, since that is where DNS runs as mentioned in section 2.2. When a packet fulfilling the criteria is sniffed, the DNS spoof function executes. This function re-constructs the packet that was sniffed, apart from the destination and source IP, which are switched, and the `rdata` value which indicates the domain name of the website requested by the victim. The newly constructed packet is sent back to the victim, who now tries to access the spoofed website, but is redirected to a different domain. The packet is sent in a loop until the user of the tool exits it.

6 Obstacles during the making of the tool

Even though we have grasped the topics of the attacks our tool is meant to do, unfortunately, we have not managed to successfully implement SSL stripping and DNS poisoning. The code for both attacks is ready and we suspect that there should be no issues with the code itself.

The main difficulty we ran into during implementation was the behavior of M1. We noticed that with certain network settings configured in VirtualBox, M1 receives no internet and hence, during DNS poisoning, when we want to redirect the victim from a certain domain, say `tue.nl`, Internet Explorer tells us that M1 is offline and no page can be accessed, even before the redirection took place. Due to this, we could not truly test SSL stripping either, since we could not even connect to any domain.

Some of our attempts to solve these issues:

1. In our first attempt, we used a shared folder to install an old version of Firefox on M1. This was successful and Firefox could take us to most domains, though not to .nl domains due to missing protocols. However, DNS poisoning or SSL stripping did not work here either.
2. Our next attempt was to play with the network settings of M1 in VirtualBox. We set Adapter 1 from Host-only adapter to NAT and we tried both setting PCnet-PCI II and PCnet-Fast III. In these situations M1 finally received internet in Internet Explorer and when trying to load tue.nl (for example), it was not showing an offline message anymore. However, with these network settings, M3 was not able to discover M1 on the same interface and M3 received no replies by pinging M1 either. We also read the documentation from Oracle VM VirtualBox to learn more about the usage of each [6]. Even when we carefully followed the instructions there, the connection was still not working properly. We could not carry out our attacks after those attempts either.
3. Our next attempt was to download a completely new Windows XP VM. We used the same network settings as for M1 originally (with Host-only adapter) and this machine had no issues with the internet even with these settings. This machine received an IP of 192.168.56.104 but it was also not discovered by M3 on the interface.
4. To see if the issue is truly with M1's configuration or our code, we also tried running code with the same purpose (ARP poison + DNS poison) different from ours. This code was tested to be running the attacks perfectly on another highly similar setup. Strangely, the ARP attacks were done without issue just like with our code but the DNS attacks did not come through again, due to the unsolved issue with M1's internet connection.
5. Finally, we also looked into whether our host machine (192.168.56.1) could be used as a victim. The ARP poison worked but DNS did not. We also tested SSL stripping, without success, but we learned that on newer systems SSL strip might not work at all due to the usage of HSTS.

7 Conclusion

In conclusion, the tool we imagined is capable of scanning the interfaces on a network, selecting all found hosts or a portion of them manually, and attacking these hosts. The attack can be either a continuous ARP attack with the possibility of SSL strip, or a DNS attack which builds on top of the ARP attack.

There are many ways in which the tool could be improved. The option for choosing multiple interfaces to discover hosts on could be present and the user could have the option to spoof all domains when doing a DNS attack (although this latter may take a longer time to add to the tool). Furthermore, whitelisting domains could be an extra addition.

We are disappointed that we could not deploy our tool in the end. We believe we put effort into making it work (after many hours of individual/team debugging and trying possible solutions) and we implemented the attacks in code well enough.

8 External links

- The video about our tool can be accessed via: `https://youtu.be/ZG33-Z04x8Q`. We recommend using x1.5 or x2 speed.
- We also want to share our GitHub repository for the project: `https://github.com/ruxache/2ic80-project/`

References

[1] Arp caching behaviour, https://docs.microsoft.com/en-us/troubleshoot/windows-server/networking/address-resolution-protocol-arp-caching-behavior.

[2] Arp poisoning: What it is how to prevent arp spoofing attacks, https://www.varonis.com/blog/arp-poisoning.

[3] Dns cache poisoning, https://www.cloudflare.com/learning/dns/dns-cache-poisoning/.

[4] How ssl strip works, https://www.https.in/ssl-security/how-ssl-strip-work/.

[5] Network command reference, https://netref.soe.ucsc.edu/node/20.

[6] Virtual networking, https://www.virtualbox.org/manual/ch06.html.

[7] What is a dns server, https://www.cloudflare.com/learning/dns/what-is-a-dns-server/.