Eindhoven University of Technology
Department of Mathematics and Computer Science

Project Report
Lab on Offensive Computer Security
2IC80 - 2018/2019

A practical example of automating ARP poisoning and DNS spoofing

Bouke Bosma - 0962750

Koen Teuwen - 1245830

Abstract

This report shows how techniques like ARP poisoning and DNS spoofing can be deployed in practice on an automated basis requiring almost no user interaction. Furthermore, the characteristics of an attack with the chosen approach are shown. Furthermore some practical use cases that could be deployed by an attacker are given.

Repository: `https://github.com/Koen1999/2IC80-Project`
Demonstration: `https://youtu.be/Vbff_NE_RN0`

April 9, 2019

Contents

1 Introduction

Trusting computers and the networks over which they operate is continuously done by many people, all over the world. One may wonder whether that trust is misplaced. In fact, even the protocols used to operate over those networks assume they can trust other devices on the network. This definitely leaves open opportunities for attackers, and thus the trust can be considered misplaced.
The goal of the project was to develop a hacking tool that automates ARP and DNS spoofing, The attack performed by the tool will then be reproduced in a virtual environment and analysed. This reproduction and the analysis of the attack is important to understand the characteristics of such attacks to allow for possible mitigation. The tool developed is fully-automated and applicable in different scenarios.

2 Attack description

2.1 ARP spoofing

2.1.1 One-way ARP spoofing

Address Resolution Protocol (ARP) is a protocol that allows devices on a local area network (LAN) to learn the MAC-address of another device within that LAN given its IP-address.
It is possible for an attacker to let other devices associate the MAC-address of the attacker with the IP-addresses of other devices. This linking of a wrong MAC-address to other IP-addresses is known as ARP spoofing. ARP spoofing can only be done when the attackers device is connected to the LAN or if a device connected to the LAN is compromised by an attacker. If these conditions are met for a device, all packets that are meant for that device will be sent to the device of the attacker instead, meaning the attacker can forward, reroute, read, modify and stop all packets to their own liking.

2.1.2 Two-way ARP spoofing

To perform a Man in the Middle (MitM) attack, say between a client and a server, both the ARP cache of the client and the server have to be poisoned. Both victims think they are communication with each other and are not necessarily aware of the presence of the attacker in the middle.
The victims are less likely to become aware of any irregularities provided that the attacker correctly forwards all packages and no information appears to be lost for the victims.

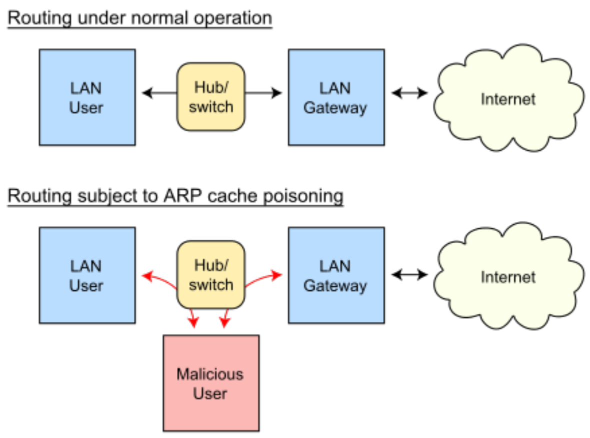

Figure 1: An abstract overview of a MitM attack, as possibly performed using two-way ARP-poisoning [1]

2.1.3 Continuous ARP poisoning

As the ARP cache of a device is only valid for a limited amount of time, the spoofed ARP entry will be automatically dropped from the victims ARP table if there has been no contact with the spoofed IP-address for a while. Therefore the results of the poisoning will not be persistent.
To achieve persistency, one can continuously poison the victim, i.e. resend the ARP package at some rate. The rate depends on the time the ARP cache is valid. Another approach to make the spoofing more persistent would be to listen for ARP requests and send spoofed responses only after a request was received. This is a more subtle approach, but on the other hand the host being spoofed will also receive the request and send a reply for it. Depending on which packet gets to the other host last, the ARP cache is filled. To be sure the ARP cache is poisoned, retransmitting would likely be necessary.
It may be possible that some hosts on the network will fight back by sending valid ARP responses when noticing that the attacked sends spoofed ARP responses. This however, is not a major issue, because the spoofed responses are resent frequently anyway. It is however likely to cause lost packages.

2.2 DNS spoofing

Domain Name System (DNS) is a protocol that translates addresses that are easily readable by humans (like `educationguide.tue.nl`) into numerical IP-addresses. With these IP-addresses devices are able

to locate and identify digital services and devices using the underlying network protocols.
The client first contacts an arbitrary DNS server known for the client. If the location of that domain is stored in its cache or it is an authoritative server for that domain, it will send a response to the client containing the location of the domain. If not, it will recursively forward the DNS query to another DNS server. Once a a response is received, the location of the domain will be cached in a recursive DNS server. Following requests can then be answered more rapidly, as the recursive server will not have to contact any other DNS servers.
If an attacker blocks the response on the request of the recursive DNS server and sends a spoofed response containing the IP address of the attacker, the recursive DNS server will link the IP-address of the attacker to the requested address. All traffic that was meant to go to the requested domain will now be diverted towards the attackers device. This is known as DNS spoofing.
Once traffic has been diverted towards the attacker, a malicious payload can be loaded, phishing attempts can be made and/or the client can be forwarded to another domain.

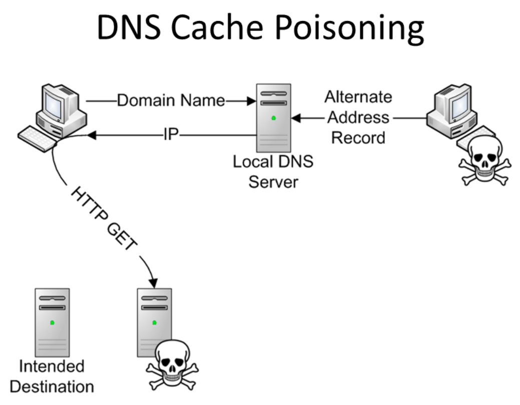

Figure 2: A schematic overview of DNS spoofing [2]

3 Technical setup

3.1 Environment

The tool is developed using Python 3.7 and uses PyY and Scapy as packages. Scapy is used as framework to send, receive and manipulate packages. PyY is used for determining whether a given IP-address is private.
The tool is intended for execution on a Linux machine. It was tested using three virtual machines, one Ubuntu attacker machine, one Windows XP client machine and one web server. All machines are located on the same local area network.

3.2 Phases of execution

The tool first launches a discovery phase which is ran for a while, after which the spoofing phase is started. The spoofing phase consists of ARP poisoning, DNS spoofing and forwarding of intercepted packages. After the attack comes the restoration phase.
The discovery phase may continue after initialisation of the spoofing phase, if the attacker specifies so. Hosts joining the network after the spoofing phase was initialised will then be poisoned as well.

3.2.1 Discovery phase

Using Scapy, all packages sent over the IP layer are caught and analysed. Both the source MAC and IP addresses are extracted from the package. The tool determines whether these addresses are both distinct from the MAC and IP addresses of the attacker and then proceeds to check whether the given IP is a private IP address. If all checks pass, the pair containing the MAC and IP addresses is stored as a host that is up. The same is done for the destination MAC and IP addresses.

3.2.2 Spoofing phase

ARP poisoning: The attacker can choose to attack all hosts online on the network, to only attack a selection of the hosts or to attack all hosts except for hosts that are on a whitelist that the attack may compose.
The tool determines all combinations of the targeted hosts. All hosts that should be attacked are then two-way ARP poisoned. An ARP package is sent from the attacker to both victims.
Continuous spoofing is used to make the attack lasting. The above two-way poisoning is repeated at a frequency the attacker can choose.

DNS spoofing: The attacker can choose to spoof all domains, whitelist some domains or to select specific domains to be spoofed. Furthermore, the attacker can choose to which IP address the clients should be sent instead.
The tool listens for DNS queries on the network. If the query is for a domain that should be spoofed, the attacker will spoof a DNS response coming from the target of the DNS query. The answer will contain that

the domain is located at the IP address the attacker specified instead of the actual IP address that belongs to the domain. The time-to-live of the response is set highly, such that the effects are more lasting.
If the DNS query is forwarded on the network, and the attacker detects this, the attacker will also send a response for the forwarded query.

Packet forwarding: The tool listens to all packets sent to the interfaces of the attacker using Scapy and checks whether the destination IP address also corresponds to the attacker. If the IP address does not match, then this package was intercepted because of the poisoned ARP caches.
The user has three options, forward all packages, forward no packages or forward all packages except valid DNS responses of domains which the tool tries to spoof.
The last option mitigates detection of the attack and maximises effectivity of the DNS spoofing at the same time.

3.2.3 Restoration phase

If the attacker wishes to restore ARP caches after the attack is called off, the tool will send out spoofed ARP responses, but with the correct MAC-IP pairs as found during the discovery phase.
Restoring ARP caches after calling off the attack also mitigates detection. After the attack the attacker is no longer forwarding packages, which would result in lost packages if ARP caches were not restored.

3.3 Automation and scaling

The tool automatically detects all available network interfaces on the attacking device. The phases described above are then launched on all interfaces selected by the client. The tool uses multiple threads to listen and send packages on multiple interfaces simultaneously.
The tool automates all parts of the attack, except for some essential options that the attacker must specify. At the same time, it is flexible and allows for possible extension or partial replacement by other tools.
Depending on the location of the attacker on the network and the amount of hosts reachable, the tool scales very well. Also, as the packages sent are small, there are no real limitations to the amount of targets due to bandwidth.

4 Attack analysis

In this section we showcase two scenarios that demonstrate possible uses of the application by an attacker.

4.1 ARP poisoning: Keeping an eye

Person A is a father of child B living at his home. Consider A wants to get to know until what time B is still active one the internet, as well as what B is doing on the internet. To find out A starts ARP spoofing on his LAN. At first A performs a Man in the Middle attack to make sure B does not know that A is checking his internet usage.
By doing this he finds out that B is watching movies on Netflix until one o'clock at night. A is now aware of what B does at night at his phone, as well as at what time B stops using his phone and presumably goes to sleep. As A thinks that B should get more sleep he decides to block all traffic coming from B his phone after midnight.
This scenario gives an example of how ARP poisoning can be used to detect and prevent internet traffic according to own liking on your own LAN, in this case with no malicious intentions.
Of course person A could have setup a firewall on the network as he owns it, but this section is merely about a possible application of ARP poisoning.

4.2 DNS poisoning: Login credentials

Person A is mad at person B for spreading rumours about him. A does not accept this and wants revenge on B. A wants to do this by hacking into the Facebook account and change the profile picture of B with a very ugly photo of B that A already has in his possession. To do this A the credentials of B.
At first A starts a Man in the Middle attack with B to make sure he is able to DNS spoof B. Once A has initiated this he forwards all traffic except when A receives the correct IP-address of `facebook.com`. At that point A spoofed DNS response to B with his own IP address is sent. Person B hosts, a web server with a page that looks the same as the one from Facebook, but is controlled by A. Once B attempts to log in on this site, not knowing that he is not logging in to Facebook, A will receive the credentials of B. Now A can log in to the Facebook account of B and change whatever he likes.
This example shows how a person can use DNS spoofing to intercept credentials. This example is fairly innocent (although being illegal), but there exist

known applications in which bank credentials were targetted.

5 Attack engineering

When starting the tool a few variables and default settings are set. After this the program will ask the attacker which interfaces to discover and attack on. Depending on the choice of the attacker the program will then start discovering new hosts on the chosen interface(s) for the indicated amount of time.
Once the tool is done with the discovery phase, it will ask the attacker which hosts to target and at which frequency. Next it will ask which packets to forward. Furthermore, it will ask the attacker which domains to spoof and to which IP-address to redirect the clients. Lastly it will prompt the attack whether to restore ARP caches after the attack and whether to continue discovery while poisoning.
The tool makes use of different threads for all interfaces and tasks. For each interface it will have a thread forwarding packets, DNS spoofing, ARP poisoning and discovery. Running multiple threads enables the tool to listen for different packets on multiple interfaces at the same time.
After setting these settings, the tool will proceed to poison ARP caches and spoof DNS requests as specified without any further interaction with the attacker.

5.1 ARP poisoning

The continuous two-way ARP poisoning is done using Scapy. For all ordered pairs of distinct victims 1 and 2, the following packet is sent to victim 1hosts at the given frequency:

Field name	Content
hwsrc	attacker MAC
psrc	victim 2 IP
hwdst	victim 1 MAC
pdst	victim 1 IP
op	is-at

Table 1: An overview of an ARP package sent to victim 1, making victim 1 sent all packages meant for victim 2 to the attacker instead

During the spoofing new hosts can still be discovered. If this is the case, the ARP poisoning thread will pick this up immediately and start spoofing these hosts as well, if indicated by the user.
Once the user interrupts the tool, the ARP caches of all victims will be restored if specified.

5.2 DNS spoofing

The tool will use Scapy again for the DNS spoofing. It listens for DNS requests sent on the LAN. Since the victims are ARP poisoned, all packets are routed through the attacker. If the attacker has indicated that a particular domain needs to be spoofed, and a DNS query for that domain is sent, the tool will act upon that by spoofing a response from the target of that query. The tool will spoof the response such that it appears to be sent by the IP-address of the target of the DNS query. The location the domain is said to be at, is not the actual domain but a domain specified by the attacker.
After receiving this DNS response, the client will go to the IP-address specified by the attacker instead.

5.3 Forwarding

The tool listens for packages sent to the attacker's interfaces, of which the target IP-address does not match the actual IP-address of the attacker. This indicates that the package had been intercepted.
If the package is a valid DNS response for a domain the tool tries to spoof, it will be dropped if specified by the attacker.
If not, or if another package, the tool will forward it to the correct host by modifying the destination on the Ethernet layer and subsequently retransmitting the packet. For finding the correct destination, the information gathered during the discovery phase is used.

6 Conclusion and future work

The tool effectively is able to spoof other hosts and domains by (mis)using the ARP and DNS protocols in an automated way. Especially the ARP poisoning may cause detection of an attack as packages have to be sent at a regular basis while DNS spoofing requires a lot less traffic.
Further improvements may be made in the area of detection prevention and by improving interaction with the attacker by means of a GUI.

References

[1] 0x55534C. Arp poisoning. `https://commons.wikimedia.org/wiki/File:ARP_Spoofing.svg`, 2011. `https://creativecommons.org/licenses/by-sa/3.0/legalcode`.

[2] google. Dns spoofing. `https://commons.wikimedia.org/wiki/File:Dns-cache-poisoning.png`, 2016. `https://creativecommons.org/licenses/by-sa/4.0/legalcode`.