Default Project

2IC80 - Lab On Offensive Computer Security

Julien Erbland (2232243) - Cedric Odille (2234256) - Daniël Braamskamp (1533371)

May 2025

Contents

1 Midterm submit

Here is the first draft of your implementation of the default project for course 2IC80. Our plan is to release a usable tool made for ARP & DNS spoofing using SSL stripping to downgrade the connection. We also want to implement a new interface where we can type the IP address and launch all the spoofing process from an API other than the terminal. Once launched, our program should automate the spoofing process by itself and reproduce it in a virtual environment. It should be able to adapt to different IP address and hosts number.

2 Introduction

Introduction will be made once we finish everything and we have a clear overview of what we achieved.

3 Attack Description

This project combines three classic network attack techniques: ARP spoofing, DNS spoofing, and SSL stripping. Together they allow an attacker to silently intercept and manipulate a victim internet traffic inside a local network. The attack chain is designed to be fully automated and work with minimal user input.

3.1 ARP Spoofing

ARP spoofing targets the way devices in a local network discover each other MAC addresses. When a device wants to send data to an IP address, it uses ARP (Address Resolution Protocol) to ask “who has this IP ?” The device with that IP replies with its MAC address. However, ARP has no security checks and thus simply accepts any response.

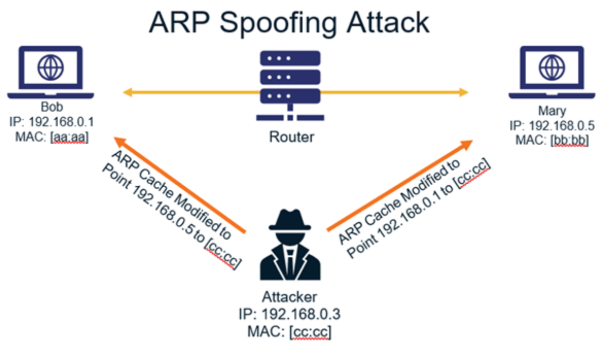

Figure 1: ARP spoofing schema

An attacker can take advantage of this by sending fake ARP replies, telling a victim "I am the router" and telling the router "I am the victim". As a result, both devices send their traffic to the attacker instead of directly to each other. This places the attacker in the middle of their communication also known as a Man-in-the-middle (MITM) position.

To make sure the attack remains active, the spoofed ARP replies are sent repeatedly. Otherwise the victim ARP caches may correct themselves.
Need to add more details on our implementation, specific things : different arp type (2ways or continuous..) or how to prevent arp (maybe in the conclusion)

3.2 DNS Spoofing

Once the attacker is positioned between the victim and the rest of the network, the next step is DNS spoofing. DNS (Domain Name System) is used to translate domain names like `google.com` into IP addresses. If an attacker can trick the victim into getting a fake IP for a domain, the victim will be redirected to a server controlled by the attacker.

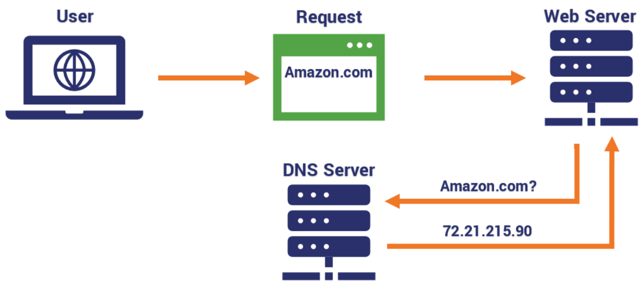

Figure 2: DNS spoofing schema

The goal of our attack would be the following (not implemented yet). We monitor the network for DNS requests from the victim. If the request is for a target domain (for example, `facebook.com`) our tool quickly sends a forged DNS reply with the attacker's chosen IP. If the spoofed reply arrives before the real one, the victim will use the fake IP and unknowingly connect to the attacker's server.

This type of spoofing works best when combined with ARP spoofing, since it gives the attacker access to all victim traffic, including DNS requests. *More details on our implementation, specific things*

3.3 SSL Stripping

Many websites today use HTTPS to protect user data with encryption. However, SSL stripping is a technique that forces the victim to use HTTP instead, which has no encryption. This is possible because the initial connection to a website often starts in plain HTTP before being redirected to HTTPS.

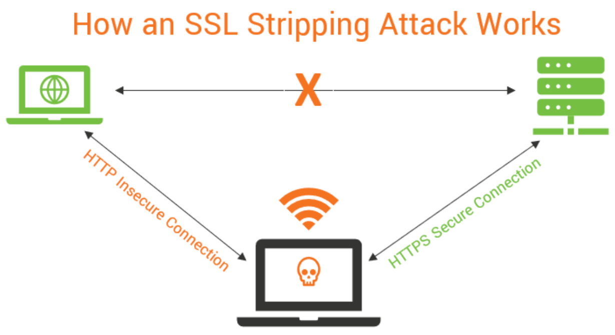

Figure 3: SSL stripping schema

The attacker can take advantage of this by intercepting the initial HTTP request and preventing the HTTPS upgrade. At the same time, the attacker connects to the real website over HTTPS and relays the data back and forth. The victim sees a normal webpage but is actually sending data (such as passwords) over an unencrypted connection.

The goal for the project would be the following (not implemented yet) : use a proxy that listens for incoming HTTP traffic from the victim and forwards it to the real server using HTTPS. *The proxy is the standard method maybe try other (TCP paquet injection, or use DNS based https downgrade if the client hasn't cached HSTS).* The proxy should removes security headers like HSTS (HTTP Strict Transport Security), so the victim stays on HTTP. This makes it possible for the attacker to read everything the victim sends, including login credentials.

4 Technical Setup

To safely develop and test our attack tool, we created a virtual lab environment using VirtualBox. All virtual machines (VMs) were connected to the same virtual local network to simulate real-world conditions without causing any harm to actual systems.

We used the following three machines in our setup:

- **Attacker VM:** Runs Ubuntu 22.04. It executes our Python scripts and hosts the spoofing tool.
- **Victim VM:** Simulates a regular user, for example by browsing websites or sending DNS queries.
- **Gateway/Server VM:** Acts as either the default gateway or a server the victim communicates with.

All machines were configured on a shared LAN using the VirtualBox "Internal Network" mode, allowing full packet visibility and control.

4.1 Tools and Configuration

The attacker machine includes:

- Python 3 and the Scapy library for crafting and injecting packets - - (Optional) Wireshark for analyzing packets *Still need to complete it*

4.2 Execution Flow

Once the environment was ready, we followed these main steps:

1. The tool scans the network to find devices and their IP/MAC addresses.
2. It launches ARP spoofing to become a man-in-the-middle between the victim and the gateway.
3. DNS requests from the victim are intercepted and responded to with fake IP addresses.
4. HTTP traffic is redirected to a proxy, which strips SSL and lets us read sensitive data like login credentials.

5 Attack Analysis

5.1 ARP Spoofing

5.2 DNS Spoofing

5.3 SSL Stripping

6 Project Goal

- Does it involve a non-trivial attack scenario worth studying in depth?
  Example of trivial: "flooding an IP address with packets XYZ"
- Can the attack be practically analyzed or reproduced?
  Example of not doable: "I want hack the NSA"
- Is reproducing/analyzing the attack non-trivial?
  Example of trivial: "prj=copy-pasting solution from ARP laboratory or GitHub"