

CSCD58 Project Proposal – Packet Sniffer

Ahmed Halat [1006332951, halatahm]

Mohamed Halat [1006322962, halatmoh]

Armand Sarkezians [1006020574, sarkezi1]

Description

As our final project, we've decided to build a network packet sniffer, like Wireshark, with additional functionalities relating to the material covered in this course. These additional functionalities could include listing available hosts on the network with their whoami data, showing ARP tables for the network, and more.

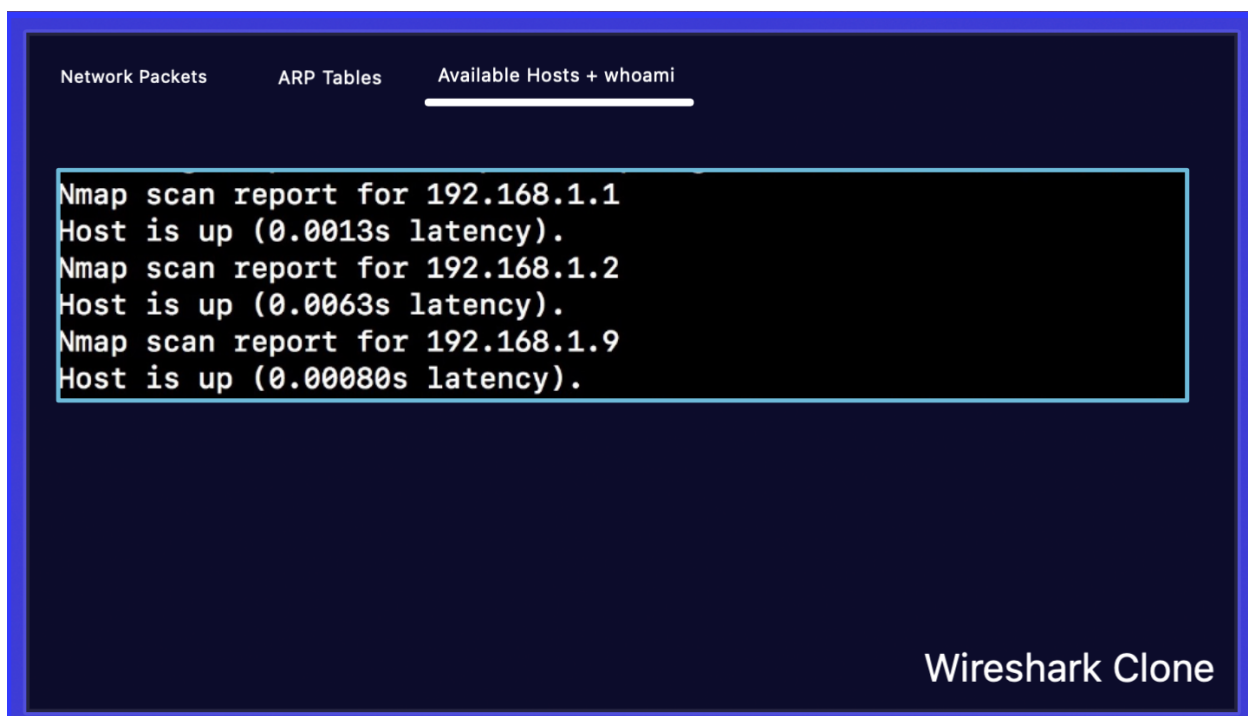
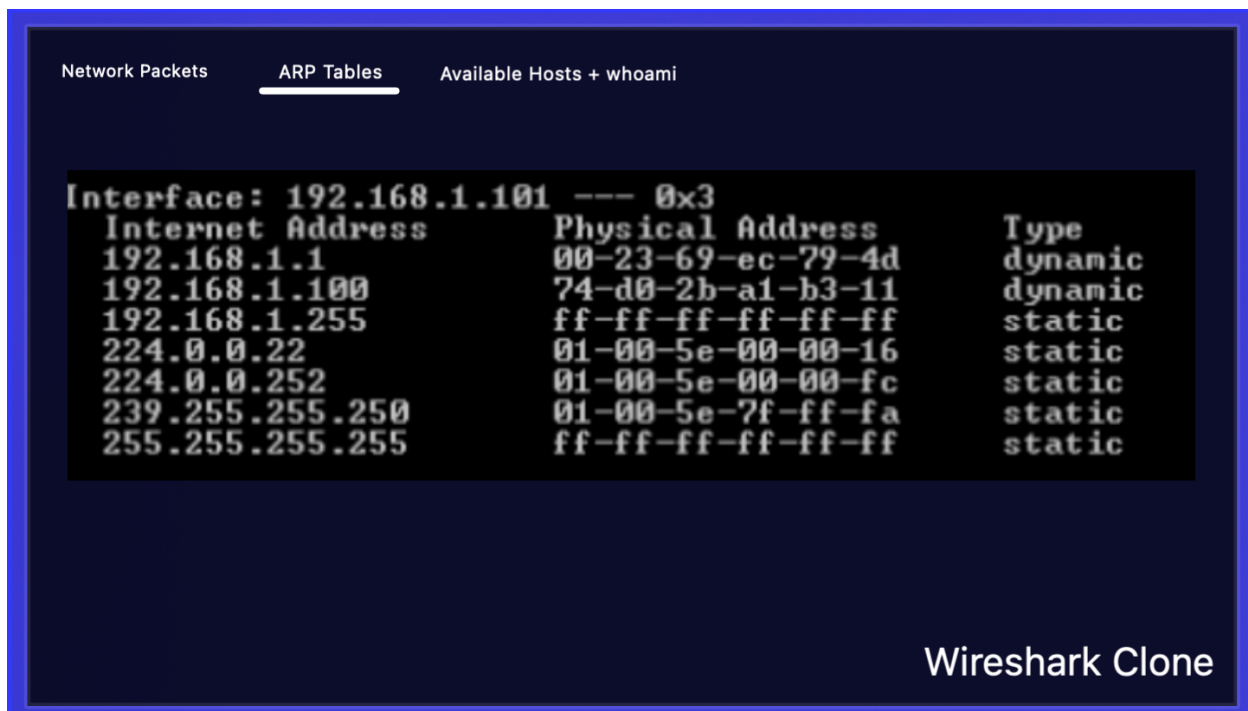
Our project would be divided into subpages, each with their own feature. The packet sniffing page would have listed data like Wireshark, including information like protocols, source, destination, payloads, and timestamps. The ARP table page would allow you to see your hosts' ARP table, whose information you can see in the available hosts + whoami page. This page would include as much information about each host that we can get our hands on.

Our packet sniffing implementation would also include some of the features that Wireshark offers, including but not limited to filtering based on protocol, selecting which interface to sniff on, and saving the network information in a file.

After selecting which interface to sniff on, our implementation would look somewhat like this (keep in mind this is something we sketched in 10 minutes, it may look very different based on design choices):

The screenshot displays a web-based application titled "Wireshark Clone" with a dark blue background. At the top, there are three tabs: "Network Packets" (selected), "ARP Tables", and "Available Hosts + whoami". To the right of the tabs are "Save" and "Filter" buttons, each with a circular icon. The main area shows a list of network packets with columns for packet number, time, source IP, destination IP, protocol, and details. The packets are numbered 222 to 246. The details column shows various protocols like UDP, TLSv1, and TCP, along with their respective lengths and application data. The interface is clean and modern, with a focus on readability and ease of use.

Packet No.	Time	Source IP	Destination IP	Protocol	Details
222	10:54:05.531337	35.186.224.16	150.148.128.183	UDP	1288 443 → 52969 Len=1246
223	10:54:05.531678	35.186.224.16	150.148.128.183	UDP	1292 443 → 52969 Len=1250
224	10:54:05.531681	35.186.224.16	150.148.128.183	UDP	1292 443 → 52969 Len=1250
225	10:54:05.531902	35.186.224.16	150.148.128.183	UDP	534 443 → 52969 Len=492
226	10:54:05.531904	35.186.224.16	150.148.128.183	UDP	1287 443 → 52969 Len=1245
227	10:54:05.531905	35.186.224.16	150.148.128.183	UDP	544 443 → 52969 Len=502
228	10:54:05.531907	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
229	10:54:05.532007	35.186.224.43	150.148.128.183	TLSv1	1166 Application Data
230	10:54:05.532044	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
231	10:54:05.532047	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
232	10:54:05.532082	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
233	10:54:05.532138	150.148.128.183	35.186.224.43	TCP	66 53503 → 443 [ACK] Seq=1 Ack=1401 Win=5635 Len=0 TSval=16
234	10:54:05.532230	150.148.128.183	35.186.224.43	TCP	66 53503 → 443 [ACK] Seq=1 Ack=2501 Win=5618 Len=0 TSval=16
235	10:54:05.532261	150.148.128.183	35.186.224.43	TCP	66 53503 → 443 [ACK] Seq=1 Ack=5301 Win=5574 Len=0 TSval=16
236	10:54:05.532286	150.148.128.183	35.186.224.43	TCP	66 53503 → 443 [ACK] Seq=1 Ack=6701 Win=5552 Len=0 TSval=16
237	10:54:05.532310	150.148.128.183	35.186.224.16	UDP	77 52969 → 443 Len=35
238	10:54:05.532419	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
239	10:54:05.532420	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
240	10:54:05.532422	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
241	10:54:05.532424	35.186.224.43	150.148.128.183	TLSv1	266 [TCP Previous segment not captured] , Application Data
242	10:54:05.532425	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
243	10:54:05.532427	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data
244	10:54:05.532429	35.186.224.43	150.148.128.183	TLSv1	1466 Application Data [TCP segment of a reassembled PDU]
245	10:54:05.532469	150.148.128.183	35.186.224.16	UDP	75 52969 → 443 Len=33
246	10:54:05.532493	150.148.128.183	35.186.224.43	TCP	66 53503 → 443 [ACK] Seq=1 Ack=10901 Win=5487 Len=0 TSval=16



(The available hosts + whoami page is just Nmap only for the purposes of this proposal)

Rationale

The rationale behind this implementation is simple. We feel as though this project will allow us to apply a large amount of the knowledge we've gained throughout this semester. In

addition to this, it would be extremely cool to learn how an application like Wireshark works under the surface.

Goals and Targets

Some of our goals for this project include:

- Develop a packet sniffer
- Develop an ARP table reader
- Develop an information page regarding available hosts on the network along with all data we can gather on them
- Develop a nice-looking UI
- Allow users to save network traffic into files
- Allow users to filter through network traffic with different parameters

Relation to Computer Networks

The relation of this project to computer networks is vivid: Wireshark is a vital tool when it comes to detecting network traffic, and we want to create a similar tool. This project would allow us to dive deep into how network traffic is parsed by our computers, how we can access it as users and how to analyze it. It would remind and educate us more on the different protocols that exist in the networking world, and the meanings/data behind each one. While working on this project we would touch on ARP tables, a prevalent topic in Computer Networks. Depending on how many additional features we can implement, it may allow us to touch on more topics under the scope of Computer Networks.