2021

Functional Specification
Network Sniffer

CW258 CYBERCRIME & IT SECURITY

ERLANDAS BACAUSKAS

INSTITUTE OF TECHNOLOGY CARLOW | Supervisor: Paul J. Barry

Abstract

From year 1986 network sniffers was introduced to our era of computing. It helped system administrators to investigate and gain better knowledge of networks they were working on. The main reason why packet sniffers are used is to diagnose underlying problems on the network by analyzing network traffic. The functionality of packet sniffer is simply that it monitors traffic and presents results in a human readable format. However, my goal is to produce a tool that was never seen before by implementing elements of packet sniffing. This tool would be targeted to anyone who is interested in cyber security. My plan is to produce a tool that would have not only packet sniffing purpose included but also that it could be used for different kinds of things. I am planning to implement a few additions to already existing packet sniffers. As there is an open-source product called Wireshark and I could say that Wireshark is the number one choice for most of us who is interested in analyzing network traffic. I want to a make tool that would have basic Wireshark functionality and some added extras. Extras would include port scanning, DNS spoofing, ARP poisoning and host discovery.

Table of Contents

Introduction

This document will outline the functional specification or my chosen project and what I ultimately plan to achieve. I will discuss the system requirements that will be needed for my project as well as the components of my application. I will provide sketches of how I envision my application to look and what functionalities it will have once completed. I will also include use case diagrams to provide a better understanding of how the application will work.

Functional Specification Scope

This document will outline how a developed network sniffer will function. This document will cover all of the functionalities related to this tool. Needed actors will be outlined that interact with this technology. Using a use case diagram, this document will explain the steps needed for interaction with the system. Examples will be provided of assumed actors input and expected output. Under each functionality there will be a desired design provided to have a feel of how this application will interact with actors. All advantages and disadvantages will be clearly stated in this document showing what this tool is able to achieve and its shortfalls. Focus of this document is to touch on all aspects of this tool and present it in a non-technical and easy to follow manner.

Requirements

System Requirements

This tool will be running on two major platforms that includes Linux and Windows. Unfortunately, I don't have the ability to test this tool on iOS platform, but I assume it should work on Linux as it is very similar. If there will be any extra requirements to run this tool, these requirements will be provided in a format of a text document within Git repository together with a guide within Git page.

Functional Requirements

No.	Function	Description	Cruciality	Dependencies
1.	Login/register	Allow user to login and sign up	High	
2.	Network sniffer	Capture network traffic on desired interface	High	No.1
		Save captures to a .pcap file		
		Open .pcap file for analysis		
		Stop network sniffer		
		Provide interface		
		Provide filter		
3.	Host discovery	Discover alive hosts within given subnet	High	No.1
4.	ARP poison	Perform ARP poisoning attack on a given host	High	No.1,3
5.	DNS spoof	Consecutive to ARP attack, spoof DNS records for given (same as ARP) IP address	High	No.1,4
6.	Notes	Quick notes if needed	Very Low	No.1

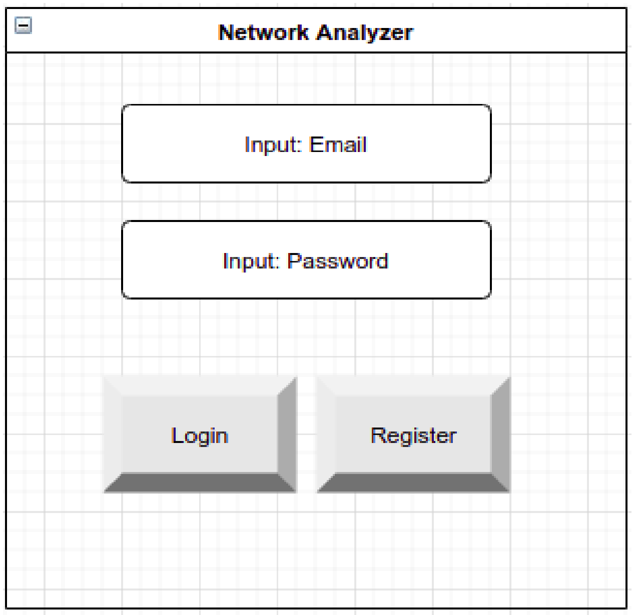
Components of an Application

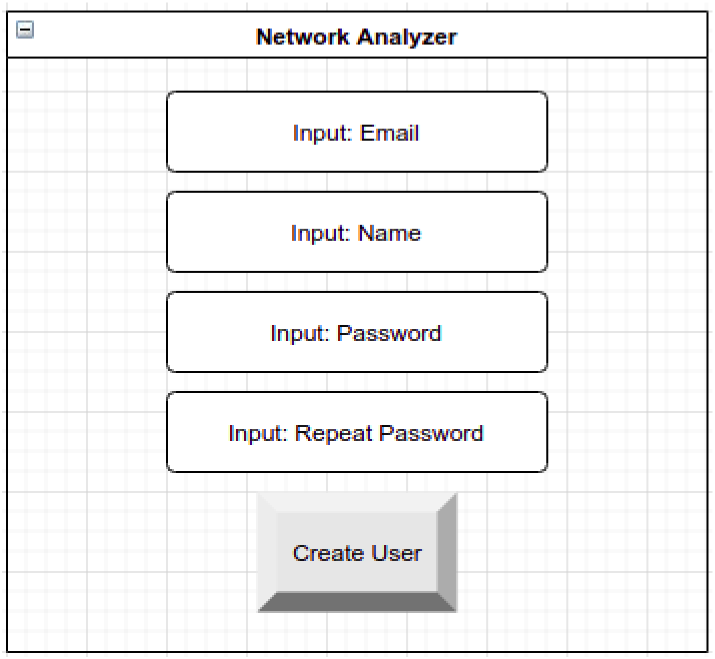
Below I will outline some brief explanations of each of the components that this application will have. Example screens will be added to have a better understanding what the application will be. There will be an extra "Notes" functionality that I believe is very needed, to keep records of ports or IP addresses without opening any other note applications.

Login/Register

Login component would have the main function to authorize users, giving privileges to a user to use all its functionalities if the user provided valid credentials to the application. Whereas the register side of the function will be responsible for creating users that have access to an application and all its functionalities. This will be required by an application when creating users to provide a unique email address, as it will be used as one of the identifiers within user pool. User credentials will be stored in a secure manner within a database, using strong currently unbroken hashing algorithms. Database configuration will be taken care of for a user using the underlying code base with all the requirements fulfilled beforehand.

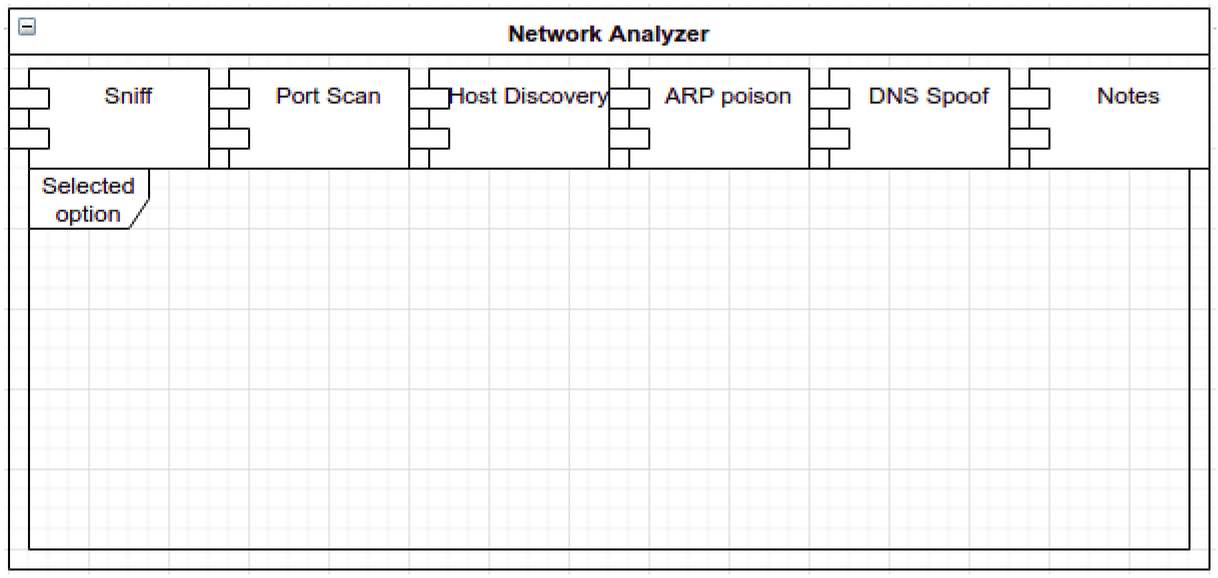
Below are some expected examples of screens of an applications login/register function:





Main Window

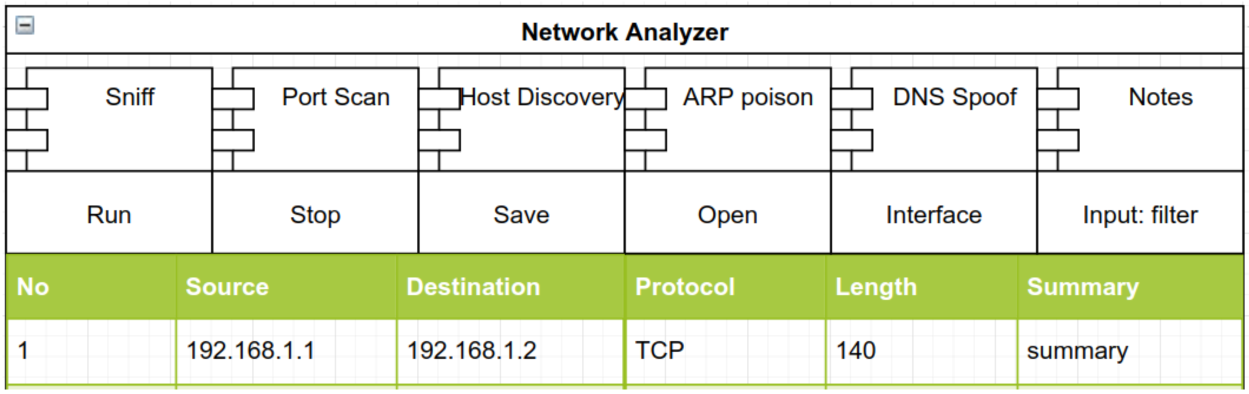
This window will be a wrapper for all the existing functions of an application. In the top bar of the window, a user will be able to choose what functionality it wants to use. Based on the plan, all components will retain data gathered within its separate frame, if the data will be needed for later reference.



Network sniffer

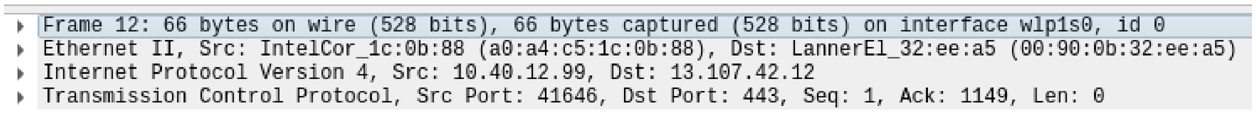
Network sniffer component is the main component of any current application. It will display most important fields related to network packets. It will be able to capture and distinguish between different protocols including ICMP, TCP and UDP. It will have the ability to take filters to give better output for a user.

The main top window will provide information related to packet in a manner of packet number, source IP, destination IP, protocol and summary of a packet. The network sniffer will be able to take parameters. It will provide choice of interfaces to use, and it will take parameters for filtering IP packets such as by protocol or by source/destination IP address. Also, the network sniffer will have implemented functionality to save network captures in a .pcap file format if needed for further analysis using different tools.



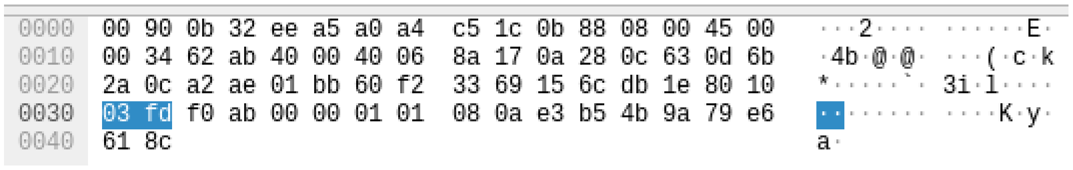
No	Source	Destination	Protocol	Length	Summary
1	192.168.1.1	192.168.1.2	TCP	140	summary

The middle frame of a network sniffer component will display more detailed information of each packet that is highlighted within the first window. This information will include OSI layers 2, 3 and 4. Thus, concluding information related to data link, network and transport. This functionality will be implemented looking at Wireshark.



```
▸ Frame 12: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlp1s0, id 0
▸ Ethernet II, Src: IntelCor_1c:0b:88 (a0:a4:c5:1c:0b:88), Dst: LannerEl_32:ee:a5 (00:90:0b:32:ee:a5)
▸ Internet Protocol Version 4, Src: 10.40.12.99, Dst: 13.107.42.12
▸ Transmission Control Protocol, Src Port: 41646, Dst Port: 443, Seq: 1, Ack: 1149, Len: 0
```

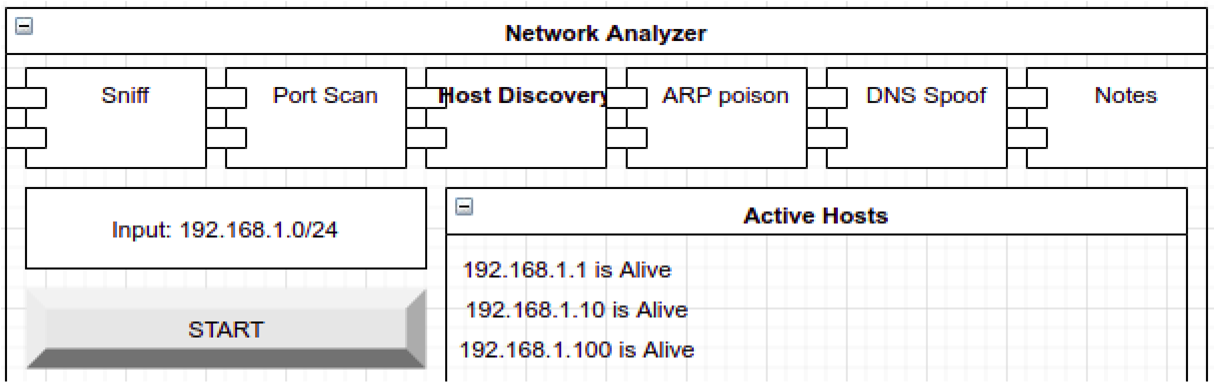
Bottom frame will display information in hexadecimal and in ASCII formats.



```
0000  00 90 0b 32 ee a5 a0 a4  c5 1c 0b 88 08 00 45 00   ···2···· ······E·
0010  00 34 62 ab 40 00 40 06  8a 17 0a 28 0c 63 0d 6b   ·4b·@·@· ···(·c·k
0020  2a 0c a2 ae 01 bb 60 f2  33 69 15 6c db 1e 80 10   *·····`· 3i·l····
0030  03 fd f0 ab 00 00 01 01  08 0a e3 b5 4b 9a 79 e6   ········ ····K·y·
0040  61 8c                                              a·
```

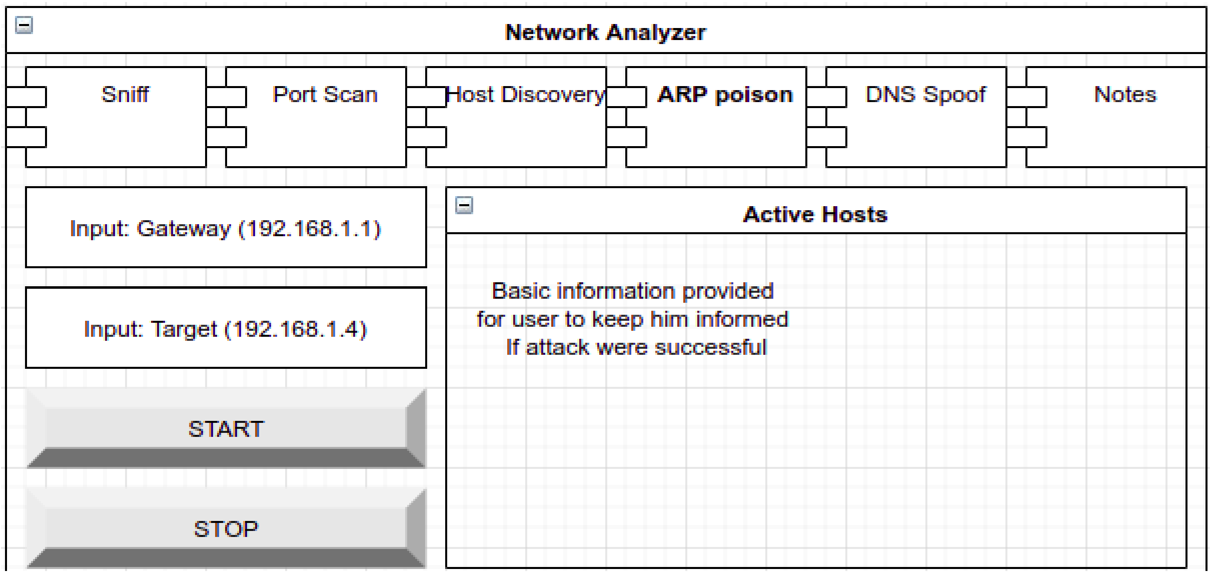
Host Discovery

This function will work by providing subnet by the user e.g., 192.168.1.0/24. After all subnets would be iterated through it will display all alive host devices on a network



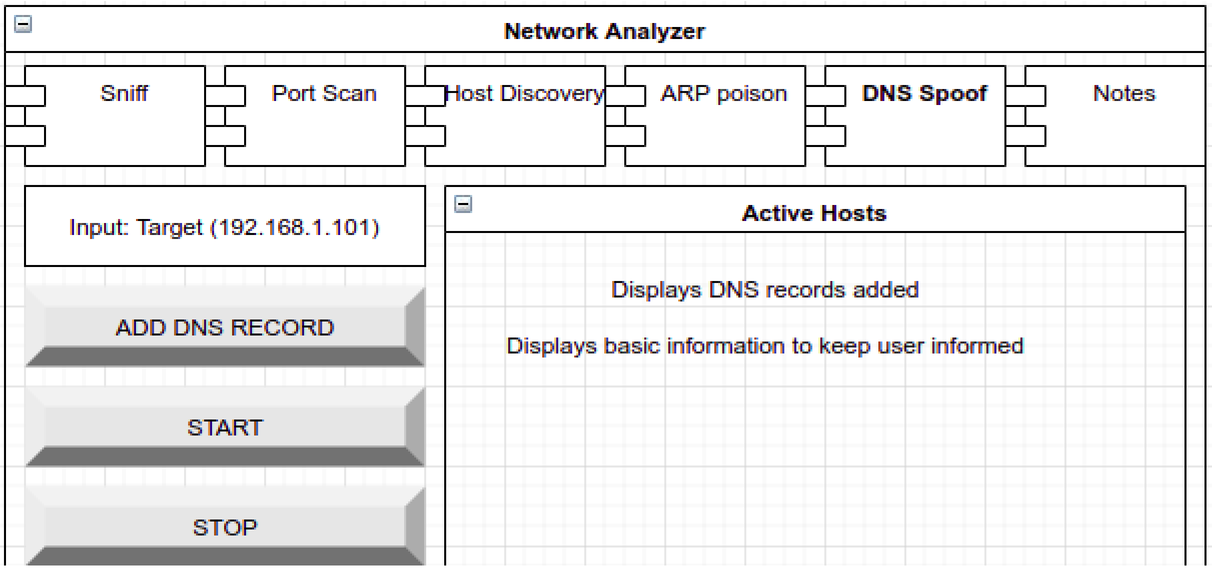
ARP poisoning

This function will require users to provide default gateway IP address together with targeted IP address. It will work by sending spoofed ARP requests with fake details. To make this functionality work, packet forwarding must be enabled within operating system commands and this will be provided within Git documentation.


Network Analyzer
Sniff
Port Scan
Host Discovery
ARP poison
DNS Spoof
Notes
Input: Gateway (192.168.1.1)
Input: Target (192.168.1.4)
START
STOP
Active Hosts
Basic information provided
for user to keep him informed
If attack were successful

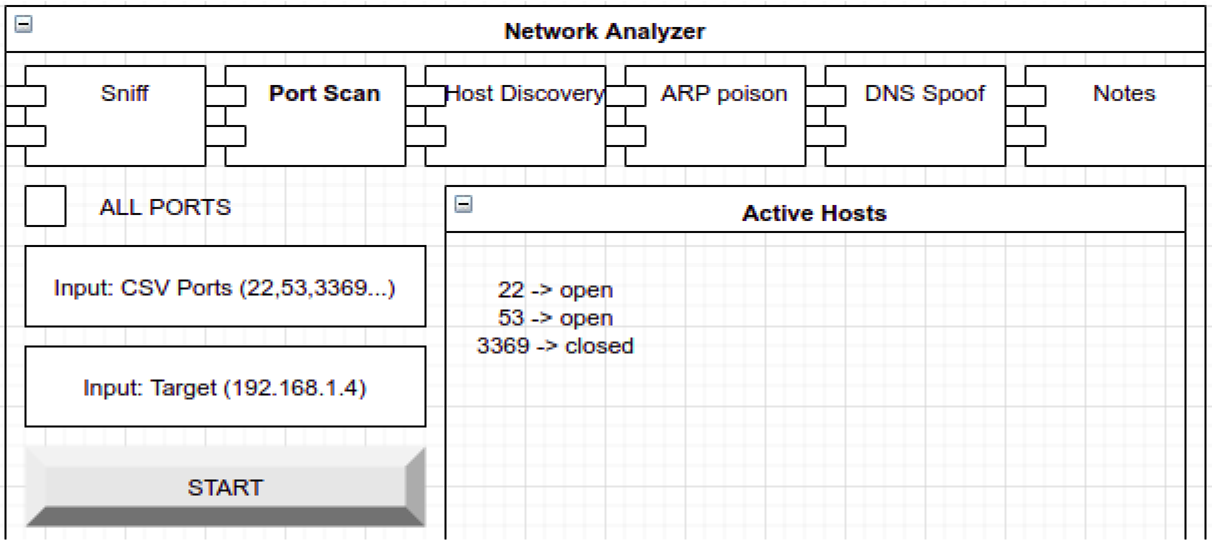
DNS spoofer

This functionality will work by taking from user targeted IP address. To make this functionality efficient a user must provide list of fake DNS records that will be fed to user's device.


Network Analyzer
Sniff
Port Scan
Host Discovery
ARP poison
DNS Spoof
Notes
Input: Target (192.168.1.101)
ADD DNS RECORD
START
STOP
Active Hosts
Displays DNS records added
Displays basic information to keep user informed

Port scanner

Port scanner functionality will work by taking inputs from the user. Inputs will be targeted IP address and ports listed or selected all.



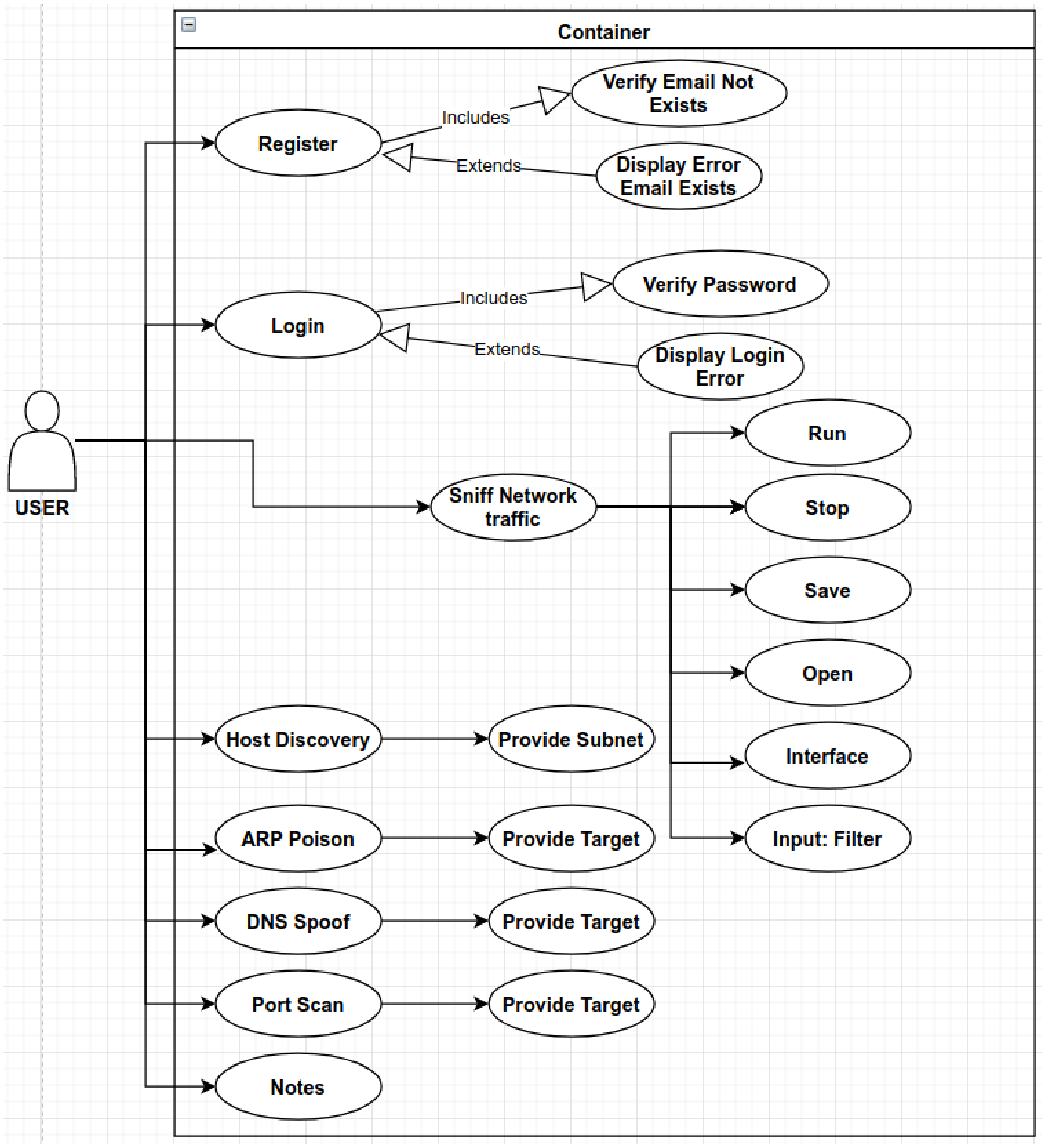
Use Case Diagram

A use case diagram shows the actor and the associations between the actor and use cases of the system. The user will be able to perform network sniffing, port scanning, host discovery, ARP poisoning and DNS spoofing.

Application Actor

This system will have multiple users, no administrator is needed. Once a user creates an account by providing an email address and password it is allowed to use this application. When the account is created or on successful login the user will be able to perform various actions listed in the application that contains network sniffing, port scanning, host discovery, ARP poisoning and DNS spoofing. Network sniffing will have subcategories that includes running sniffer, stopping sniffer, saving to .pcap file format, opening .pcap file, selecting interface and providing filters.

Diagram



Short Use Cases for a User

No.	Component	Precondition	Steps	Output
1.	Register	N/A	- User clicks on "Register" button - User provides email and password twice - User clicks register	User has an account created
2.	Login	Must have account	- User clicks on "Login" button - User provides valid credentials - User clicks "Login"	User enters an application
3.	Sniff	Must be logged in	- User clicks "Sniff" button	Sniffer screen presented
4.	Sniff/Run	Must be logged in	- User clicks "Run"	Sniffer captures traffic
5.	Sniff/Stop	Must be logged in	- User clicks "Stop"	Sniffer stops
6.	Sniff/Save	Must be logged in	- User clicks "Save" - User provides name for .pcap file - User clicks "Save"	.pcap file created
7.	Sniff/Open	Must be logged in	- User clicks "Open" - User chooses file - User clicks "Open" on chosen file	.pcap file imported
8.	Sniff/Interface	Must be logged in	- User clicks "Interface" - On dropdown chooses interface	Interface set
9.	Sniff/Filter	Must be logged in	- User inputs filter condition before running sniffer	Condition set
10.	Host discovery	Must be logged in	- User clicks on "Host discovery" button - User provides subnet - user clicks start	Alive hosts shown
11.	ARP Poison	Must be logged in	- User clicks "ARP Poison" button - User provides target - User clicks "Start"	Target has its ARP table poisoned
12.	DNS Spoof	Must be logged in	- User performs previous step first - User clicks "DNS Spoof" button - User provides same target as in previous step - User clicks "Start"	Target has its DNS spoofed
13.	Port scan	Must be logged in	- User clicks "Port scan" button - User provides target - User clicks "Start"	Opened ports on a target listed
14.	Notes	Must be logged in	- User clicks "Notes" Button - User enters note	Quick notes if needed