9/27/2024

Automated Network Tester

Research Document

(C00271395) Paul Loftus
SETU CARLOW

Abstract

Network Reconnaissance is a necessary stage in the life cycle of both securing networks and exploiting them. My Automated Network Tester aims to simplify the process for penetration testers and network administrators alike. This document delves into the background of cyber reconnaissance, where the term comes from, and common techniques used in Reconnaissance to gather information. I then discuss the languages I researched to use as a code base for my program along with the disadvantages and advantages of each. After discussing my languages, I expanded on the GUI libraries and frameworks within my chosen language and gave some examples on how the backend would look like.

All that was left was to discuss and compare the technologies assisting me in completing cyber reconnaissance along with an in-depth analysis of how I would make use, or not benefit from the use of, each technology or library discussed. I summarized all my points in my summary at the bottom of this document.

Table Of Contents

Introduction

Network automation has been growing increasingly within the past decade. With the establishment of languages like python and the increasing usage of the internet and network devices, traffic on networks has reached an all-time high with previous tasks getting harder and harder to manage manually. To support this, there are a myriad of new technologies, companies and solutions coming into play (Hicks, 2024).

My tool has the main purpose of Reconnaissance. Reconnaissance in a cyber context has always been manual until recent years. The original idea of cyber reconnaissance dates to early days of the computer when it was being used for analysis rather than cyberattacks (SentinelOne, 2023). Of course, the world kept going and the internet kept growing, Sophisticating the process of Reconnaissance into a completely new meaning (SentinelOne, 2023). Nowadays, reconnaissance consists of automated tools and social engineering techniques (SentinelOne, 2023). Many people would do specialized studies to become cyber reconnaissance experts as the techniques were complicated and diverse (Sumanth, 2023). Automated Reconnaissance tools did not start popping up until software and websites like Shodan.io or NMAP were released. After which people realized the power of automating the easy parts of reconnaissance while keeping the complexity of manual reconnaissance at its heart.

I have always had an interest in automation in the field of other expertise like networking, software testing and security tooling. This was only exacerbated by my time interning at Yahoos Product Security Team. I gained a large amount of experience with tools I had never used before in the company, which allowed me to think of my own thesis idea by combining things I had done recently with something I **wanted** to do, which was a reconnaissance type tool that would do enumeration scans and tests of sorts to try find possibly exploitable parts of a network and vulnerabilities across the network. This research document will dive into the structure I will go with for the program. I will also discuss the technologies I planned to use, with comparisons and contrasts between other choices. I will also give background information on everything needed to understand my development and the goal I wish this program to achieve. Some of the topics I will be going over are as follows, The Programming Language, the approach I will take and the tools and techniques I will incorporate into the program to perform the reconnaissance for which I am aiming.

Background

What is Reconnaissance?

Cyber Reconnaissance is a concept dating back to the very start of computing networks where it was employed for purposes unrelated to hacking or penetration testing (SentinelOne, 2023). At first it was a major help in network diagnostics and management until the evolution of networks and the internet gave threat actors the idea to use it maliciously to scope out and map networks for a pre-emptive attack (SentinelOne, 2023).

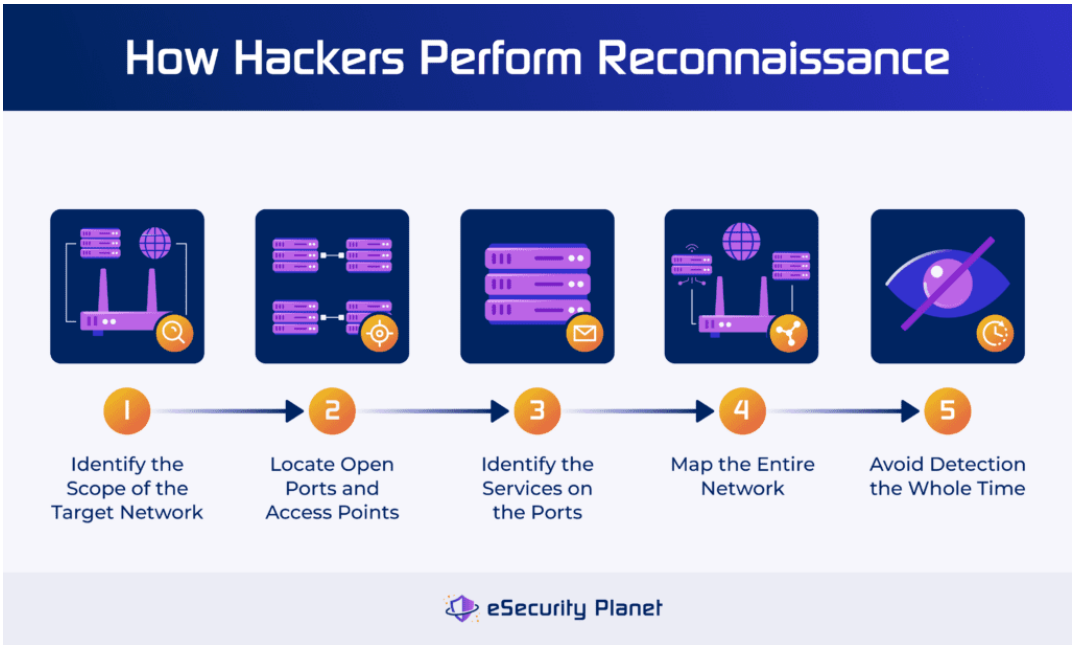

Figure 1 - Reconnaissance Steps (Phipps, 2024)

Reconnaissance is the preliminary phase of cyber-attack, simulated or authentic, as defined by Imperva (Imperva, N/A). The term originates from military operations, although in the cybersecurity industry, it carries a different meaning. Reconnaissance is gathering data about a target's systems, services, and applications, to find vulnerabilities to exploit in a subsequent attack. (Imperva, N/A). Useful information in terms of reconnaissance can be any data, for example, target architecture, software versions, security measures, services currently running, user information and more (Blumira, 2024).There is a structure to reconnaissance. Threat actors will first determine the scope of the network and its scale. After discovering the scope, the next step will be to locate open ports by using tools like NMAP which allows you to probe all ports for responses to see if they are closed, open or filtered through complex scans with specific settings. Information on services running on a network will also be searched for to gain as much knowledge on the network as possible (Imperva, N/A). After gathering all the information they can, threat actors would finally map the network out, seeing how each service and port communicate with each other to get a better understanding of the network infrastructure they wish to infiltrate (Blumira, 2024) .There are 2 main differentials when it comes to Reconnaissance, Passive and Active.

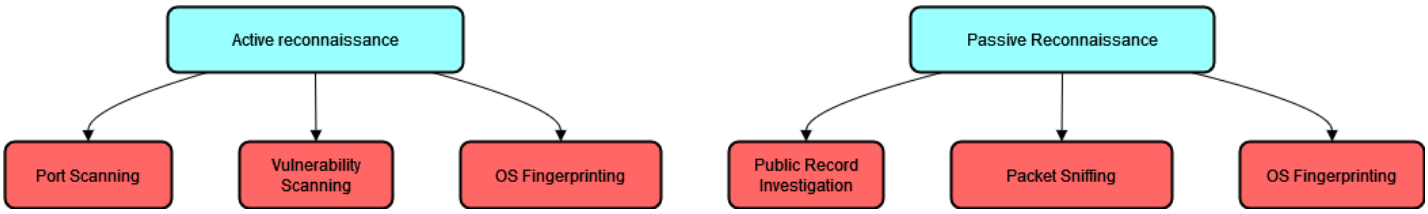

Figure 2 - Types of Reconnaissance

Passive

Passive Reconnaissance involves being more of a bystander in terms of gathering the information you wish to use against a system. It is much stealthier and leaves less dirt on your footprint. The most common techniques in Passive Reconnaissance are as follows: network traffic analysis, using packet sniffers like Wireshark (GUI) or sniffr (CLI). Monitoring public channels of communication like social media or investigating public files and employees. For my project specifically, I wished for the user to be able to choose an approach. With the technologies listed below, I would like to implement passive Nmap scans for host discovery and enumeration. In addition to this, I would make use of the scapy module, also discussed below, to attempt to capture packet data sent to open ports found through the passive nmap scans.

Active

Active Reconnaissance can be much more effective and fast with the downside of being extremely detectable and alerting (Imperva, N/A). Usually, it consists of a direct involvement with the target's systems meaning there is a much larger chance of logs of your attempts being left (Imperva, N/A). Common Active Reconnaissance tools include port scanners, network scanners and vulnerability scanning. These tools can give you a myriad of information to use to propagate an attack, either through DNS poisoning, DHCP Spoofing, MiTM Attacks, and other techniques. the most secure companies and systems would have many of their services, servers and technologies hidden on the internet so that information gathering attempts would not work against them.

Common Reconnaissance Techniques

Port Scanning

Ports are ever present in this age of technology. No device has none unless they have been purposefully left out of the design or removed. This means there is always a need to secure your ports for important devices.

Port scanning is a popular technique that involves surveilling computer ports on either common port numbers or large amounts to find open and vulnerable ports (Blumira, 2024). There are a total of 65,535 ports in 1 IP Address (Blumira, 2024). This leaves 65,535 ports that could become an attack vector into your network.

Many tools exist to perform port scanning on networks, one of these being Nmap which I discuss below in relation to my program (Shivanandhan, 2020). These tools can be very powerful within a threat actors toolbelt. Port scanning can be done in two ways, silently/Stealthily or Aggressively (Blumira, 2024). Usually, port scanning is performed on port numbers after the 1024 mark as anything before this are standard service ports and may have already been filtered or closed (Blumira, 2024).

To avoid having to scan every single port unless specified, my tool was to scan the most popular ports as a start and then attempt other methods if unsuccessful. Some common ports I found through my research are as follows:

- Port 80 (HTTP) – One of the most common ports on the internet, used for access to web server resources. (unsecure)
- Port 23 (Telnet) - Telnet, a predecessor of SSH, used for remote connections(unsecure)
- Port 443 (HTTPS) - SSL-encrypted web servers use this port by default. (secure)
- Port 21 (FTP) – File Transfer Protocol
- Port 22 (SSH)—Secure Shell, an encrypted replacement for Telnet (and, in some cases, FTP).
- Port 25 (SMTP)—Simple Mail Transfer Protocol (also insecure).
- Port 3389 (ms-term-server)—Microsoft Terminal Services administration port.
- Port 110 (POP3)—Post Office Protocol version 3 for email retrieval (insecure).

- Port 445 (Microsoft-DS)– commonly used for file or printer sharing
- Port 139 (NetBIOS-SSN)–NetBIOS Session Service for communication with MS Windows services (such as file/printer sharing).
- Port 143 (IMAP)–Internet Message Access Protocol version 2. An insecure email retrieval protocol.
- Port 53 (Domain)–Domain Name System (DNS), an insecure system for conversion between host/domain names and IP addresses.
- Port 135 (MSRPC)–Another common port for MS Windows services.
- Port 3306 (MySQL)–For communication with MySQL databases.
- Port 8080 (HTTP-Proxy)–Commonly used for HTTP proxies or as an alternate port for normal web servers
- Port 1723 (PPTP)–Point-to-point tunnelling protocol (a method of implementing VPNs which is often required for broadband connections to ISPs).
- Port 111 (RPCBind)–Maps SunRPC program numbers to their current TCP or UDP port numbers.
- Port 995 (POP3S)–POP3 with SSL added for security.
- Port 993 (IMAPS)–IMAPv2 with SSL added for security.
- Port 5900 (VNC)–A graphical desktop sharing system (insecure).

Data Aggregation

Data Aggregation is, as my source states, ‘the process of gathering and consolidating diverse sets of information and resources from disparate sources into a comprehensive framework’ (LarkSuite Editorial Team, 2024). Now in the context of cybersecurity, this has proved very useful in terms of incident response rates and dealing with cybersecurity related attacks or issues (LarkSuite Editorial Team, 2024). Normally, sources will consist of network logs, IDS/IPS Intrusion logs, System event logs, application-specific logs and more (LarkSuite Editorial Team, 2024). Data aggregation is normally used to detect cyber incidents but for this research document, I am going to focus more on data aggregation on the side of threat actors.

Malicious Data Aggregation is slightly different, as the intention behind the data collection is to use against a certain individual, company, or asset (VPNUnlimited, 2024). A threat actor would access online sources, possibly on social media or through public databases,

to gather personal information like your age, name, what your face looks like, your family members, etc. to ultimately use against you (VPNUnlimited, 2024). This can come in a variety of forms from brute forcing your passwords to personalized phishing messages designed to catch your eye.

Operating System Fingerprinting

O.S Fingerprinting involves analyzing data packets from a network, to find intelligence that could be used in a malicious attack (Firewalls, 2024). Information about OS specific configurations can determine what Operating System a packet originated from, which in turn makes it easier for threat actors to target known vulnerabilities. For networks containing outdated network and end devices, and even remote devices, this reconnaissance method can very quickly become a weakness in a network's defenses. The only downside of this is the time it takes to gather valuable information although with the fact that only 55% of companies actually run cybersecurity assessments on their networks and infrastructure, there is a larger chance of finding a vulnerability through outdated security policies and versions (TerraNovaSecurity, 2024)

Operating System Fingerprinting can be executed passively or actively. Passively, it involves analyzing data packets being sent between devices on a network (Firewalls, 2024). Actively, O.S Fingerprinting transform into sending packets to devices on a network and analyzing the TCP packet contents for valuable data. The difference between the two execution methods is:

- Your presence and likelihood of being detected.
- The execution time

The chosen method of O.S Fingerprinting should depend on your situation and what you are trying to achieve, the time limit you have, and how exposed you want yourself to be on a foreign network.

The Breakdown

Programming Language

Python

My first choice for this project was Python. As a language, Python is an Interpreter-based, high level programming language with Object Orientated Programming, polymorphism, dynamic types, and easy setup (Python Software Foundation, N/A). Python was created in 1991 by a programmer named Guido Van Rossum, who was looking to create a language that was more human readable (Python Software Foundation, N/A). I had a high ability in it already which made the logic of the program much easier to imagine. I also had complete projects with similar specifications in my time at Yahoo doing my work placement which gave me confidence on if I (University Of Michigan, 2024)would be able to complete this project in the necessary time constraints. Python also had features that appealed to me in this project, like its high library count for things like data manipulation (GeeksForGeeks, 2024) and its vast array of networking Libraries (JavatPoint, 2024). The fact it is more human readable is also a bonus for both my academic supervisors and me.

Figure 3 - Python Logo (Python Software Foundation, N/A)

The only problems I had with python was that in terms of speed, it was lackluster in comparison to my other possible choice and it was also not as accessible with other tools due to it being a high level language, which meant I would have to either find libraries or workarounds to use the tools I wanted to use (Python Software Foundation, N/A). It also has dynamic types which can be a double-

edged sword as exceptions can arise from unexpected data types (GeeksForGeeks, 2024). Garbage collection is also a downside, as it is implemented but not natively, meaning you would manually have to set up garbage collection (Builtin, 2024). Python is also known as a universally portable language and doesn't have many problems with running on different systems and distributions unless you are using an OS-specific library which would of course fail on another system. (GeeksForGeeks, 2024)

Bash

Bash is a good and bad contender for this program. It is fast and used in automation all over the world which seems very fitting considering the project title (Hoffman, 2016). I had done 5 months Bash scripting for my internship meaning I did not have to go in blind to begin coding my project and I had already been learning more about Bash commands throughout the year. The problem with Bash was that it was not built to be a fully-fledged programming language. Bash is designed to handle text processing, file operations and scripting. (IuvoTech, 2024). The syntax is very awkward and strict, and errors are only discovered at run-time. It is nice that is available on all Linux systems, but portability is an issue as attempting to run bash scripts on windows does not fare well unless using the new Windows Subsystem for Linux (WSL). (Hoffman, 2016)

Figure 4 - Bash Logo (Akulov, 2001)

Golang

Go is a language developed by Google Engineers with system level programming, infrastructure, and networking in mind (Boyd, 2024). It has a simple syntax with similarities to C/C++. I had the chance to see a matured, Enterprise-level project written in Go and thus I thought it might be a choice for my project. The compiler does not

require a VM and compiles into a Binary File, making execution easy and portable on most distributions of Operating Systems (Barney, 2023). The language has independent error handling, automatic garbage collection and concurrency which is vital in a program like this which can be running many scans each iteration. (Barney, 2023) The limitations with this language is that the binaries you produce can be extremely large in some cases as they directly compile into Executable files. (Barney, 2023) Another issue arises when you put RAM usage into consideration as this language can devour your RAM when running multiple instances/files. This could mean disaster for machines attempting to run the tool with lower ram with the memory being filled to the brim after running the binary. (Barney, 2023). Go also has limited GUI Programming support (Esenyi, 2023).

Figure 5 - Golang Logo (Google, 2024)

Summary

After hard consideration, I decided to go with the Python approach. There were many deciding factors in this decision. Firstly, Python, in comparison to Go or Bash, had the easiest to read Syntax along with the status as an open source project which meant there was no proprietary worries when I would be publishing this tool to GitHub. (GeeksForGeeks, 2024)

Secondly, there was GUI Programming support, which was impossible on Bash as it was a CLI only Tool. Golang could have a GUI, but I had more experience building GUIs on Python rather than Golang which I had brief experience with (Esenyi, 2023). Thirdly, Python has a massive standard library along with over 137,000 installable libraries available through the Python package Index meaning that anything I want to do, is most likely possible using one of these libraries (University Of Michigan, 2024) unlike Bash which was mainly built for

scripting and text/file processing (Hoffman, 2016). Another factor I took into consideration was the ram usage my program would have from Go. I had researched and found a company that had shown how much their CLI tool was using the ram on their machine, below is a figure to show an example (Gritter, 2021)

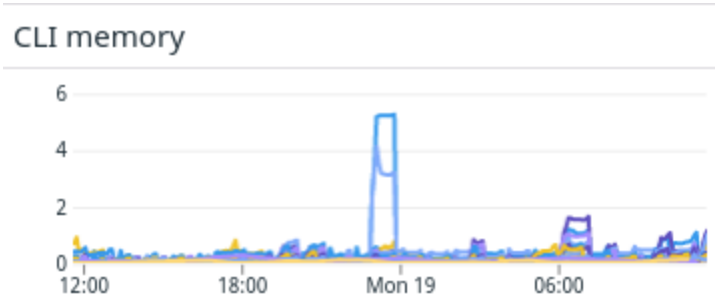

Figure 6 - GO RAM Usage (GBs) (Gritter, 2021)

As you can see from the above figure, the memory usage of their tool would reach up to 5 Gigabytes of memory usage (Gritter, 2021). This was an extreme amount, no matter how large their program is. In the article, the author was able to reduce Go's RAM Usage to a stable amount, but this was after having to alter the internal Go Garbage Collector and add limits to how much memory the program can use while being careful to not limit more than he should. Thus, crashing the program (Gritter, 2021). The author had to essentially trim at every part of the program to reduce the memory usage (Gritter, 2021).Personally, I saw this as a lot of extra work when I could avoid this problem entirely by not using Go.

GUI Framework

What is a GUI Framework?

A Graphical User Interface (GUI) is defined as a digital interface in which users interact with graphical components like buttons, menus and icons to operate a program, as opposed to a CLI (Juviler, 2022). For my program, I opted for a GUI rather than a Command Line Interface (CLI) tool to make using the program easier for other users. CLI tools are notoriously harder to use so a GUI made sense to implement. I had used some frameworks in the past for my personal projects, but I did

not have enough experience to not research so below I list some of the possible options. Most of the libraries I researched had similar syntax and structure which means that migrating to a different library would also be simple enough (Fitzpatrick, 2024).

Tkinter

Tkinter is the most commonly used python library for building GUIs, so much so that it has been integrated into the python standard library and comes bundled with python's default installation (Fitzpatrick, 2024). The library is a pure GUI library, not a framework, unlike other similar libraries I list below (Fitzpatrick, 2024). Tkinter lacks built-in support for some functions like displaying multi-media, interfacing with data sources and databases, and the components can look outdated on windows OS (Fitzpatrick, 2024). The advantage with Tkinter is that it needs no additional dependencies.

Tkinter seems like a good option as it is already built into the python library meaning I will not need to increase the size of my project any more than I already am with the many tools I plan to use. In a way this would be efficient to use Tkinter but if I wish to make a more complex GUI then I know that is where Tkinter falls. I also saw how some components looked on Windows OS as mentioned, and it seemed to me that the components did look outdated. The other factor that comes into consideration is that Tkinter is cross-platform making it easier to keep my program universally accessible to both Windows and Linux Distributions.

PyQt/PySide

PyQt and PySide are both wrappers built around the Qt Framework (Fitzpatrick, 2024). The Qt framework is a cross-platform GUI framework built in C++ (Kubara, 2023). With a large tech community behind it and many online resources to develop, PyQt is favored for modern commercial application development (Fitzpatrick, 2024). Qt can run on most common and even uncommon Operating Systems, e.g., Windows, Linux, QNX, Android and IOS (Kubara, 2023).

With the Qt framework, there are many added pieces of functionality. They come in the form of addons and include the following. Interfacing with SQL Databases, playing multi-media files like mp4, templated/quick layouts, and even network layer assistance with server communication and implementing network protocols (Kubara, 2023). The benefits of Qt can be easily seen, but it does come with disadvantages. One to note is the complexity of the framework as with all tools online, the more you can do with a tool, the more you need to learn (Fitzpatrick, 2024). For my project I was unsure if this framework would work in my favor by enhancing the GUI, and if the time required to build it would be worth it. As I did not have basic GUI library experience, this library seemed daunting to use. This was the deciding factor behind my decision not to use this library. Below is an example of a simple GUI built with the PyQt Library to show you how the program is structured (Fitzpatrick, 2024).

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Hello World")

        button = QPushButton("My simple app.")
        button.pressed.connect(self.close)

        self.setCentralWidget(button)
        self.show()


app = QApplication(sys.argv)
w = MainWindow()
app.exec()
```

Figure 7 - Example of PyQt Code (Fitzpatrick, 2024)

WxPython

WxPython is a wrapper for a GUI toolkit named WxWidgets (Fitzpatrick, 2024). WxWidgets is a cross-platform toolkit written in C++. Both WxWidgets and WxPython are open source and free to use. My goal is to only use open-source tools which this library supports towards that end. Like the Qt Framework, this library uses native widgets and components to create GUIs, decreasing the size of the codebase behind the application built with it (Fitzpatrick, 2024). One of the disadvantages of using WxPython I found through my research, was the

fact it has platform-specific ‘Quirks’ or bugs (Fitzpatrick, 2024). This unfortunately means that maintaining cross-platform compatibility would be harder as I build the program. I wanted to ensure that cross-platform compatibility would always be a priority of the program, so I decided that perhaps WxPython was not the best fit for my project.

Tools

Cython

Cython is a static compiler for the python language that helps its performance issues but interlacing it with the C language. (Cython, 2007). It will allow me to add static types to variables, call functions from optimized low-level libraries in C, and interact with large datasets in an efficient way (Cython, 2007). I wanted to try make my program as efficient as possible so that I could do much more in a single iteration without compromising the performance of the tool. I am still on the fence on whether I will be required to use Cython depending on the natural runtime of my program, but it is good to have a backup plan.

Nmap

Network Mapper (Nmap) is an open-source tool developed by Gordon Lyon which would allow users of the tool to scan networks for open ports, accessible services and map out a network (Shivanandhan, 2020). You are given access to a very efficient program that allows you to build both simple and complex commands using their own scripting engine. This tool is a staple in the security industry. This tool would be vital in my program’s backend as this will allow me to get a large dataset to begin investigating a network. I would need to run multiple queries for my program to test open ports and services, which thankfully Nmap is very flexible and portable allowing me to discard my worry of a cross-platform bug.

Subprocess

For me to interface with Nmap through Python, I could choose two approaches. The first one I thought of was one that I had implemented before in a previous project, the Subprocess Module (Geeks4Geeks,

2024), The purpose of this module is to spawn child processes with shell commands running (Geeks4Geeks, 2024). I had previously used this method to interface with other CLI Tools with successful results. To properly use this method, first you would need to create methods to spawn the processes using `subprocess.check_output` (Geeks4Geeks, 2024). This returns the output of whatever command you placed inside the function. Then you would need to parse the output from the command line into a format you can access and re purpose the data with.
This is very possible, but I wanted to save this module for the other CLI tools that have no API to interface with Python, as I would have to spend extra time on building the commands and parser for each module.

Python3-Nmap

The second and final option I went with to interact with Nmap was a library I found on the Pypi index (Wangolo, 2024). The package comes filled with methods that would prove very useful to me. I planned on using the TCP/IP Fingerprinting command
`nmap_os_detection("192.168.178.2")`
To retrieve any data on the Operating Systems being ran. there is also the ping scan method along with different scanning techniques built in so that I can try multiple different ways to scope out the network my program runs on. Other commands include:

- Version Scanning
- Idle scanning
- Fin scan
- Only port scan (-Pn)
- Only host discover (-sn)
- Arp discovery on a local network (-PR)
- Disable DNS resolution (-n)
- Dns brute script scan (subdomain enumeration)

With these commands, it seems like a large enough codebase for me to use to interface with Nmap. I could also attempt to implement a subprocess call to make up for the methods that were not implemented yet into the library.

Scapy

Scapy is a powerful packet manipulation library built in 2008 and maintained since then (Scapy, 2008). It has such an extensive set of classes and functions that it is now built into the standard library

that comes with python. When I say packet manipulation, what I mean is that Scapy can decode, manufacture, capture, and match packets within the python language (Scapy, 2008). My goal with this library is the utilization of capturing packets if I can make a connection to any ports or sockets on the target network. Scapy has extensive documentation which gives me a lot of wiggle room as I do not have to worry about running into undocumented errors or problems.

Requests

Requests is a simple HTTP Library built into python's standard library. It can send requests, receive responses, and alter parameters, headers, and SSL Certificates (Ronquillo, 2024). I have experience using requests from some web scraping projects I did which made this library come to mind when I was researching. With this library, I hope to test web servers connected to networks to see if they can be accessed by sending test HTTP(S) requests to active web servers on a network and analyzing the response sent back.

SubBrute

When doing my research, I wanted to find a tool that could do DNS enumeration in the case that if the program is executed on an internet facing domain, it can scan the DNS server for possible entry points. This is when I found SubBrute, a DNS-Spider Subdomain enumeration tool built in Python (TheRook, 2024). Support has been added to use it within python code instead of only being a standalone tool (TheRook, 2024). Using SubBrute, I hope to investigate domain servers and their connections within a network, mapping the network to the best of the module's ability.

Knock

Knock is another Subdomain enumeration tool. The difference between SubBrute and Knock is that Knock is a passive reconnaissance tool that uses dictionary attacks, while SubBrute is a DNS Spider tool with a much more aggressive approach. There is a flag to opt for brute force scanning within the tool, but the default is passive. This tool also allows you to control many parts of the scan including custom DNS, User agent, and threads. I will attempt to use both knock and SubBrute in my program and after testing, will decide if running two enumeration tools yields better results than just the one, or if the performance difference will invalidate such claims. Knock would also

be a good option for the passive reconnaissance side of things as I wish to implement an option between the two.

NetMiko

NetMiko is a pure python SSHv2 Implementation available for python 3.6+ (PyNetLabs, 2024). The library has gained mass popularity with network administrators and automation experts (PyNetLabs, 2024). NetMiko supports a large selection of network device brands including the following and more (Samoylenko, 2024):

- Arista vEOS
- Cisco ASA
- Cisco IOS
- Cisco IOS-XR
- Cisco SG300
- HP Comware7
- HP ProCurve
- Juniper Junos
- Linux

Along with multiple-device support, Netmiko also supports telnet connections, albeit restricted to cisco devices (Samoylenko, 2024). Netmiko is built upon the paramiko library meaning it has extended functionality and better support overall (PyNetLabs, 2024). My goal with NetMiko is to attempt testing vulnerable SSH/Telnet ports when my program can find open ssh ports. I would be using the library for its ability to connect to network devices rather than to change configuration or send commands. I have no need to interact with the device OS but just to see if it is possible to make an unauthenticated connection.

Conclusion

As I have researched into Reconnaissance and its importance in the world of cyber security, I have gained a deeper understanding of how to build a tool that will not only simplify but be a proper choice for the starting steps of scoping a network and its devices. My analysis of common techniques used in reconnaissance have given me ideas on how to approach testing networks on both the internet and internally, along with the information usually gathered in such steps. After comparing programming languages that I thought would suit my purpose,

I have also come to find that Python in its simplicity and extensive functionality would be the best option for me as a programmer, both personally and logically, while other languages fell in rank with their different disadvantages. Like bash's scripting functionality, among other things, which would have decreased my tools ease of use and the level of programming I would need to put in place.

My research into the technologies for my programs GUI I found online also revealed to me that certain GUI frameworks in python are for very different use cases, like Tkinter's easy design and setup versus PyQt/PySides framework which gives expanded functionality in the form of addons. They can also have their disadvantages as I found with WxPython's "quirks" which are platform specific leading me to believe that I might have trouble maintaining cross-platform compatibility.

I have also done extensive research into the tools that I May or may not need to use to reach the full potential of my program. I have found some industry standard libraries and tools like Nmap, Subprocess and Scapy with would allow me to interact with network layer protocols and devices through Python. I have also found some smaller, uncommon tools that could boost my tools productivity like Knock or SubBrute for subdomain enumeration or NetMiko for testing connections to accessible network devices. I also contrasted the issues I had found with these tools and whether they validated my reason for using them versus the drawbacks they would give me in my implementation section of this project.

Figure Table

Works Cited

Akulov, D., 2001. *Bash Logo*. [Online]
Available at: https://bashlogo.com/
[Accessed 10 11 2024].

Barney, N., 2023. *What Is Golang?*. [Online]
Available at: https://www.techtarget.com/searchitoperations/definition/Go-programming-language
[Accessed 10 13 2024].

Blumira, 2024. *Port Scanning*. [Online]
Available at: https://www.blumira.com/glossary/port-scanning
[Accessed 29 10 2024].

Blumira, 2024. *Understanding Reconnaissance Techniques*. [Online]
Available at: https://www.blumira.com/glossary/reconnaissance
[Accessed 20 10 2024].

Boyd, W., 2024. *What is Go, An Intro into googles Go Programming Language A.K.A Golang*. [Online]
Available at: https://www.pluralsight.com/resources/blog/cloud/what-is-go-an-intro-to-googles-go-programming-language-aka-golang
[Accessed 13 10 2024].

Builtin, 2024. *Garbage collection in Python*. [Online]
Available at: https://builtin.com/articles/garbage-collection-in-python
[Accessed 13 10 2024].

Cython, 2007. *Cython*. [Online]
Available at: https://cython.org/
[Accessed 13 10 2024].

Esenyi, S., 2023. *Best GUI Frameworks for Go*. [Online]
Available at: https://blog.logrocket.com/best-gui-frameworks-go/
[Accessed 12 10 2024].

Firewalls, 2024. *OS Fingerprinting*. [Online]
Available at: https://www.firewalls.com/blog/security-terms/os-fingerprinting/
[Accessed 4 11 2024].

FITA Academy, 2024. *ultimate guide to tools and techniques of reconnaissance in ethical hacking*. [Online]
Available at: https://www.fita.in/ultimate-guide-to-tools-and-techniques-of-reconnaissance-in-ethical-hacking/
[Accessed 1 11 2024].

Fitzpatrick, M., 2024. *Which Python GUI Library should you use?*. [Online]
Available at: https://www.pythonguis.com/faq/which-python-gui-library/?gad_source=1&gclid=Cj0KCQjwsoe5BhDiARIsAOXVoUtZWARH9HAOBjqKYaQoNna-kCplL2F_ffNSSldf2pUlkN59KJ-o0mUaArNjEALw_wcB
[Accessed 23 10 2024].

Geeks4Geeks, 2024. *Subprocess Module*. [Online]
Available at: https://www.geeksforgeeks.org/python-subprocess-module/
[Accessed 07 11 2024].

GeeksForGeeks, 2024. *Libraries in Python*. [Online]
Available at: https://www.geeksforgeeks.org/libraries-in-python/
[Accessed 13 10 2024].

GeeksForGeeks, 2024. *Python Features*. [Online]
Available at: https://www.geeksforgeeks.org/python-features/
[Accessed 13 10 2024].

Google, 2024. *Directory blog/go-brand/Go-Logo/PNG*. [Online]
Available at: https://go.dev/blog/go-brand/Go-Logo/PNG/
[Accessed 10 11 2024].

Gritter, M., 2021. *Taming Go's Memory Usage, or How We Avoided Rewriting Our Client IN Rust*. [Online]
Available at: https://www.akitasoftware.com/blog-posts/taming-gos-memory-usage-or-how-we-avoided-rewriting-our-client-in-rust
[Accessed 07 11 2024].

Hicks, M., 2024. *The Path to Network Automation is now clearer and more defined.*. [Online]
Available at: https://itbrief.com.au/story/the-path-to-network-automation-is-now-clearer-and-more-defined
[Accessed 26 10 2024].

Hoffman, C., 2016. *How to create and run bash shell scripts on windows 10*. [Online]

Available at: https://www.howtogeek.com/261591/how-to-create-and-run-bash-shell-scripts-on-windows-10/
[Accessed 13 10 2024].

Imperva, N/A. *What is Reconnaissance*. [Online]
Available at: https://www.imperva.com/learn/data-security/cybersecurity-reconnaissance/
[Accessed 8 10 2024].

Imperva, N/A. *What is Reconnaissance in CyberSecurity*. [Online]
Available at: https://www.imperva.com/learn/data-security/cybersecurity-reconnaissance/
[Accessed 9 10 2024].

IuvoTech, 2024. *Bash VS Python. which scripting language is right for your needs*. [Online]
Available at: https://blogs.iuvotech.com/bash-vs.-python-which-scripting-language-is-right-for-your-it-needs
[Accessed 13 10 2024].

JavatPoint, 2024. *Top Python Libraries for Network Engineering*. [Online]
Available at: https://www.javatpoint.com/top-python-for-network-engineering-libraries
[Accessed 10 12 2024].

Juviler, J., 2022. *What is a GUI*. [Online]
Available at: https://blog.hubspot.com/website/what-is-gui
[Accessed 21 10 2024].

Kubara, I., 2023. *What is the Qt Framework and why should you use it?*. [Online]
Available at: https://lembergsolutions.com/blog/why-use-qt-framework
[Accessed 01 11 2024].

LarkSuite Editorial Team, 2024. *Data Aggregation*. [Online]
Available at: https://www.larksuite.com/en us/topics/cybersecurity-glossary/data-aggregation
[Accessed 05 11 2024].

Phipps, J., 2024. *How Hackers Use Reconnaissance*. [Online]
Available at: https://www.esecurityplanet.com/threats/how-hackers-use-

reconnaissance/
[Accessed 31 10 2024].

PyNetLabs, 2024. *What is NetMiko*. [Online]
Available at: https://www.pynetlabs.com/what-is-netmiko-and-how-to-use-it-in-python/
[Accessed 25 10 2024].

Python Software Foundation, N/A. *What is Python? Executive Summary*. [Online]
Available at: https://www.python.org/doc/essays/blurb/
[Accessed 13 10 2024].

Ronquillo, A., 2024. *Requests in Python*. [Online]
Available at: https://realpython.com/python-requests/
[Accessed 17 10 2024].

Samoylenko, N., 2024. *Python for network Engineers*. [Online]
Available at: https://pyneng.readthedocs.io/en/latest/book/18_ssh_telnet/netmiko.html
[Accessed 25 10 2024].

Scapy, 2008. *Scapy*. [Online]
Available at: https://scapy.net/
[Accessed 18 10 2024].

SentinelOne, 2023. *What Is Cyber Reconnaissance*. [Online]
Available at: https://www.sentinelone.com/cybersecurity-101/threat-intelligence/what-is-cyber-reconnaissance/
[Accessed 06 11 2024].

Shivanandhan, M., 2020. *What is NMAP and how to use it*. [Online]
Available at: https://www.freecodecamp.org/news/what-is-nmap-and-how-to-use-it-a-tutorial-for-the-greatest-scanning-tool-of-all-time/
[Accessed 13 10 2024].

Sumanth, G., 2023. *Manual Reconnaissance VS Automated Reconnaissance.*. [Online]
Available at: https://www.appsecengineer.com/blog/manual-vs-automated-reconnaissance
[Accessed 26 10 2024].

TerraNovaSecurity, 2024. *130 Cybersecurity Statistics*. [Online]
Available at: https://www.terranovasecurity.com/blog/cyber-security-statistics
[Accessed 10 11 2024].

TheRook, 2024. *Sub Brute, a Subdomain Enumeration Tool*. [Online]
Available at: https://github.com/TheRook/subbrute
[Accessed 21 10 2024].

University Of Michigan, 2024. *Installing Libraries And Packages*. [Online]
Available at: https://docs.support.arc.umich.edu/python/pkgs_envs/
[Accessed 14 10 2024].

VPNUnlimited, 2024. *Aggregation Attack*. [Online]
Available at: https://www.vpnunlimited.com/help/cybersecurity/aggregation-attack
[Accessed 4 11 2024].

Wangolo, J., 2024. *Python3-nmap*. [Online]
Available at: https://pypi.org/project/python3-nmap/
[Accessed 7 11 2024].

(Fitzpatrick, 2024)