

Large Scale Anonymous Port Scanning

By

Rodney R Rohrmann

A Master's Paper Submitted to the Faculty of the

DEPARTMENT OF MANAGEMENT INFORMATION SYSTEMS

ELLER COLLEGE OF MANAGEMENT

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2017

STATEMENT BY AUTHOR

This master's paper has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona.

Brief quotations from this master's paper are allowable without special permission, provided that an accurate acknowledgement of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part must be obtained from the author.

SIGNED: Rodney R. Rohrmann

APPROVAL BY MASTERS PAPER ADVISOR

This thesis has been approved on the date shown below:

05/05/2017

Dr. Mark Patton

Date

Professor of Management Information Systems

Contents

List of Figures	5
List of Tables	6
ABSTRACT.....	7
1 INTRODUCTION	8
2 PORT SCANNING.....	9
2.1 Port Scanning Overview	9
2.1.1 Connection-oriented scanner: Nmap.....	9
2.1.2 Connectionless scanners: ZMap and Masscan.....	10
3 ANONYMIZATION	11
3.1 Anonymization overview.....	11
3.1.1 VPN.....	12
3.1.2 Tor.....	12
4 LITERATURE REVIEW	13
4.1 Port Scanning	13
4.1.1 Literature.....	13
4.1.2 Key findings.....	14
4.2 Anonymization.....	15
4.2.1 Literature.....	15
4.2.2 Key findings.....	16

5 RESEARCH GAPS AND QUESTIONS.....	17
6 ANONYMIZATION METHODOLOGY	18
6.1 Design overview	18
6.2 Tool selection.....	19
6.2 Data collection	22
6.2.1 Packet analysis and scan tuning.....	23
6.2.2 Scan performance analysis.....	24
6.3 Scalability testing.....	25
7 TOOL DESIGN METHODOLOGY AND RESULTS	30
7.1 Design requirements	30
7.1.1 IP/Port input requirements	30
7.1.2 Number of scanners to run in parallel.....	30
7.1.3 Tor use specification	31
7.2 Tool coding and functionality.....	31
7.2.1 Functionality overview	31
7.2.2 Argument parsing.....	31
7.2.3 Port/IP parsing and bucket creation	32
7.2.4 Scan threading and output.....	33
7.2.5 Large scale tool example	33
7.2.6 Single device testing and output example.....	34

7.3 XML parsing and results example	35
8 REAL WORLD IMPACT	36
9 NEXT STEPS	37
9.1 Tool next steps	37
10 CONCLUSION.....	37
11 REFERENCES	39
12 APPENDIX A: TOOL SAMPLE CODE	41
13 APPENDIX B: XML PARSER SAMPLE CODE	42
13 APPENDIX B: LARGE SCALE SCANNING RESULTS	43

List of Figures

Figure 1. TCP Handshake. (Schidao)	9
Figure 2, ZMap visualized (CyberPunk)	11
Figure 3 Tor Network Example (Farivar, 2014).....	13
Figure 4 Research Design	18
Figure 5 Proxychains Example	20
Figure 7 Anonymization browser test through Proxychains.....	21
Figure 8IP address leak during ping	23
Figure 9 Anonymous scan results of SCADA device identified by Shodan	24
Figure 10 Performance monitoring during parallel scanning	27
Figure 11 Initial anonymization tests.....	27

Figure 12 Parallel scan results 29

Figure 13 Scanning tool input example 33

Figure 14 Tool single device scan results 34

Figure 15 Parser result example..... 35

List of Tables

Table 1 Research Design**Error! Bookmark not defined.**

Table 2 Nmap flags used.....**Error! Bookmark not defined.**

Table 3 Port 502 results**Error! Bookmark not defined.**

Table 4 Port 80 results**Error! Bookmark not defined.**

Table 5 Port 443 results**Error! Bookmark not defined.**

ABSTRACT

As computers become faster and more efficient, the ability to port scan large portions of the IPv4 range increases. Organizations such as the University of Michigan and Shodan have both created tools and open sourced their scan results, allowing researchers to use scan data to map and understand the IPv4 range. Though such sources exist, there are benefits of running scans internally to collect data. When sourcing port scans internally, there is a risk of the source scanning a target that may retaliate maliciously.

The practice of openly scanning ports, and allowing sites which are scanned to request an opt-out of future scans, is not always effective. Some individuals and organizations will attempt to retaliate if they detect a scan through malicious activities. To combat these retaliatory actions, I have developed a methodology to run port scans through Tor, which anonymizes the scans and mitigates the risk of retaliation. When scanning new portions of the IPv4 range, anonymous port scanning has been successfully achieved and is currently in use.

The goal of this research project was to identify a combination of anonymization methods and port scanning tools that successfully hide the source's IP address while providing an accurate port scan of the target. Further efforts were placed on the scalability and accuracy of such scanning methods when used on a large portion of the IPv4 range.

The research proved to be successful and I now have a tool that can be used to scan any port/IP combination in the IPv4 range while remaining anonymous. As scalability was a concern of the project, significant efforts were put into decreasing throughput time. This was achieved and I reduced the scan time of the test bed from an average of 10 hours down to an average of 5 minutes.

1 INTRODUCTION

The MIS department at the University of Arizona has a research initiative to conduct research on the landscape of the IPv4 range. This includes, but is not limited to, ICS devices, scientific instruments, IoT devices, and networking equipment. Raw data feeds from Shodan are downloaded and parsed into a database due to Shodan's ability to conduct a rolling scan of the IPv4 range in a two to four-week period. When ranges or devices of interest are being evaluated, additional data gathering may be required in the form of port scanning. To best preserve the security of the University and mitigate risk, there was a need to port scan anonymously. The team that created ZMap at the University of Michigan reported that when they were using their tool on the IPv4 range that they received threats of retaliation because of their scanning even though they have a public opt-out site (Dumeric, 2015). To avoid any such incident at the University of Arizona, I developed a method of scanning that hides the University's IP address while performing port scans.

The two key components of the anonymization technique were the method of anonymization and the port scanner to be used. Requirements for the project were that the tools used in the anonymization method needed to be open source, available in Linux, and provide consistent and accurate results every scan. Anonymization methods considered for the project were VPNs and Tor, with Tor ultimately being the selected method of anonymization. Selection of the port scanner itself was based on compatibility of the tool with the Tor network. Initially selected tools for testing were ZMap, Masscan, and Nmap. Nmap was the chosen port scanning tool as it is both accurate and meets the compatibility requirements for Tor. Once Nmap and Tor were paired, I was able to identify the settings and parameters necessary to successfully perform anonymous port scans by monitoring for data leakage while starting on a lab owned test bed and eventually migrating to public devices on the IPv4 range.

2 PORT SCANNING

2.1 Port Scanning Overview

Port scanners are tools that are used to check for open ports on devices. They can be run either locally or remotely. As with many tools, port scanners come in both open source and enterprise varieties. Examples of open source tools are Nmap, ZMap and Masscan; examples of enterprise solutions include Nessus (Tenable, 2016) and the HP Network Port Scanner (HP, 2017). Many of the enterprise solutions for port scanning are bundled into larger suites of tools for vulnerability assessments. The research for the anonymization tool focuses entirely on the use of open source tools for port scanning.

The three main types of port scanning are TCP, SYN, and UDP scanning. Both TCP and SYN scans are connection-oriented and UDP scans are connectionless.

2.1.1 Connection-oriented scanner: Nmap

The connection-oriented scanner that the team used during research and development of the anonymization tool is Nmap (Nmap, 2017). It is a tool that has evolved from being a simple port scanner to a tool that can scan websites, audit passwords, perform some vulnerability assessments, and run custom scripts. It relies mainly on TCP scanning. In TCP scanning, the

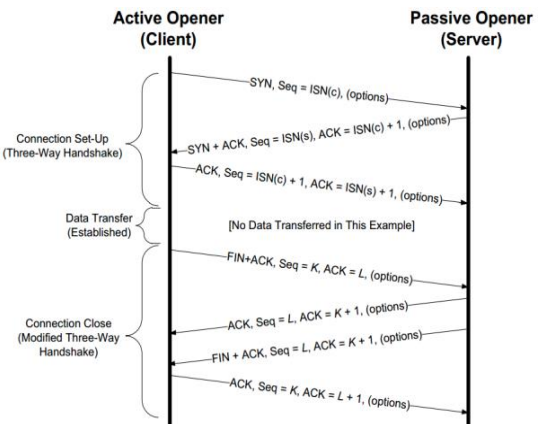


Figure 1. TCP Handshake. (Schidao)

host opens a TCP connection with the target using the three-way TCP handshake. If the target responds with a SYN ACK message, the port is marked as open and the connection is closed. If

the response is filtered or there is no response received, the port is marked as either filtered or closed. It is important to note that during TCP scanning the scanner must close the connection to avoid potentially causing a denial of service (DoS) to the target. A single scan will likely not create a DoS on a device, but if devices are scanned multiple times and connections never closed, the multiple open connections may cause the DoS if there are no processes in place on the target to kill TCP connections that are left open.

The benefit of using a TCP scan is that the results are much more accurate than those achieved during a UDP scan (Markowsky, 2015). The reason that Nmap was chosen as a connection-oriented scanner was because of the additional measures it takes to ensure accuracy during scans. While scanning, Nmap probes each of the specified ports and uses ID fields, probe sequence numbers, and source/destination ports to ensure accurate results. By creating an ID for each probe with source/destination data, Nmap can track each probe that is sent out and retransmit probes if they error out (Nmap, 2017). The drawback to Nmap is that it is much slower than scanners that utilize connectionless scanning by default. The reason for this is that Nmap puts all port/IP combinations in a queue and cannot move onto the next port/IP in the queue until one of the TCP connections has been released (Lyon, 2002). There are timing options in which Nmap can be set to allow for more TCP connections and faster throughput, but these scans are more easily detected and resource intensive on networking equipment, the sources, and targets.

2.1.2 Connectionless scanners: ZMap and Masscan

Connectionless scanners use UDP to scan instead of TCP for transmission. As UDP is not connection-oriented, it does not rely on the completion of one port probe to start the next probe in the queue. Scans are timed according to the setting in the tool.

By default, ZMap sends one probe per port/IP combination and claims to achieve up to 98 percent accuracy on a single probe per host (CyberPunk). In addition to the claim of achieving 98 percent accuracy on open ports, Zmap can scan the entire IPv4 address space in 45 minutes with a gigabit ethernet connection. Given a 10-gigabit connection, it can theoretically scan the entire IPv4 address space in under 5 minutes. The speed of connectionless scanners is the biggest reason to explore them as an option when developing a scanning tool. Below is an image based on the Zmap architecture; note specifically how packet transmission is not contingent upon a previous response and resources being freed.

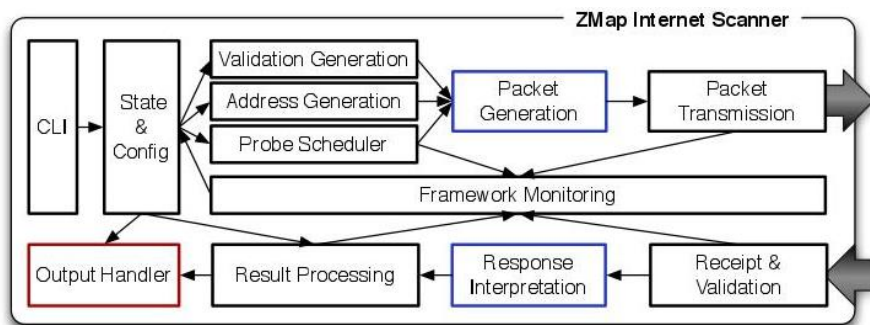


Figure 2, ZMap visualized (CyberPunk)

Though connectionless scanners are fast and claim to have nearly one hundred percent accuracy, they often do not perform in the real world as well as they do in closed environments and as such, are not as accurate as connection-oriented scanners (Markowsky, 2015).

3 ANONYMIZATION

3.1 Anonymization overview

The requirement for anonymization during the project was the ability to mask the original IP address of the scanning source. Anonymization methods for consideration were public Virtual Private Networks and Tor.

3.1.1 VPN

A Virtual Private Network(VPN) is a private network that routes all traffic from the source through a network designed specifically to allow for secure communications. They are commonly used in enterprise settings as they encrypt data between the source and the VPN, allowing users to access company materials on public networks without displaying the traffic in plain text. There are also public VPN's that are available for purchase, although many organizations that run public VPNs restrict port scanning in their license agreements. The benefits of using a VPN for port scanning are that the VPNs allow for faster throughput of scans and the port scans do not require any additional setup to be run. Additionally, many organizations such as NordVPN and PureVPN are public VPN's which have servers in multiple countries that allow users to select the servers manually which would further help throughput time by reducing latency (Eddy, 2017). Drawbacks of VPNs are that there is a monetary cost involved in using them and that port scanning is generally not allowed through VPN services.

3.1.2 Tor

Tor is a network that was developed in 1996 by the United States Navy that encrypts and anonymizes TCP traffic(Goldschlag, 1996). It is a network accessible to anyone with access to public internet and is used to anonymize web traffic. Originally created for secure communications in the Department of Defense, its use has grown to include researchers, citizens in countries with no freedom of the press, and people who use its anonymity to conduct illegal activities. Once a connection to Tor is established, traffic that is intended for the target is only decrypted at a Tor

exit relay and cannot be read by any other node used during transmission. Refer to figure 3 for a visualization on which traffic is encrypted during flight. In addition to security through encryption

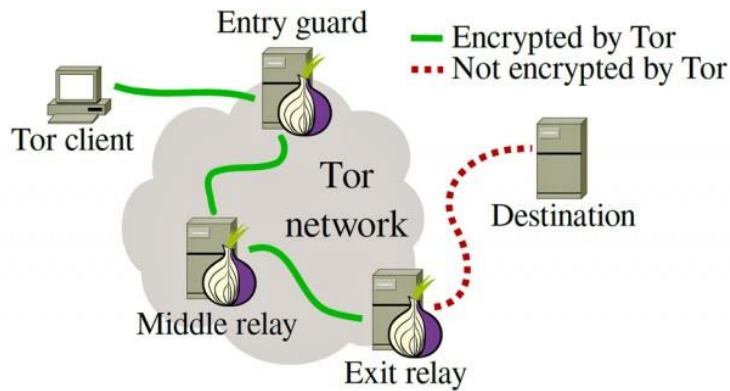


Figure 3 Tor Network Example (Farivar, 2014)

methods, there is an additional layer of security provided in that the source of the traffic is anonymized. Traffic is sent into the network and ‘bounced’ between multiple nodes in the network to aid in anonymization; middle relays only know the relay that the traffic was sent from and the relay its sending the data to in the next hop. Benefits of Tor are that it is always available and there is rarely any downtime, it allows for complete anonymization of source traffic, and can be used at no monetary cost to the source. Drawbacks of Tor are that it is much slower than using a VPN and since it only allows for TCP traffic, scanners that only use UDP to scan cannot have their traffic routed through Tor.

4 LITERATURE REVIEW

4.1 Port Scanning

4.1.1 Literature

Paper	Focus	Methods	Date Source	Results
Matthew (2014)	Nmap benchmarking on standard, registered, and unregistered ports	Scanning 45 devices running differing operating systems with Nmap	Nmap, Linux, Windows, Android, iOS	Different operating systems often have different standard ports, allowing for easier OS identification

Rohrmann (2016)	Nmap scanning performance through Tor	Scanning with Nmap	Scanning lab and public devices	Nmap is a viable tool for port scanning through Tor
Markowsky (2015)	Scanning of IoT devices and printers	Scanning devices with Nmap, Masscan, and PFT	Nmap, Masscan, and PFT	Nmap more accurate but slower than Masscan

4.1.2 Key findings

There are correlations between open ports and operating systems. Matthew’s team performed port scans of 45 different devices to identify similarities and try to identify the devices based on open ports. Devices running Linux, Windows, Android, and IOS operating systems were scanned. The research indicated that desktop operating systems have more ports open than mobile operating systems and that Windows devices usually have more open ports than Linux devices. Also interesting is that the team outlined that ports 0-1023 are assigned by the Internet Numbers Assigned Authority and usually run root processes (IANA, 2017). Ports 1024 through 49151 are registered ports and then 49152 through 65535 are private ports. The paper shows that there are correlations between open ports and operating systems, giving credence to the possibility of fingerprinting devices based on open ports.

The team working on this project had a paper published for the initial port scanning and anonymization methodology. Rohrmann scanned a lab owned device with various port scanning tools to benchmark scanning performance and viability of scanning through Tor. Findings from this paper were that using Nmap to scan through Tor, the source could completely hide their IP address from their target during a port scan. Benchmarking analysis showed that scan speed of ports 1-151 was significantly slower than scan speed of ports 152-65535. Additionally, running multiple scans in parallel on one device was possible and decreased total scan time of the target.

Markowsky’s research focused on reconnaissance and vulnerability detection techniques of IoT devices. The author leveraged Nmap to find vulnerable printers in an IP range. They scanned a total of 65536 on their organization’s class B network and. In the paper, Nmap is compared to Masscan and the author has greater success with Nmap as it is a more accurate scanner because of the connection-oriented nature of the tool. “If a busy router halfway along the network path drops 80% of the scanner’s packet flood, the [stateless] scanner sill consider the run successful and print results that are woefully inaccurate (Markowsky, 2015)”. When deciding on which scanning tool to use, the author tested Masscan for Heartbleed and Nmap for vulnerable printers on a subnet. What they outlined is that using a fast scanner to do initial reconnaissance on an IP range is favorable to using Nmap and that leveraging Nmap for detailed reconnaissance of specific devices is a better utilization of the tool because of its speed.

4.2 Anonymization

4.2.1 Literature

Paper	Focus	Methods	Date Source	Results
Goldschlag (1996)	Development and analysis of onion routing	Using onion routing to limit ability to analyze traffic in the network	Asymmetric encryption over HTTP	Traffic could be successfully transmitted to the target without being analyzed by intermediate nodes in the network
Shirazi (2015)	Using open source Tor network tools	Setting up internal Tor networks with open source tools	Shadow, TorPS, and ExperimentTor	Performance varied between tools and the team identified 10 key metrics for analyzing performance
Ghafir (2014)	Blocking Tor traffic	Maintaining active lists of Tor exit nodes and blacklisting all traffic from them	IDS, IPS, firewalls, and open source Tor exit node lists	Most traffic was identified and stopped within one second of reaching the target network

4.2.2 Key findings

Onion routing was developed by Goldschlag's team in 1996. It uses asymmetric encryption and node-based networks that hide the source's address from all devices in the network besides the entry node and allows for secure communication as the traffic is encrypted. Onion routing consists of a network of routing and Proxy nodes. Routing nodes only know the next routing node that the traffic is addressed to and do not have access to the originating source or intended target of the traffic. Multiple routing nodes are used during transmission to aid in anonymization.

Since Tor is a public network and nodes in the network aren't owned by a single source, researching the nature of traffic flow and performance in Tor is unfeasible. The focus of Shirazi's research is to outline methods to test the potential performance and routing nature of Tor without routing traffic through the public Tor network. Since Tor is open source, there are tools and methods that exist which would allow users to create their own internal Tor networks. Tools such as Shadow, TorPS, and ExperimentTor allow researchers to test anonymization techniques without sending traffic through the public Tor network.

Though Tor is intended for ethical usage, there are actors who leverage the anonymization capabilities of Tor to carry out malicious attacks. Ghafir researched the method of blocking connections to a network by keeping an active directory of tor exit nodes and blacklisting all Tor devices in the organization's firewall. Tor maintains a current list of exit nodes on the Tor project website which can be used for creating blacklists (Tor, 2017). According to the author, most traffic was detected by the IDS in 0 seconds and the remaining traffic was detected in 1 second. The focus was on stopping Tor traffic before an attacker got into a network to install malicious malware onto targets. The assumption was that by keeping a current list of Tor exit nodes that they would be able to mitigate threats in which attackers used tor to carry out the attacks.

5 RESEARCH GAPS AND QUESTIONS

Achieving anonymous port scanning and developing an anonymous scanning tool had not been formally explored in academia before this project. Multiple online sources claimed that it was possible and there were blogs that had tutorials on how to set up anonymous scanning, but none supported the claim of anonymity with network packet captures to ensure that anonymity was achieved. Initial testing and research was centered around the following questions:

- Which tools and techniques allow a source to completely anonymize their IP address from their target during a port scan?
- What flags must be set in Nmap during a port scan through Tor to obscure the source's IP address?
- At what point during the scan was the source's IP address leaked and what flags caused leaks to happen?

After anonymization was achieved, focus was changed to explore ways in which scanning could be done in a more scalable way. Research concluded in the spring of 2016 concluded that anonymous scanning was possible but it would take hundreds of years to scan the entire IPv4 range with the initial method used to achieve anonymous port scanning. The questions addressed during the next phase of the project were as follows:

- How can port scanning through Tor be done in a way that is scalable?
- How can port scanning through Tor be done in a way that is repeatable?
- How do scans from a third party such as Shodan or the University of Michigan compare in reliability to scans performed from our own lab through TOR?

Once a tool had been created that could scan at a reasonable rate allowing scalable exploration, focus was placed on testing the results of the tool against existing third-party scans of the IPv4 range. The goal of the last phase was to determine whether using Tor to scan through was reliable, as there are many public lists of the exit nodes which people use for blacklisting.

6 ANONYMIZATION METHODOLOGY

6.1 Design overview

Creating an anonymous port scanning method was centered around the goals of successfully achieving anonymous scanning and making the scanning process scalable and repeatable. The process for research design and tool creation was broken up into three main phases: tool selection, data collection, and scalability testing. Analysis of results was completed at the end of each phase.

Refer to figure 4 for an overview of the process:

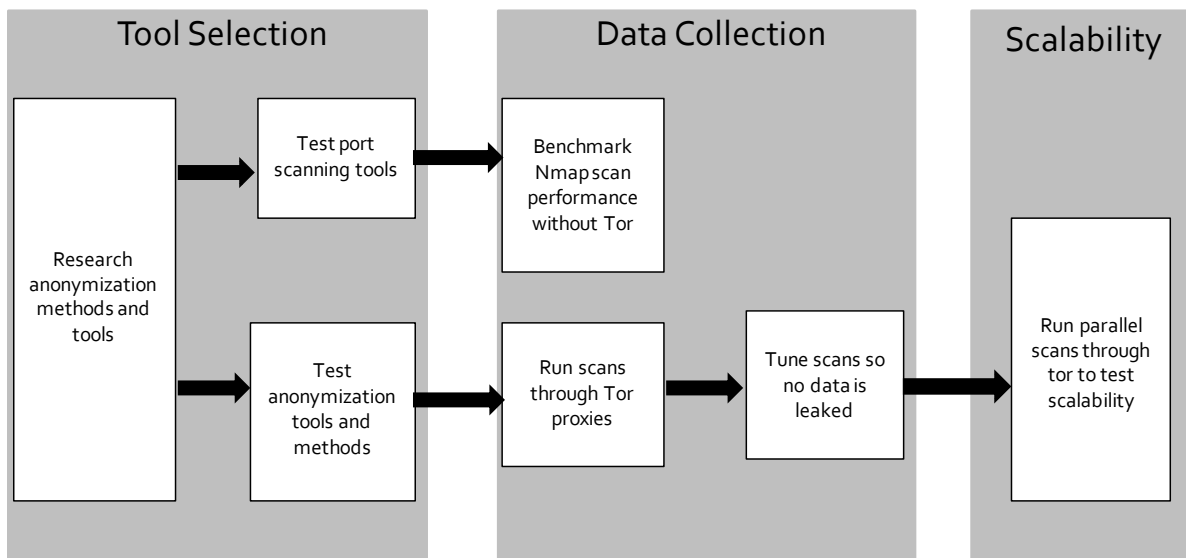


Figure 4 Research Design

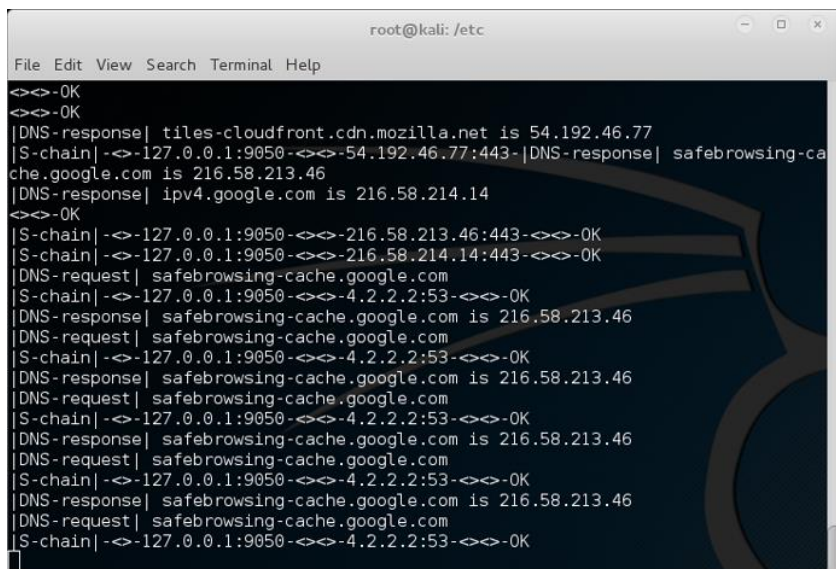
6.2 Tool selection

The initial phase of the project was research of port scanning/anonymization methods and testing of the selected methods. Though no work had been done in academia specifically addressing anonymous port scanning, there were many papers that addressed the selection of port scanning tools and anonymization methods. The research phase yielded Nmap, Zmap, and Masscan as potential port scanning candidate tools which were used during the testing phase. Tor would be the sole anonymization medium tested. Two methods for routing Tor traffic were identified and tested for the ability to provide accurate results during scanning.

To test the selected port scanners, anonymization methods needed to first be explored to ensure that they would work in a normal setting such as in a browser or email client. Two approaches were taken when exploring options for anonymization: complete anonymization of all traffic generated from the operating system (OS) or anonymization at an application level.

For the OS level anonymization, a router virtual router from Whonix that connects to Tor by default was used (Whonix, 2017). This required the VM to be run in VirtualBox alongside the router and for specific network setting to be used in both the VirtualBox hypervisor and the virtual machine (VM). Benefits of using the Whonix router were that all traffic from the source was anonymized and that the firewall on the router stopped any traffic that was not TCP, as Tor only accepts TCP traffic. If communicating with a protocol that is not TCP such as UDP or ICMP, the firewall blocked the traffic at the source hence the source's IP address was not exposed to the target. Benefits of Whonix were that no leaks occurred of any kind during use and that it is open source and so could be configured to the team's needs. Cons were that the firewall comes configured in a way that port scans were being stopped at the firewall and incorrect results were reported during port scans.

To achieve anonymization at an application level, Proxychains was used in conjunction with TOR (Proxychains, 2013). It is a local proxy that allows the user to specify a proxy to send traffic through it, and it uses Tor by default. To use Proxychains, Tor must be installed on the machine



```
root@kali: /etc
File Edit View Search Terminal Help
<<<>-OK
<<<>-OK
|DNS-response| tiles-cloudfront.cdn.mozilla.net is 54.192.46.77
|S-chain| -<-127.0.0.1:9050-<<<-54.192.46.77:443-|DNS-response| safebrowsing-cache.google.com is 216.58.213.46
|DNS-response| ipv4.google.com is 216.58.214.14
<<<>-OK
|S-chain| -<-127.0.0.1:9050-<<<-216.58.213.46:443-<<<-OK
|S-chain| -<-127.0.0.1:9050-<<<-216.58.214.14:443-<<<-OK
|DNS-request| safebrowsing-cache.google.com
|S-chain| -<-127.0.0.1:9050-<<<-4.2.2.2:53-<<<-OK
|DNS-response| safebrowsing-cache.google.com is 216.58.213.46
|DNS-request| safebrowsing-cache.google.com
|S-chain| -<-127.0.0.1:9050-<<<-4.2.2.2:53-<<<-OK
|DNS-response| safebrowsing-cache.google.com is 216.58.213.46
|DNS-request| safebrowsing-cache.google.com
|S-chain| -<-127.0.0.1:9050-<<<-4.2.2.2:53-<<<-OK
|DNS-response| safebrowsing-cache.google.com is 216.58.213.46
|DNS-request| safebrowsing-cache.google.com
|S-chain| -<-127.0.0.1:9050-<<<-4.2.2.2:53-<<<-OK
|DNS-response| safebrowsing-cache.google.com is 216.58.213.46
|DNS-request| safebrowsing-cache.google.com
|S-chain| -<-127.0.0.1:9050-<<<-4.2.2.2:53-<<<-OK
```

Figure 5 Proxychains Example

used for scanning and the tor service must be running when Proxychains is called from the command line. Additionally, one configuration file needs to be modified to successfully tunnel Tor traffic through Proxychains. An example of successful Proxychains

tunneling can be seen to the left; note how traffic is tunneled from the local address on port 9050 where Tor is running (127.0.0.1:9050) to a public address for google.com. Benefits of Proxychains were that it allowed the user to only access the Tor network at an application level, allowing them to bypass Tor whenever using Tor wasn't necessary. The ability to selectively use Tor allowed for port scanning benchmarking as scans would later be run back to back both through Tor and outside of Tor and compared for speed and accuracy.

After comparing the anonymization methods for ease of setup and use as well as ability to successfully port scan, Proxychains was chosen as the preferred anonymization method. There were fewer setup steps in using Proxychains and it allowed for faster switching between using and bypassing Tor. To confirm that anonymization was achieved during the test phase before port scanning efforts, 'what is my IP' was used in the Google search engine. It allowed for fast

confirmation that the Tor service was running successfully before targets were scanned. See image 6 for the result of Google search and confirmation of successful anonymization (46.105.61.142 is a Tor exit node in France):

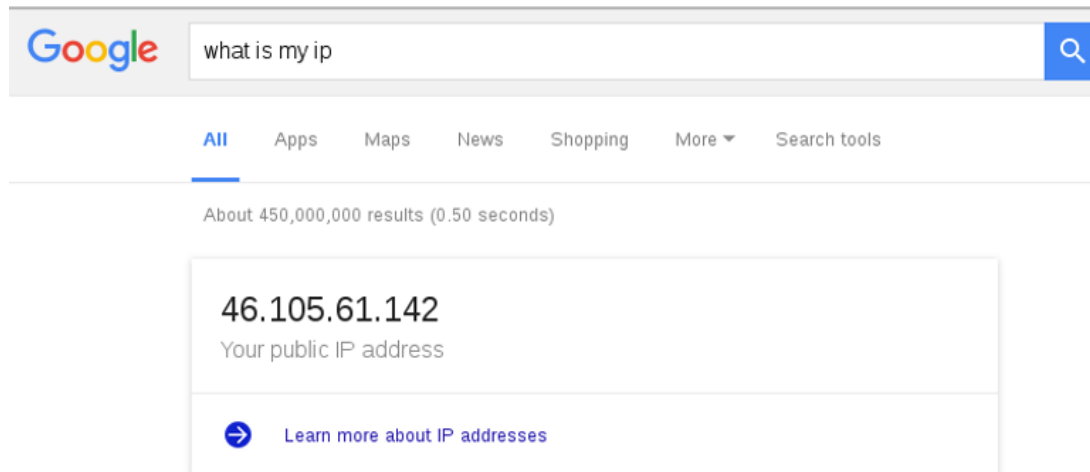


Figure 6 Anonymization browser test through Proxychains

Testing of the port scanning tools was done both through the Tor network and bypassing Tor to test speed and ability to route traffic through the network. Once all tools were installed, each scanner was tested for anonymization. Guides in blogs had stated that using Nmap through Tor was possible but there was no mention of Zmap or Masscan being used. After further researching Zmap and Masscan, they were identified as unusable for testing as they have no option to force a connection-oriented scan using TCP. As Tor only accepts TCP traffic and Zmap and Masscan only utilize UDP for port scans, using them with Tor was not feasible.

Nmap was chosen as the sole port scanning tool to be used based on its ability to run scans through TCP. Initial testing of the tool was to ensure that it would accurately scan a device through Tor. To set up the test, a lab owned VM with a public facing IP address was used. Flags in Nmap were set based on flags that were identified in blogs that would ensure a successful port scan through Tor. The most important flag used during the tool selection phase was the flag that forced Nmap

to use TCP as the sole networking protocol to perform the port scans. The virtual machine used for scanning was run on a local host and the target for scanning was run on a separate host that allowed the use of setting up the machine with a public facing IP address so it could be accessed from the Tor network. Devices behind a NAT could not be directly accessed for scanning so it was important to use a device with a public IP for testing. Below is a table that outlines information on the host, scanning VM, and target machine:

Table 1 Research Design

Machine	Operating System	Software Used
Host	Windows 10	VirtualBox 5.0.10, Wireshark 2.0.1
VM	Kali Linux, Gnome 3.14.1	Nmap 7.0.1 Wireshark 1.12.6 Proxychains 3.1-6
Target	Ubuntu 14.0.1	Wireshark 2.0.1

6.2 Data collection

Once Proxychains and Nmap were identified as the tools to be used for the duration of the project, the next step of the research design was to collect data during scans such as benchmarking of scan performance, validating the Nmap flags needed to be set during a scan to achieve complete anonymization. Three elements of every scan were recorded:

1. Scan Time
2. Whether the IP address was leaked (only during scans through Tor)
3. Complete packet captures of the host, scanning VM, and target VM

6.2.1 Packet analysis and scan tuning

The most meaningful data collected during the early phase of scanning was the packet captures on the host, scanning VM, and target VM. This allowed every scan to be analyzed at the packet level. Data collected from Wireshark was used to identify at which what phase of the port scan that the IP address was being leaked. Scan settings were adjusted, or tuned, to eliminate these leaks. What the PCAP files revealed was that the IP address of the source was never leaked during the scans themselves, but rather when additional Nmap services were run such as pings or OS detection.

Below is an example of an IP address leak that was captured during the initial data collection phase. Tor does not allow ICMP traffic and thus, the ping during the Nmap scan had to be disabled using the “-Pn” flag

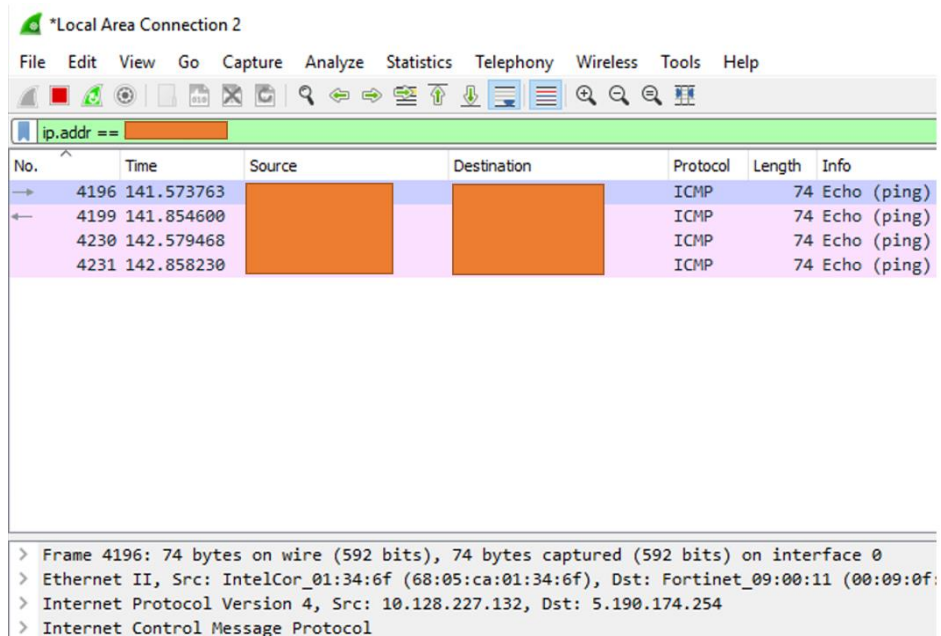


Figure 7 IP address leak during ping

which treats all remote hosts as online and disables ping during the discovery phase of an Nmap scan. This decreases performance as if Nmap pings a target and doesn't receive a response, it moves to the next target, but is necessary to ensure complete anonymization during all phases of the scanning process. Using Wireshark as a guide for successful and unsuccessful anonymization attempts, the following results were recorded and used in the next phases of the project.

Table 2 Nmap flag results

Flags Used	IP Address Leak
“-A”	Yes
“-sT, -A”	Yes
“-sT, -Pn”	No
“-sT, -Pn, -sV”	No
“-sT, -Pn, -sV, -O”	Yes

6.2.2 Scan performance analysis

After Wireshark data had been analyzed and anonymization attempts were proven successful, the anonymization methods were used to conduct port scans of all 65535 ports on the target VM and

```

root@kali: /etc
File Edit View Search Terminal Help
[proxychains] Strict chain ... 127.0.0.1:9050 ... :54311 <-denied
[proxychains] Strict chain ... 127.0.0.1:9050 ... :53971 <-denied
[proxychains] Strict chain ... 127.0.0.1:9050 ... :2506 <-denied
[proxychains] Strict chain ... 127.0.0.1:9050 ... :48912 <-denied
[proxychains] Strict chain ... 127.0.0.1:9050 ... :25999 <-denied
Nmap scan report for 
Host is up (0.42s latency).
Not shown: 65524 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
102/tcp   open  iso-tsap
502/tcp   open  mbap
1433/tcp  open  ms-sql-s
1723/tcp  open  pptp
2000/tcp  open  cisco-sccp
3389/tcp  open  ms-wbt-server
4081/tcp  open  unknown
8010/tcp  open  xmpp
8110/tcp  open  unknown
8291/tcp  open  unknown
8729/tcp  open  unknown
Nmap done: 1 IP address (1 host up) scanned in 29337.27 seconds
root@kali: /etc#

```

Figure 8 Anonymous scan results of SCADA device identified by Shodan

time was between 8 and 10 hours for the device. Once anonymization had been confirmed during multiple full port scans of the target VM, the anonymous scanning method was used on multiple devices on the IPv4 range.

various machines on the IPv4 range. Scan throughput without Tor of the target VM averaged around 150 seconds or two and a half minutes to scan every port of the device. Through Tor, total scan

Using Shodan data, 2 SCADA devices were identified and scanned in their entirety. Figure 8 on the previous page is a screenshot of the Nmap results page displayed after scan completion. Note specifically the total scan time of 29337.27 seconds or around 8.1 hours, with a scan rate of around 2 ports per second. Both the SCADA device in figure 8 and the Tor exit node were in the United States. When scanning a device in Syria with a Tor exit node located in Europe, scan time was roughly 16.5 hours.

After scanning multiple devices in differing regions in the world, the data showed that the two biggest factors that affected scan time were the Tor exit nodes and the physical location of the targets to be scanned in relation to the exit node. Given that Tor is an open public network, nodes in the network have differing resources and bandwidths and the exit node being used at any given time may have lower resource allocations than other nodes on the network. Tor had previously allowed users to specify which exit nodes they wanted to route traffic through but that functionality was removed due to users abusing exit nodes with higher bandwidth. The network now randomly selects the exit node to be used upon connection. Restarting the Tor service often results in a new exit node being selected and there are sources online which display the bandwidths of known exit nodes, so it was useful to restart the service until one of the faster exit nodes was being used. This method allowed throughput time to decrease by between 2 and 8 hours, with best performance at around 8 hours of a complete scan of a device.

6.3 Scalability testing

Upon successfully ensuring that anonymous scanning was possible, further efforts were placed on how to make the scanning as scalable as possible. Nmap threads scans by default but when traffic is routed through Proxychains, scans are run serially regardless of the Nmap timing options that are set. Nmap is naturally slower than connectionless scanners because of the need to maintain

connections until responses are received from the target and the lost functionality of threading was the main contributing factor as to why scan time went from two and a half minutes to around eight hours.

To decrease throughput time, multiple scans were run at a single time, with the following two methods identified as potential solutions for increasing performance:

1. Running multiple virtual machines with low overhead and running one scan per machine
2. Running multiple scans in parallel on a single virtual machine with additional resources

Method one of running multiple virtual machines was briefly explored; by setting up multiple instances of a GUI-less operating system that required few resources, between 5 and 10 virtual machines could be run on a single workstation, decreasing throughput time by up to ten times. Ubuntu server with Nmap, Tor, and Proxymchains installed was identified as the solution for such scanning and was to be used if multiple scans could not be run in a single virtual machine.

Method two of running multiple instances of Nmap in a single virtual machine was preferable as a single Nmap scan is a far more efficient use of computing resources than an entire virtual machine. The main concern was that multiple scans could not be run though the Tor proxy at one time and that there would be interference. Small scale testing of no more than three scanners running in parallel was done and results were satisfactory. Running two scans in parallel cut scanning time in half. Once parallel scans on one machine was identified as a viable solution to scalability, testing was done to benchmark resource allocation requirements for multiple scans to be run.

During initial parallelization testing, small ranges of 5-10 ports were used per scanner and the number of scanners running in parallel gradually increased until adding one scanner did not decrease total throughput time. Given the slow scan rate through Tor, small port ranges were preferable as they allowed performance to be benchmarked more quickly. Regardless of number of ports scanned, each scanner

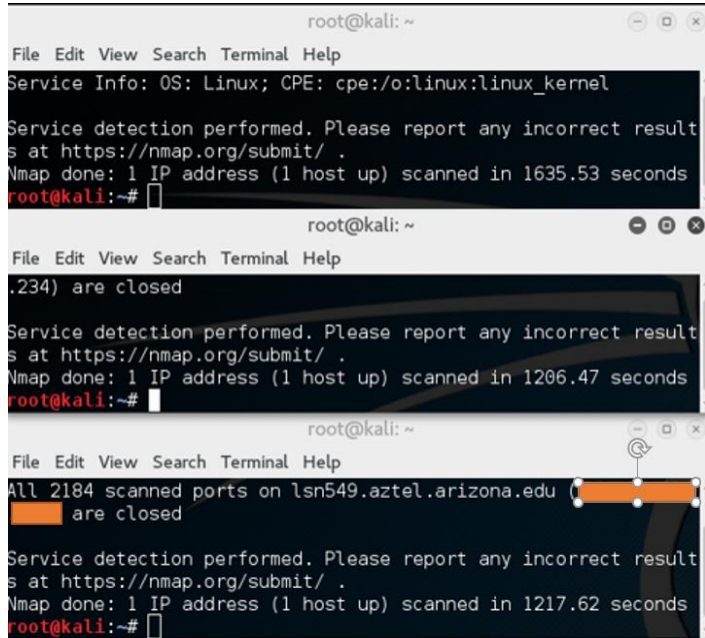


Figure 9 Initial anonymization tests

scanned between one to two ports per second. Testing of parallelization consisted of writing Nmap commands that were to be sent to each scanner in a word processing document and pasting each command into the corresponding shell that was open for each command. Once the number of scanners in parallel exceeded 10, additional ports needed to be included in each scan so the beginning scans did not complete before the end scans started. An example of a parallel scan test can be seen in figure 9. Performance of the virtual machine was monitored using the 'top' command in Linux. Figure 10 is an image captured from 66 parallel scans running at one time on a machine that was allocated 4GB of RAM. Of interest in figure 10 is that with 66 scans running

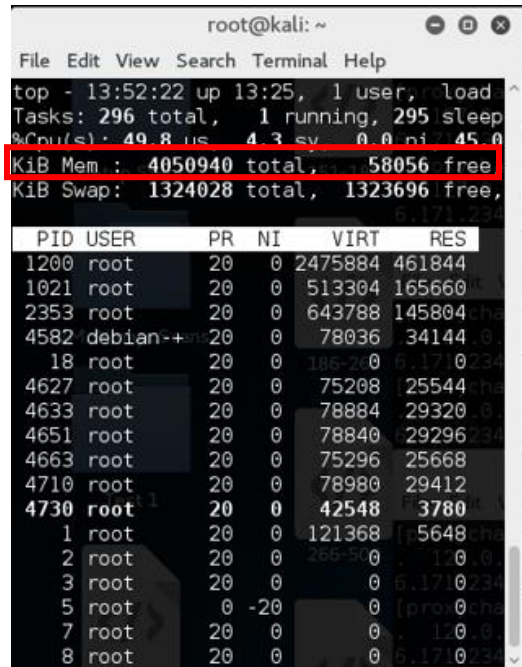


Figure 10 Performance monitoring during parallel scanning

of the virtual machine was monitored using the 'top' command in Linux. Figure 10 is an image captured from 66 parallel scans running at one time on a machine that was allocated 4GB of RAM. Of interest in figure 10 is that with 66 scans running

in parallel, nearly all the allocated RAM had been used. In addition to the RAM usage, 66 scans were the most that could be run manually without any of the shells on the screen overlapping and increasing the risk of pasting commands into the wrong shell.

Factors that affected performance during parallelization efforts were the ports selected for each individual scanner and the Tor exit node used for each test. The Tor service was restarted multiple times during parallelization efforts to capture variances to be expected during normal scanning operations. Port selection for each scan was significant because scan time for ports 1-151 took longer than ports 152-65535. The reason scan time was longer for the first 151 ports every scan was that instead of responding as closed, the ports didn't respond and Nmap waited until the connection timed out. Ports 1-151 had to be separated into 25 ports per individual scanner to complete at the same time as scans that were running around 1100 ports. Table 3 is an example of the data that was collected during the parallel scanning testing. Note that for scan 1, ports 1-250 resulted in a pointer error and did not complete. By reducing the range to 50 ports per scanner in scan 2, performance was less than one port per second but did not result in a pointer error.

Table 3 Parallel scan data collection

Scan	Parallel scans	Time (seconds)	Time (minutes)	Range start	Range end	Total Ports	Ports/second	Comments
1	4	error	#VALUE!	1	250	250		Pointer error
1	4	226	3.8	251	500	250	1.11	
1	4	265	4.4	501	750	250	0.94	
1	4	203	3.4	751	1000	250	1.23	
2	8	297	4.9	1	50	50	0.17	
2	8	34	0.6	51	100	50	1.46	

After parallel testing, total throughput time was reduced from 8-10 hours with a single scanner to between 10 and 15 minutes with 66 scans running in parallel. Each scanner averaged between 1

and 2 ports scanned per second. Back to back tests were run with 66 scanners in parallel through different Tor exit nodes; one scan completed in 14.6 minutes and the other in 10.9 minutes. Throughput time was impacted most by the resources allocated to machine with RAM being the limiting factor in ability to run additional scans. Refer to figure 11 for a visualization of scan performance during testing.

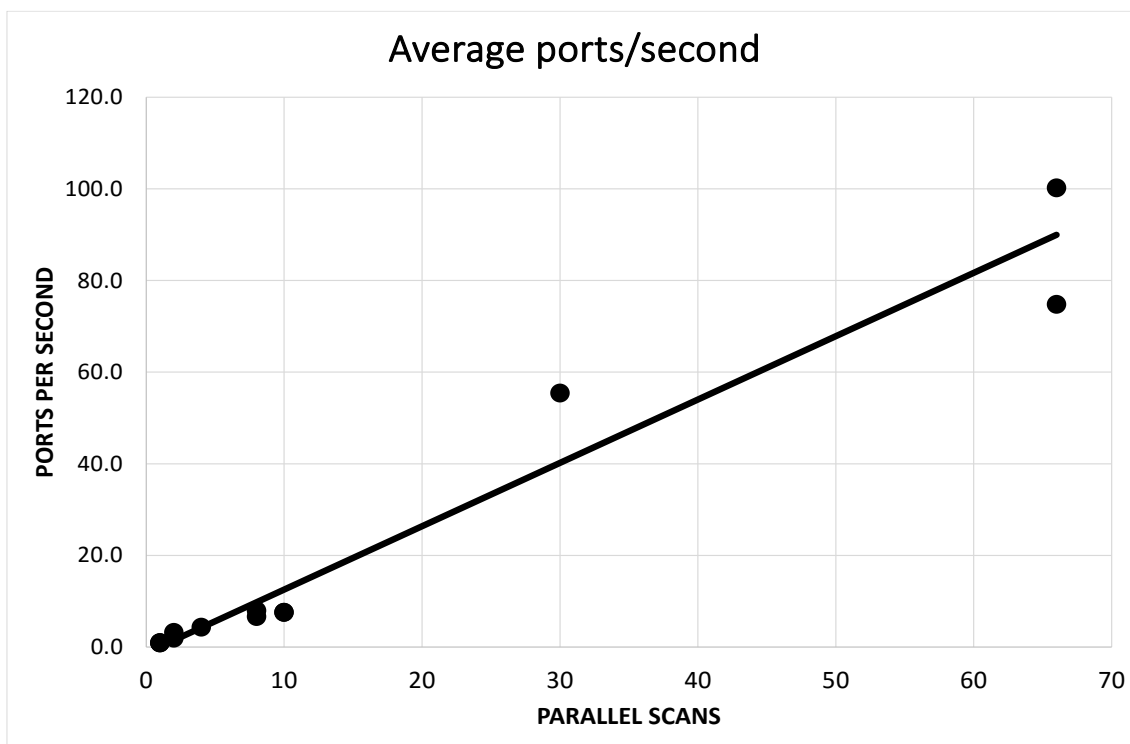


Figure 9 Parallel scan results

The initial phase of research was successful. Anonymous scanning had been achieved and tests were done that allowed for tuning of scans so there was no leak of the source's IP address at any point during the scan. Shifting the focus to scalability, parallel scans were successfully run on a virtual machine and the amount of RAM allocated was identified as the limiting factor in the ability to run additional scans. Based on the results of the initial research design and analysis, the next step was to create a tool which would allow the scanning efforts to be done in a way which was more scalable and repeatable.

7 TOOL DESIGN METHODOLOGY AND RESULTS

7.1 Design requirements

To further test resource requirements when additional resources were allocated to the scanning VM and test parallelization on a large scale, a tool needed to be written that the user could specify IP/port ranges and they be split up evenly into multiple scanners. The following were identified as key functionalities of the tool during the requirements gathering phase:

- Specification of IP addresses and ports to be scanned in range or comma separated format
- Selection of how many port scanners to run in parallel
- Ability to select whether to scan through Tor or not

7.1.1 IP/Port input requirements

Creating a tool that is robust and can handle multiple types of IP and port arguments was one of the most important design decisions. Depending on the type data being collected, the user may want to scan every port of every IP in a list, a single port for every IP address, or multiple ports in a comma separated input. It was important to allow the user flexibility in port/IP inputs.

7.1.2 Number of scanners to run in parallel

One of the most significant observations during the parallelization testing phase was that copying and pasting Nmap commands into multiple shells was inefficient and ineffective if the user accidentally miscopied or pasted in the wrong shell. The user had to manually open and resize as many shells as they needed during scanning. By specifying how many scans to run in parallel at runtime, more in depth testing could be done regarding resource allocation and use to further tune scans and achieve the best performance to resource use ratio.

7.1.3 Tor use specification

Though the tool was designed with parallel scanning through Tor in mind, there are situations in which the user will not want to use Tor. If there is an agreement with the target and the source has no need to hide their IP address, it is beneficial to scan outside of Tor to decrease throughput time. Through very brief testing, results suggested that running multiple Nmap scans in parallel outside of Tor decreased total throughput time. For a user not interested in hiding their IP address and scanning as quickly as possible, having the option to use a command line tool for parallel scanning is beneficial.

7.2 Tool coding and functionality

7.2.1 Functionality overview

Once the key features of the tool were identified in the requirements gathering phase, specifications were created for the processes to be handled by the tool. It was broken into four main processes including:

- Argument parsing
- Port parsing and bucket creation
- IP parsing and bucket creation
- Scan threading and shell output

7.2.2 Argument parsing

To handle multiple use cases, arguments had to be coded that allowed the user flexibility when sending scan commands. The following arguments and their functionalities are as follows:

- **Port input:** Either a single port, comma separated list, or range (e.g. 1-100) of ports to be scanned

- **IP input:** Either a single IP, comma separated list, or range (127.0.0.1-127.0.0.10) of ports to be scanned
- **Port buckets:** The number of port ‘buckets’ to be created for scanning.
- **IP buckets:** The number of IP ‘buckets’ to be created for scanning.
 - Total scanners created is port buckets * IP buckets. 4 port buckets and 5 IP buckets will yield 20 parallel scans
- **Anonymization:** Allows the user to either scan through Tor or straight to the target, circumventing Tor
- **Shell keep-alive:** Either keeps the Nmap shells alive after scanning or kills them (killing shells automatically is useful when hundreds of scans are run in parallel and this functionality was added during the second iteration of tool development)

7.2.3 Port/IP parsing and bucket creation

Port/IP parsing and bucket creation were two different processes in the tool but the behavior of the two processes is similar. The first subprocess parses the user input and each value parsed is inserted as an element in a list. Ports were inserted as integers and IP addresses were converted from the IP address as a string in IP address format (1.1.1.1) to an integer for easier separation into IP buckets. Once parsed into a list, the list is shuffled so the outputs to Nmap are randomized, decreasing the likelihood that IPs and ports will be serially scanned which reduces the likelihood of the Tor exit node being blacklisted during scanning. The second subprocess involves calculating how many IPs or ports will go into each bucket. This is done by dividing the total count of elements in the list by the user specified integer in the bucket argument. Any remainders are added to a separate bucket. Once bucket size has been calculated, a new list is created which contains each IP or port bucket. The buckets are lists that are placed inside of the IP or port list, creating a list of

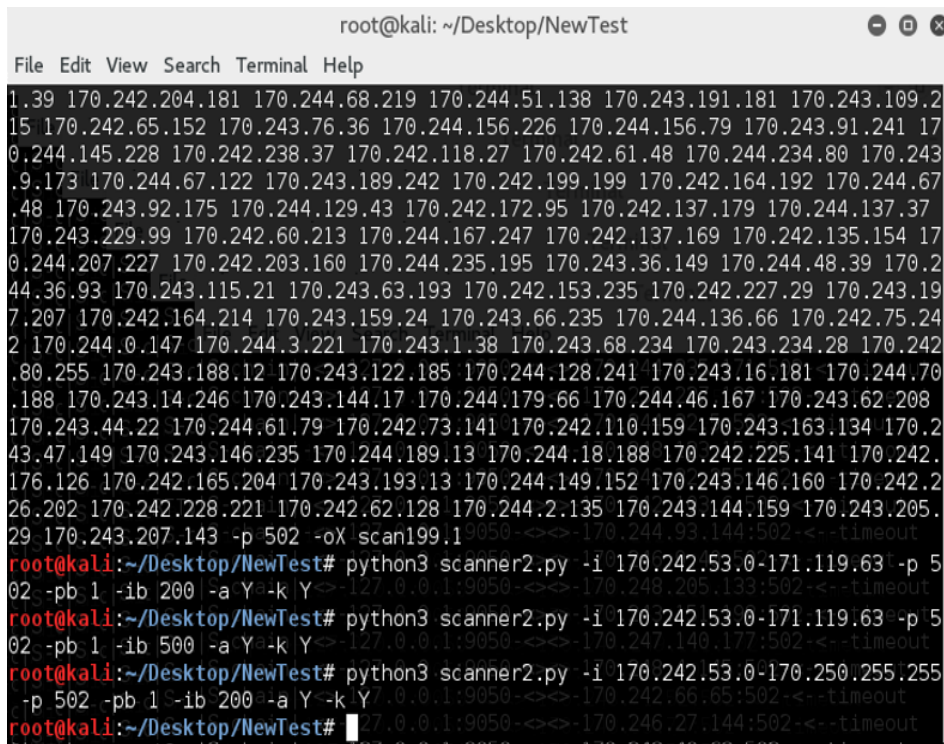
lists which will be iterated through during scan creation. Finally, the IP integers are converted back into IP addresses.

7.2.4 Scan threading and output

In the final process of the tool, threading is used to send multiple commands to the OS simultaneously. The thread subprocess iterates through the IP bucket list to a worker thread, which sends commands to the shell. A nested for loop is used that iterates through every port bucket in every IP bucket. The worker thread is responsible for parsing inputs from the lists into bash commands that start the Nmap scans. Also in the worker thread are 'if' statements that deal with the keep-alive and anonymization arguments. Nmap results are output to a file in the directory of the scanner and given unique identification numbers so no files are overwritten.

7.2.5 Large scale tool example

See below for an example of tool input and walkthrough of how the tool handles IP/port bucket



```
root@kali: ~/Desktop/NewTest
File Edit View Search Terminal Help
1.39 170.242.204.181 170.244.68.219 170.244.51.138 170.243.191.181 170.243.109.2
15 170.242.65.152 170.243.76.36 170.244.156.226 170.244.156.79 170.243.91.241 17
0.244.145.228 170.242.238.37 170.242.118.27 170.242.61.48 170.244.234.80 170.243
.9.173 170.244.67.122 170.243.189.242 170.242.199.199 170.242.164.192 170.244.67
.48 170.243.92.175 170.244.129.43 170.242.172.95 170.242.137.179 170.244.137.37
170.243.229.99 170.242.60.213 170.244.167.247 170.242.137.169 170.242.135.154 17
0.244.207.227 170.242.203.160 170.244.235.195 170.243.36.149 170.244.48.39 170.2
44.36.93 170.243.115.21 170.243.63.193 170.242.153.235 170.242.227.29 170.243.19
7.207 170.242.164.214 170.243.159.24 170.243.66.235 170.244.136.66 170.242.75.24
2 170.244.0.147 170.244.3.221 170.243.1.38 170.243.68.234 170.243.234.28 170.242
.80.255 170.243.188.12 170.243.122.185 170.244.128.241 170.243.16.181 170.244.70
.188 170.243.14.246 170.243.144.17 170.244.179.66 170.244.46.167 170.243.62.208
170.243.44.22 170.244.61.79 170.242.73.141 170.242.110.159 170.243.163.134 170.2
43.47.149 170.243.146.235 170.244.189.13 170.244.18.188 170.242.225.141 170.242.
176.126 170.242.165.204 170.243.193.13 170.244.149.152 170.243.146.160 170.242.2
26.202 170.242.228.221 170.242.62.128 170.244.2.135 170.243.144.159 170.243.205.
29 170.243.207.143 -p 502 -oX scan199.1
root@kali:~/Desktop/NewTest# python3 scanner2.py -i 170.242.53.0-171.119.63 -p 5
02 -pb 1 -ib 200 -a Y -k Y <-> 127.0.0.1:9050 <-> 170.248.205.133:502 <-> timeout
root@kali:~/Desktop/NewTest# python3 scanner2.py -i 170.242.53.0-171.119.63 -p 5
02 -pb 1 -ib 500 -a Y -k Y <-> 127.0.0.1:9050 <-> 170.247.146.177:502 <-> timeout
root@kali:~/Desktop/NewTest# python3 scanner2.py -i 170.242.53.0-170.250.255.255
-p 502 -pb 1 -ib 200 -a Y -k Y <-> 127.0.0.1:9050 <-> 170.242.66.65:502 <-> timeout
root@kali:~/Desktop/NewTest# <-> 127.0.0.1:9050 <-> 170.246.27.144:502 <-> timeout
```

Figure 10 Scanning tool input example

creation. The three commands in figure 12 were captured when the tool was being tested large scale on the public IPv4 range. The first two commands were used to test how the tool would handle incorrect user input

(note the incomplete IP address input) and the final command was used to test the scanning capability. The input `python3 scanner2.py -I 170.242.53.0-170.250.255.255 -p 502 -pb 1 -ib 200 -a Y -k Y` separates the input into 200 scanners (200 port buckets * 1 IP bucket = 200 buckets). Approximately 576,000 IP's were scanned in the example for port 502, with around 2800 IP addresses per bucket being scanned.

7.2.6 Single device testing and output example

To benchmark throughput of more than 66 scanners in parallel, the tool was run against the target VM created in in the research design phase. With more resources allocated to the scanning VM,

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
[S-chain] -<-127.0.0.1:9050-><-><-><-> :36663 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :31154 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :50082 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :17758 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :26791 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :13974 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :65129 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :16561 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :10927 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :7617 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :32669 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :64849 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :12974 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :27166 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :38981 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :60384 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :38406 <--denied
[S-chain] -<-127.0.0.1:9050-><-><-><-> :20801 <--denied
Nmap scan report for lsn549.aztel.arizona.edu ( )
Host is up (0.69s latency).
All 327 scanned ports on lsn549.aztel.arizona.edu ( ) are closed
Nmap done: 1 IP address (1 host up) scanned in 310.47 seconds
root@kali:~/Desktop#
```

Figure 11 Tool single device scan results

additional scanners were run in parallel. On a device with 4 cores and 8GB of RAM, the scanning limit was around 200 scanners at one time. The keep-alive flag was set to Y in to allow for easier analysis of each

scanner. The IP address of the target is hidden and port numbers are highlighted green. Ports are randomized and by the tool as seen by the highlighted area, Note the Total scan time in figure 13 was 310 seconds for this scanner, reducing the throughput time of anonymous scanning from 8-10 hours with a single scanner during the initial research phase to 5 minutes with the tool. Adding additional RAM and processing power allowed for a scan performance of approximately one port

per second to be scanned by each scanner which is the benchmark noted during initial parallel scanning testing.

7.3 XML parsing and results example

Once results were recorded to output files, a parser was written to pull data that was relevant to the research being done. Coding examples of the XML parser can be seen in appendix B. The parser walks through either a user specified directory or the current directory to find all Nmap xml files to be parsed. A list is created and files to be parsed are added as individual elements in the list. The parser iterates through each file in the list until all files have been parsed. Data elements extracted were IP, port, state (open or closed), service, and hostname. Nmap captured all results for devices regardless of state and when larger scans were done, millions of devices were recorded

```
ip,port,state,service,hostname
502,open,mbap,50.sub-166-239-7.myvzw.com
113,502,open,mbap,113.sub-166-241-164.myvzw.com
16,502,open,mbap,16.sub-166-239-158.myvzw.com
88,502,open,mbap,88.sub-166-241-71.myvzw.com
179,502,open,mbap,179.sub-166-242-119.myvzw.com
179,502,open,mbap,179.sub-166-242-119.myvzw.com
179,502,open,mbap,179.sub-166-242-119.myvzw.com
179,502,open,mbap,179.sub-166-242-119.myvzw.com
179,502,open,mbap,179.sub-166-242-119.myvzw.com
179,502,open,mbap,179.sub-166-242-119.myvzw.com
179,502,open,mbap,179.sub-166-242-119.myvzw.com
90,502,open,mbap,90.sub-166-159-73.myvzw.com
216,502,open,mbap,216.sub-166-163-61.myvzw.com
216,502,open,mbap,216.sub-166-163-61.myvzw.com
216,502,open,mbap,216.sub-166-163-61.myvzw.com
216,502,open,mbap,216.sub-166-163-61.myvzw.com
222,502,open,mbap,222.sub-166-159-155.myvzw.com
```

Figure 12 Parser result example

in the XML files. As such, an argument was added that allowed the user to specify whether they wanted all results to be parsed or only devices with open ports. Reverse DNS lookups were done to find hosts

of devices and potential device owners. Depending on how many scans were running in parallel and how many IP addresses were queued in the scans, significant load was placed on DNS servers. During scans where millions of devices are scanned, using a large public DNS server is preferable as to not overload smaller servers. Results were parsed into CSV format for easier data analysis in

spreadsheets and to load results into a database if necessary. The parser is meant to handle single IP/port combinations for use in analysis with scan comparisons to third party scan data such as Shodan.

8 REAL WORLD IMPACT

After confirmation that the tool and parser worked as expected, scan results were compared against third party scan data. Shodan raw data feeds were available and had been parsed into a database so they were used during verification. Tool verification consisted of identifying three standard port/service combinations that would be scanned and compared against the Shodan data. To create the ranges for scanning, the database was queried for million address ranges that had the highest density of each port that was open.

Ports identified for scanning were 80 (http), 443 (https), and 502 (Modbus). Ports 80 and 443 were selected because they are very common ports and the open port density was high in Shodan (between 50 and 80 percent of the ports in the ranges were open). Port 502 was selected because it is a port commonly used in the ICS industry and would be a good gauge as to how using Tor may negatively impact results depending on how many organizations block Tor by default. Nine ranges were identified for each port, totaling 27 million devices to be scanned. The anonymous port scanning tool found more devices for each port than was identified in Shodan. Largest variances were recorded for port 502, with Shodan identifying around 20 percent of the devices that the tool identified. Port 80 and 443 variances were less, with the tool identifying 2 to 3 percent more devices.

The original hypothesis was that the tool would find less devices than Shodan since there are entities that actively block Tor traffic. Data from the scans suggests that a port scan done through

Tor will often yield more results than a query of Shodan data from a two-week period. Shodan is much faster than the tool that was created so the most efficient use of both sources revolves around identifying IP ranges of interest in Shodan and scanning the ranges with the anonymization tool.

9 NEXT STEPS

9.1 Tool next steps

Anonymous scanning has become much more feasible with the development of the tool that was developed. During scanning, it was noted that when the scanning VMs were locked and the shells were not updating the screen during scans, significantly more scanners could be run in parallel with 8GB of RAM and 4 cores (200 scanners when in the foreground and 600 when in the background). For individual scanner performance, further work may want to be put into running the scans in the background and not opening shells for each Nmap instance.

By the end of data collection, three different scanning VMs were being run and had to be manually accessed to start scans and collect data. Around 10 million devices could be scanned for one port in one day. To more efficiently manage scanning from multiple machines, a master-slave architecture could be employed that would allow a master VM to send commands to multiple slaves that are connected to it, performing the scans and sending the output files to the master so all results are centralized.

10 CONCLUSION

Anonymous port scanning is both possible and can be made scalable. Specific flags need to be set during scanning to ensure that the IP address of the source is not leaked to the target during scanning. The biggest factors that affected scan performance included which Tor exit node was

used, how many scans that were run in parallel, and whether the scans were running in the background or foreground. Next steps in development for the tool include allowing the user run the scans in the background at runtime and implementing a master-slave architecture to allow for more scalable scanning.

11 REFERENCES

- CyberPunk, “The Internet Scanner: ZMap”, <https://n0where.net/the-internet-scanner-ZMap/>
- David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. “Hiding Routing Information” Workshop on Information Hiding, Cambridge, UK, May, 1996.
- Cyrus Farivar, “Report: Rare leaked NSA source code reveals Tor servers targeted”, <https://arstechnica.com/tech-policy/2014/07/report-rare-leaked-nsa-source-code-reveals-tor-servers-targeted/>, July 3, 2014
- Fatemeh Shirazi, Matthias Goehring, Claudia Diaz, “Tor Experimentation Tools”, 2015
- Gordon Lyon, “Nmap Network Scanning” Sunnyvale, CA: Insecure.com, LLC, 2002 p.129
- HP, “Network Node Manager i”, <https://saas.hpe.com/en-us/software/network-node-manager-i-network-management-software>, 2017
- IANA, “Service Name and Transport Protocol Port Number Registry”, <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>, 2017
- Ibrahim Ghafir, Jakub Svoboda, Vaclav Prenosil, “Tor-Based Malware and Tor Connection Detection”, 2014
- John Strand, “Pen Testing through the Tor Network”, <http://securitystreetknowledge.com/?p=283>. January 1, 2011.
- Kuruvilla Mathew, Mujahid Tabassum, Marlene Valerie Lu Ai Siol, “A Study Of Open Ports As Security Vulnerabilities In Common User Computers “, 2014
- Linda Markowsky, George Markowsky, “Scanning for Vulnerable Devices in the Internet of Things”, 2015
- Max Eddy, “The Best VPN Services of 2017”, <http://www.pcmag.com/article2/0,2817,2403388,00.asp>, 2017
- Nmap, “Nmap Reference guide”, <https://nmap.org/book/man-briefoptions.html>, 2017
- Nmap, “TCP/IP Fingerprinting Methods Supported by Nmap”, <https://nmap.org/book/osdetect-methods.html>, 2017
- Proxychains, <http://proxychains.sourceforge.net/howto.html>, 2013
- Rodney Rohrmann, Dr. Mark W. Patton, Dr. Hsinchun Chen, “Anonymous Port Scanning”, 2016
- Shichao, “TCP Connection Management”, <https://notes.shichao.io/tcpv1/ch13/>

Tenable, “Nessus”, <https://www.tenable.com/products/nessus-vulnerability-scanner>, 2017

Tor, <https://check.torproject.org/exit-addresses>, 2017

Whonix, <https://www.whonix.org/>, 2017

Zakir Dumeric et Al, “ZMap: Fast Internet-Wide Scanning and its Security Applications”, 2015

12 APPENDIX A: TOOL SAMPLE CODE

```
#Splits the port range argument into a comma separated list of individual port numbers
portRange = (x.split('-') for x in ports.split(","))
portList = [i for r in portRange for i in range(int(r[0]), int(r[-1]) + 1)]
random.shuffle(portList)

#Calculates the number of ports being scanned and how many will go into each scanner
portCount = int(len(portList))
portSize = portCount // portBuckets
portLeftovers = portCount % portBuckets

#Separates the ports to be scanned into even buckets
ports = []
addItem = 0
for i in range(1, portBuckets + 1):
    if addItem < portLeftovers:
        ports.append(portList[0:portSize + 1])
        del portList[:portSize + 1]
        addItem += 1
    else:
        ports.append(portList[0:portSize])
        del portList[:portSize]

#worker to be threaded, sends nmap commands to terminal
def worker(i, iCount):
    pCount=0
    i = str(i).replace("'", '')[1 : -1]
    for p in ports:
        pCount += 1
        #print("proxychains nmap -Pn -sT " + str(i).replace(',', ' ') + " -p " + str
(p).replace(' ', '')[1 : -1] + " -oX " + "scan" + str(iCount) + "." + str(pCount))
        if anonymous.upper() == "Y":
            os.system("gnome-terminal -e 'bash -c \"service tor start; \""")
            if keepAlive.upper() == "Y":
                os.system("gnome-terminal -e 'bash -c \"proxychains nmap -Pn -sT " + str
(i).replace(',', ' ') + " -p " + str(p).replace(' ', '')[1 : -1] + " -oX " + "scan" + str
(iCount) + "." + str(pCount) + "; exec bash\""")
            else:
                os.system("gnome-terminal -e 'bash -c \"proxychains nmap -Pn -sT " + str
(i).replace(',', ' ') + " -p " + str(p).replace(' ', '')[1 : -1] + " -oX " + "scan" + str
(iCount) + "." + str(pCount) + "; \""")
        else:
            if keepAlive.upper() == "Y":
                os.system("gnome-terminal -e 'bash -c \" nmap -Pn " + str(i).replace
(',', ' ') + " -p " + str(p).replace(' ', '')[1 : -1] + " -oX " + "scan" + str(iCount) +
"." + str(pCount) + "; exec bash\""")
            else:
                os.system("gnome-terminal -e 'bash -c \" nmap -Pn " + str(i).replace
(',', ' ') + " -p " + str(p).replace(' ', '')[1 : -1] + " -oX " + "scan" + str(iCount) +
"." + str(pCount) + "; \""")

#threads = []
iCount = 0
for i in ip:
    t = threading.Thread(target=worker(i, iCount))
    iCount += 1
    # threads.append(t)
    t.start()
```

13 APPENDIX B: XML PARSER SAMPLE CODE

```
import os
import argparse
import xml.etree.ElementTree as ET
import csv

# Get script arguments
parser = argparse.ArgumentParser()
parser.add_argument('-d', '--directory', help='Directory to be parsed',
                    type=str, default = './')
parser.add_argument('-p', '--portState', help='Desired port state to be parsed out',
                    type=str, default = 'a')
args = parser.parse_args()

userDirectory = args.directory
portState = args.portState.upper()

fileList = []

#finds all files created by scanner2.py and puts them in a list
for dirpath, subdirs, files in os.walk(userDirectory):
    for file in files:
        if file.startswith("scan") and not file.endswith(".py") and not file.endswith(".csv"):
            fileList.append(os.path.join(dirpath, file))

scannerCSV = open('scannerCSV.csv', 'w')
csvWriter = csv.writer(scannerCSV)
csvWriter.writerow(['ip','port','state','service','hostname'])

#
for i in fileList:
    print(i)
    tree = ET.parse(i)
    for host in tree.iter('host'):
        ipAddr = host.find('address').attrib.get('addr')
        for hostnames in host.find('hostnames'):
            hostname = hostnames.attrib.get('name')
        for port in host.find('ports'):
            portNo = port.attrib.get('portid')
            state = port.find('state').attrib.get('state')
            service = port.find('service').attrib.get('name')
            if portState == 'A':
                csvWriter.writerow([ipAddr, portNo, state, service, hostname])
            if portState == '0':
                if state == 'open':
                    csvWriter.writerow([ipAddr, portNo, state, service, hostname])
            #print(ipAddr, portNo, state)

scannerCSV.close()
```

13 APPENDIX B: LARGE SCALE SCANNING RESULTS

Table 4 Port 502 results

Port 502					
VM	IP Start	IP End	Shodan Open	Tool Open	Variance
Kali2	166.137.14.128	166.152.80.191	617	2,123	1,506
Kali3	166.152.80.192	166.167.146.255	40	1,972	1,932
Kali3	123.204.129.64	123.219.195.127	374	345	(29)
Kali2	80.12.141.192	80.27.207.255	264	796	532
Kali1	5.1.189.0	5.16.255.63	245	2,560	2,315
Kali3	166.228.156.0	166.243.222.63	172	796	624
Kali1	170.242.53.0	171.1.119.63	48	35	(13)
Kali2	166.243.222.64	167.3.32.127	167	786	619
Kali1	99.46.160.0	99.61.226.63	4	14	10
		Totals	1,931	9,427	7,496

Table 5 Port 80 results

Port 80					
Box	IP Start	IP End	Shodan Open	Tool Open	Variance
Kali 1	23.32.105.0	23.47.171.63	799,691	827,700	28,009
Kali 2	23.200.65.192	23.215.131.255	740,597	812,923	72,326
Kali 3	104.78.225.128	104.94.35.191	734,552	749,300	14,748
Kali 1	23.47.171.64	23.62.237.127	717,552	682,745	(34,807)
Kali 2	104.63.159.64	104.78.225.127	654,482	671,232	16,750
Kali 3	104.94.35.192	104.109.101.255	648,237	670,330	22,093
Kali 1	23.1.228.128	23.17.38.191	618,500	588,935	(29,565)
Kali 2	104.109.102.0	104.124.168.63	597,143	613,891	16,748
Kali 3	23.62.237.128	23.78.47.191	543,664	526,341	(17,323)
Kali 1	23.215.132.0	23.230.198.63	527,603	605,401	77,798
		Totals	6,582,021	6,748,798	166,777

Table 6 Port 443 results

Port 443					
Box	IP Start	IP End	Shodan Open	Tool Open	Variance
Kali 2	23.32.105.0	23.47.171.63	787,858	811,858	24,000
Kali 3	23.200.65.192	23.215.131.255	735,780	810,482	74,702
Kali 1	104.78.225.128	104.94.35.191	718,538	749,546	31,008
Kali 2	23.47.171.64	23.62.237.127	711,066	666,901	(44,165)
Kali 3	104.63.159.64	104.78.225.127	645,791	677,167	31,376
Kali 1	104.94.35.192	104.109.101.255	639,692	676,565	36,873
Kali 2	23.1.228.128	23.17.38.191	608,181	586,107	(22,074)
Kali 3	104.109.102.0	104.124.168.63	593,670	590,843	(2,827)
Kali 1	23.62.237.128	23.78.47.191	534,942	544,133	9,191
		Totals	5,975,518	6,113,602	138,084