Identifying Devices Across the IPv4 Address Space

By

Ryan Jicha

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

2016

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona.

Brief quotations from this thesis 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: Ryan Jicha

APPROVAL BY MASTERS PAPER ADVISOR

This paper has been approved on the date shown below:

_____ 05/06/2016

Dr. Mark Patton Date

Lecturer of Management Information Systems

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Many devices today are internet-enabled. This results in more threat vectors in the IPv4 space. In order to determine the scale of vulnerabilities being introduced to the internet, a new methodology of scanning must be implemented to allow the entire internet to be scanned for types of devices.

Currently, network scans can be connection-oriented, where the connections to ports are tracked, or connectionless, where packets are sent as fast as possible while a separate process listens for server responses. Connection-oriented scanners result in more accurate scanning while connectionless scanners are magnitudes faster.

At the University of Arizona, SCADA devices have been identified based on their banners by using Shodan. Shodan is an online search engine of monthly scan results that are conducted by the sites owner, John Matherly. Not every port is scanned by Shodan; therefore there is a lack of information for identifying all device types based on their port information. In the past, security tools have been combined to improve the accuracy of service scanning, but there are no mentions of combining tools to improve the speed of scans across the entire IPv4 range.

The goal of this research was to create a framework to allow scanning of the entire IPv4 range based on port profiles for device types. This was done by using a connectionless scanner to determine if ports relating to a port profile.

The results from the framework were an improvement of speed from several hours to just three minutes for scanning a device and completing a detailed service scan. After testing the framework on a controlled network, several SCADA devices were found and confirmed to be SCADA using the framework.

1 INTRODUCTION

Many devices are internet-enabled in today's world. Televisions, cameras, washers and dryers, and even door locks are connected to the internet to be controlled by smart phones or monitored while not at home. There are also a large number of devices that control critical infrastructure such as power, water, and gas. The security of all of these devices is of great concern. For example, in the case of smart televisions, these devices have the capability of recording anything said in the room and transfer that data to a third party (Samsung 2016) which can lead to leaked personal information. If vulnerabilities are found in the internet-enabled door locks, these locks could be opened, leaving your personal belongings exposed. Vulnerabilities in critical infrastructure can likewise be abused to damage equipment or deny access to essential resources. Ensuring the security of these devices is of upmost importance. In order to ensure these devices are secure, we must first identify where these devices are.

Most of these devices are connected using Internet Protocol version 4 (IPv4). This version of the protocol allows for a total of nearly 4.3 billion unique IP addresses. Each of these device addresses can communicate with other machines over 65535 ports. By scanning what ports are open on devices and determining the services and service versions running on each of those ports, as well as the operating systems those devices are running, possible vulnerabilities for the devices can be determined. Network scanning is a fundamental way to determine potential vulnerabilities in a network, but with so many ports and IP addresses, current methods do not allow in-depth scans to be done across the entire IPv4 range. The main motivation for this research is to create a way to accurately identify these devices in a reasonable amount of time across the entire IPv4 address space.

2 Network Scanners

2.1 Connection-Oriented Scanners

2.1.1 Summary

In the case of connection-oriented scanners, a connection is made to a single port of a device. This connection is maintained until a response from that device is received. If no response is received, the scanner can assume that the packet was lost in transmission and send a new request to the same port. The scanner does not move on to the next port until either a certain amount of packets are dropped to the destination, or the device responds to the connection. An illustration of how a connection-oriented scanner works can be seen in Figure 1.

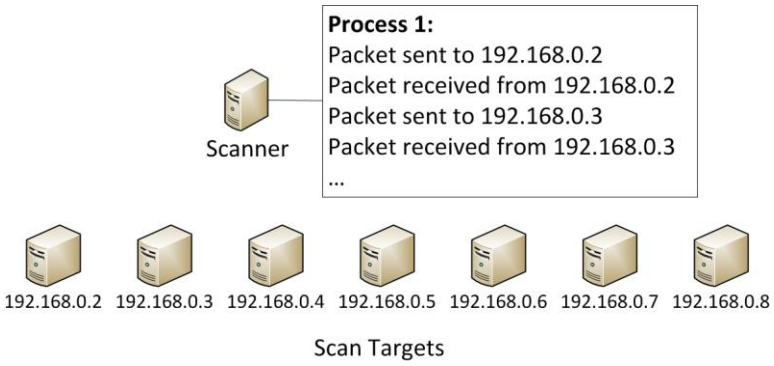


Figure 1- Connection-Oriented Scanner

Note that Figure 1 shows the most basic of connection-oriented scanners, and modern tools have implemented algorithms and parallelization to greatly improve the efficiency of scanning.

2.1.2 Benefits and Weaknesses

There are several benefits associated with connection-oriented scanners. First, they are more accurate than connectionless scanners. Because these scanners maintain the connection state with the target being scanned, any dropped packets can be detected causing the scanner to send another request. Another benefit with connection-oriented scanners is that they are more

developed than connectionless scanners; many of these scanners have algorithms that enhance the speed and accuracy of scans such as Nmap, the de facto standard for network scanning. Due to being more developed, these tools also have additional scanning capabilities, such as service detection and operating system detection, and various types of scans to attempt to bypass common firewall protections. While these strengths greatly improve the accuracy and depth of results returned by scans, they also limit the potential speed of the scanner. A limited number of connections can be stored on a machine; therefore once the maximum number of concurrent scans are in progress, the next scan cannot start until resources are freed by completing a scan. In addition, the service detection and operating system detection take time. In the case of Nmap, these scans compare the return value from the service probes against a set of about 6500 patterns (Lyon). Overall, connection-oriented scanners can be used when accuracy and additional information on a host are required and speed is less important.

Benefits	Weaknesses
More accurate More developed applications More thorough scanning capabilities	Slow scanning speed Speed affected by network latency

Table 1 – Connection-Oriented Scanner Benefits vs Weaknesses

2.2 Connectionless Scanners

2.2.1 Summary

Connectionless scanners, often referred to as “Asynchronous stateless TCP scanners”, do not track any of the connections the scanner is attempting to make. Instead, the scanner runs two separate processes. The first process sends out the connection requests as fast as the network interface card will allow or at some user-defined rate; at the same time, the second process is listening for the responses from the devices. Generally in connectionless scanners, the second process continues running for a predetermined amount of time after the final packet has been

sent by the first process to allow responses to reach the scanner. An illustration of how a connectionless scanner works can be seen in Figure 2.

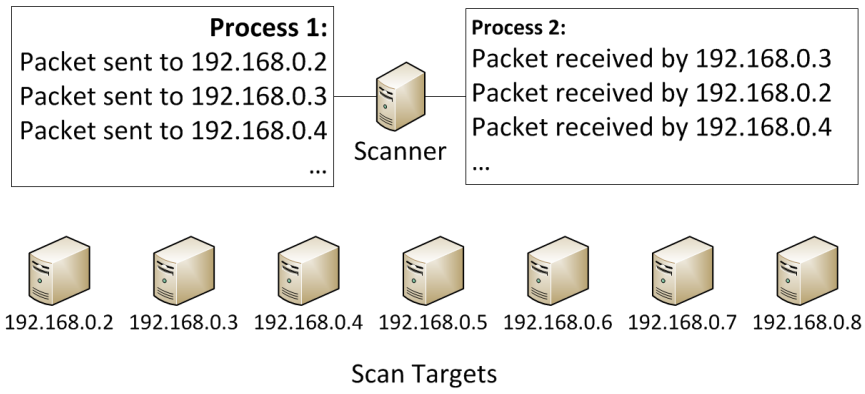


Figure 2 – Connectionless Scanner

2.2.2 Benefits and Weaknesses

One of the main benefits of using a connectionless scanner is the speed at which it can test large ranges of devices. The connectionless scanner can send packets as fast as the NIC, source network, and target networks will allow. Because the second process of connectionless scanners remains open for a specified time after the final packet is sent, the scanners are unaffected by latency; as long as the target hosts respond by the end of the second process, the order in which they respond will not impede the performance of the scanner. With most connectionless scanners, there is very limited information returned; in some cases, a list of IP addresses with their associated ports for open ports is returned. One of the biggest weaknesses of connectionless scanners has to do with their biggest benefit: because they can send packets so quickly, they can easily overwhelm a network resulting in a denial of service.

Benefits	Weaknesses
Extremely fast Unaffected by Latency	Limited information return Can cause a denial of service if not limited

3 LITERATURE REVIEW

Literature in four relevant areas of interest were investigated across a variety of resources available at the University of Arizona. The four areas include Nmap, the most popular connection-oriented scanner and a standard in the industry; Zmap, one of the more popular connectionless scanners; Internet of Things and SCADA, the main device type of interest for the purposes of this research; and internet scanning.

3.1 Nmap

3.1.1 Literature

Paper	Focus	Methods	Data Source	Results
Ghanem (2013)	Application Fingerprinting	Combining Nmap, Amap, and Ettercap	Scanning 20 lab devices	Combination fingerprinting most accurate for service and version identification
Markowsky (2015)	IoT Scanning and Printer Identification	Scanning with Masscan, Nmap and PFT, and searching Shodan	Shodan, Masscan, Nmap, PFT	Vulnerable printers and routers, and servers vulnerable to Heartbleed
Middleton (2012)	Vulnerability Scanning by Country	Scan country IP space with Nessus and Nmap	Nessus and Nmap Scans	Vulnerability patterns by country

Table 2 - Nmap Literature

3.1.2 Key Findings

Nmap is an extremely robust scanner. In Ghanem's study of a comparison of Nmap, Amap, and Ettercap to identify application fingerprints, Nmap had a 90% accuracy of determining the service compared to Amap's 54% accuracy and Ettercap's 50% accuracy. In regards to determining the correct service and version, Nmap was again ahead with a rate of 52% accuracy compared to Amap's 4% accuracy and Ettercap's 32% accuracy; when all of these tools were combined, the service and version accuracy rates were 94% and 87%, respectively (Ghanem and Belaton).

Markowsky uses Nmap to determine the types of devices that are vulnerable to Heartbleed on a university network such as polycom devices. Markowsky also used Nmap to scan a relatively small network for open default printer ports (Markowsky).

In Middleton's conference paper about analyzing vulnerabilities between seven countries, a combination of Nmap and Nessus were used to scan the IP address spaces. Due to the differences in size of the address spaces being compared, an extremely small sample of devices were scanned. 0.001% of the total IP addresses from each country were identified as actively being used at random; 10% of those devices were tested, again randomly, for vulnerabilities, resulting in under 400 devices being scanned in total. Nmap was used to identify the active IP addresses, with Nessus determining the vulnerabilities. In this case, the small sample size was used to ensure an "unbiased sample" and also to create a "manageable sample" for scanning, most likely due to the 7 selected countries making up more than 10% of the IPv4 address space. Because Nmap maintains a connection to the devices it is scanning, it is limited in how quickly it can scan ports, therefore scanning the hundreds of millions of IP addresses with Nmap and Nessus would take a significant amount of time; using a connectionless port scanner would be much quicker and more reasonable (Middleton, Day, and Lallie).

The common factor between Ghanem's and Markowsky's research papers is the use of Nmap for gathering more detailed information about devices on a network. Ghanem showed Nmap to be the most effective for identifying services, and when combining Nmap with other fingerprinting tools, the accuracy can be improved significantly for version identification. In the case of Middleton, Nmap is used to determine active IP addresses at random to then pass along to Nessus for further scanning.

3.2 Zmap

3.2.1 Literature

Paper	Focus	Methods	Data Source	Results
Forbis (2015)	Comparison of Zmap to Shodan	Manual Zmap scanning and Shodan searches	Shodan and Zmap scans	Shodan is more accurate for ports it does scan, but Zmap is more flexible in which ports can be scanned
Durumeric (2013)	Internet-wide network scanning	Optimized probing	Scan Results	IPv4 Scan in 45 minutes on 1GbE connection
Durumeric (2014)	Scanning for Heartbleed	Zmap payloads	Zmap Scans	Vulnerabilities associated with Heartbleed across the IPv4 Range
Zmap Documentation	Usage and Best Practices	Proprietary	Industry Professionals	Zmap Recommendations and Usage

Table 3 - Zmap Literature

3.2.2 Key Findings

Forbis found in a comparison of Zmap and Shodan that one of the main weaknesses of Zmap is the inability to scan multiple ports at once; however, it can easily be scripting by calling multiple Zmap scans. Overall, it was found that Zmap always found at least the same ports as Shodan, and usually found more open ports due to Shodan not scanning those ports at all (Forbis). One issue with this data set is the extremely small sample; Zmap has the ability to send hundreds of thousands of packets per second; by using a sample size of 3 honeypots and a few lab computers, Zmap does not have much of a chance for packet loss.

Durumeric, in his 2013 paper, did a comparison of Zmap and Nmap using a 1 GbE connection. The results show that Zmap is able to scan a single port in the entire IPv4 range in less than one hour with sending two probes per machine, assuming a response was not received for the first probe, for improved accuracy (Durumeric, Wustrow, and Halderman). By setting Nmap to a much faster rate of scanning, it was estimated that it would finish a scan of the entire IPv4 range

in just over 116 days using a maximum of 2 probes per IP address (Durumeric, Wustrow, and Halderman).

In Durumeric's 2014 paper, Zmap was modified to send Heartbeat requests to port 443 of 1% of the devices on the IPv4 range to identify Heartbleed vulnerabilities (Durumeric et al). Zmap can also be modified to send other payloads to ports to identify services or vulnerabilities related to services that may be running on those ports (Durumeric et al).

Forbis's findings of Zmap's accuracy in identifying open ports on a small scale show Zmap as a potential scanning tool for small networks. Durumeric's 2013 paper shows that Zmap can easily be scaled to scan massive IP ranges, including the entire IPv4 range, in a reasonable amount of time with the appropriate hardware and network bandwidth (Durumeric, Wustrow, and Halderman). While Durumeric shows the flexibility of Zmap by detecting Heartbleed among machines in the IPv4 range, additional Zmap modules must be created for every vulnerability or service the user wants to probe for; due to the maturity of Nmap, more than 650 services have already been identified based on 6500 patterns submitted by Nmap users (Lyon).

3.3 Internet of Things and SCADA

3.3.1 Literature

Paper	Focus	Methods	Data Source	Results
Patton (2014)	Finding SCADA Vulnerabilities with Shodan	Vulnerability testing with Python scripts	Shodan	Vulnerable SCADA Systems, printers, and cameras found
OWASP IoT Top 10 (2014)	Internet of Things Vulnerabilities	Proprietary	Industry Experts	List of Top 10 Vulnerabilities
HP Internet of Things Research Study 2015 Report	Internet of Things Vulnerabilities	Automated and manual vulnerability assessment	Analysis of Vulnerabilities of most popular IoT devices	Extremely high rate of vulnerabilities in IoT devices
Shodan.io	Banner Search Engine	IPv4 Internet Scans	Shodan Module Scans	Currently scanned ports

Table 4 - Internet of Things and SCADA Literature

3.3.2 Acronyms

OWASP	The Open Web Application Security Project
IoT	Internet of Things
SCADA	Supervisory Control and Data Acquisition

Table 5 – Internet of Things Acronyms

3.3.3 Key Findings

In Patton's research, the researchers used the Shodan banner search engine to manually search Shodan for SCADA devices based on banners (Patton et al). According to the Shodan API, only 234 ports are scanned by Shodan (Matherly). When comparing Shodan's port list to a list of common SCADA ports found on a Pastebin page posted by an anonymous user, Shodan's crawlers only search for 7 of the 86 ports ("Default SCADA Ports"); the list on Pastebin is most certainly not a comprehensive list of SCADA devices, but shows how sparse the range of ports Shodan is scanning really is in the realm of SCADA. Despite this lack of ports being scanned,

the researchers were still able to find such devices as traffic control devices, printers, Niagara SCADA devices, and webcams.

The Open Web Application Security Project offers yearly top 10 threats for Internet of Things devices. The Top 10 list is not only about vulnerable web code; it also includes vulnerabilities associated with the physical security of the web server, cloud interfaces, encryption, and various other aspects. In addition to ranking the top 10 vulnerabilities for the internet of things, OWASP also publishes the exploitability, detectability, prevalence, and impact of each threat. Finally, OWASP also gives examples of how to find vulnerabilities and how to secure IoT devices from each threat (OWASP).

According to HP's Internet of Things Research Study 2015 Report, in a vulnerability assessment of the 10 most popular IoT devices in the areas of TVs, webcams, home thermostats, door locks, and various other common IoT niches, a large majority of the devices had serious vulnerabilities. Most of these devices included some form of cloud service and all of the devices included mobile applications to remotely control the devices. 80% of the devices collected personal information from the user such as credit card numbers, health information, and addresses. 80% of these devices did not enforce complex password policies; this could leave these devices vulnerable to password brute-force and dictionary attacks, and many of the devices propagated the weak passwords to cloud websites and the mobile applications. 70% of the devices did not encrypt their communications to the internet and local network. This lack of encryption becomes extremely important when considering the personal information collected by these devices and the types of data they may be passing over the network. 60% of the devices had insecure web interfaces. Some of the vulnerabilities associated to these web interfaces included cross-site scripting, poor session management, and weak default credentials. Finally, 60% of the devices

did not use encryption when downloading updates for software and firmware; some of the updates were intercepted and mounted in Linux where the software could be viewed and modified (Hewlett Packard).

3.4 Internet Scanning

3.4.1 Literature

Paper	Focus	Methods	Data Source	Results
Schloesser (2014)	Internet-wide scanning	Scan IPv4 space with Zmap	Results from scans	scans.io
Markowsky (2015)	IoT Scanning and Printer Identification	Scanning with Masscan, Nmap and PFT, and searching Shodan	Shodan, Masscan, Nmap, PFT	Vulnerable printers and routers, and servers vulnerable to Heartbleed
Rapid 7 (2016)	Internet Scanning Recommendations	Proprietary	Industry Experts	Internet-wide scannning best practices and recommendations

Table 6 - Internet Scanning Literature

3.4.2 Key Findings

Schloesser's Blackhat presentation showed the types of data that could be gathered by scanning the internet, including root shells via telnet, listing processes and getting credentials using SNMP, and accessing serial port servers that give network connectivity to network-disabled devices. Some data included on the scans repository at scans.io include HTTPS Heartbleed vulnerability checks using Zmap, FTP banner grabs using Zmap and Zgrab, HTTP get requests using Zmap, and many others (Schloesser).

Markowsky explains a methodology of using Masscan to find Devices vulnerable to Heartbleed and using Nmap to provide more detailed information about the device, as well as verify the existence of the Heartbleed vulnerability; this process was done on a University network, and not the entire IPv4 range, though the methodology could easily be adapted. In addition to scanning

for Heartbleed vulnerabilities, Markowsky also scanned the university network for open printer ports and connected to the printers' RAW port using the PFT tool (L. Markowsky and G. Markowsky).

One common aspect of the three papers mentioned above is the fact that more than one tool is used, each used for what they are especially good at. In the case of Middleton, Nmap's ping scan is used to find devices that are on before sending IP addresses to Nessus. In the case of scan.io's scans, Zmap is used for quick scans to identify open ports and Zgrab is used to grab the banners of ports that are found to be open (Schloesser). Markowsky used Masscan, a connectionless scanner known for its scanning speed, to find devices vulnerable to Heartbleed on the network and used Nmap to verify the vulnerability and gather additional device information (L. Markowsky and G. Markowsky). This methodology of using a connectionless scanner for preliminary port findings and a more accurate, robust scanner for verification will be the basis for the suggested framework.

3.5 Research Gaps

From the literature review, there are two main points that can be gathered. First, Shodan is lacking in SCADA ports. To truly determine what SCADA ports are vulnerable in the IPv4 space, more than a mere 8% of the known SCADA ports need to be scanned. Next, most "mass scanning" research is done on a small subset of the IPv4 range. In the case of Durumeric, 1% of the IP addresses were scanned to show heartbleed trends in the IPv4 address space (Durumeric et al). For Middleton, the vulnerability patterns among countries were based on .0001% of all devices in the address range (Middleton, Day, and Lallie).

3.6 Research Question

The question to be answered by the research presented in the paper is “Can we quickly and effectively identify devices in the IPv4 address space based on their open ports?”

4 Methodology

4.1 Introduction

In order to identify devices based on their open ports across the entire IPv4 range, we first need to determine which ports are open. The most accurate way to do this would be to run probing scans on all devices in the entire IPv4 range; however, this would take far too long. According to Durumeric, a single port with a basic Nmap scan without the service and version detection takes about 116 days to run (Durumeric 2013). Because this is not feasible, a connectionless scanner can be used to scan the entire IPv4 range. Just because a port is open does not mean it is running the standard service that runs on that port. Therefore, before we can confirm that a device with port 80 is truly an HTTP server, the device must be checked. By using a more sophisticated connection-oriented scanner, these services can be verified. In order to scan the entire IPv4 range a combination of the speed of connectionless scanners and the thoroughness of connection-oriented scans is crucial.

4.2 Approach

To address this question, the entire project was split into two main sections. The first section was creating a framework that would combine a connection-oriented scanner and a connectionless scanner. This section was then split into two tasks: researching tools, and then engineering the framework and writing the code. The second section was about testing the newly created

framework. This section was also split into two tasks: benchmarking and analyzing the results of the benchmarks. The overall project methodology can be seen below in Figure 3.

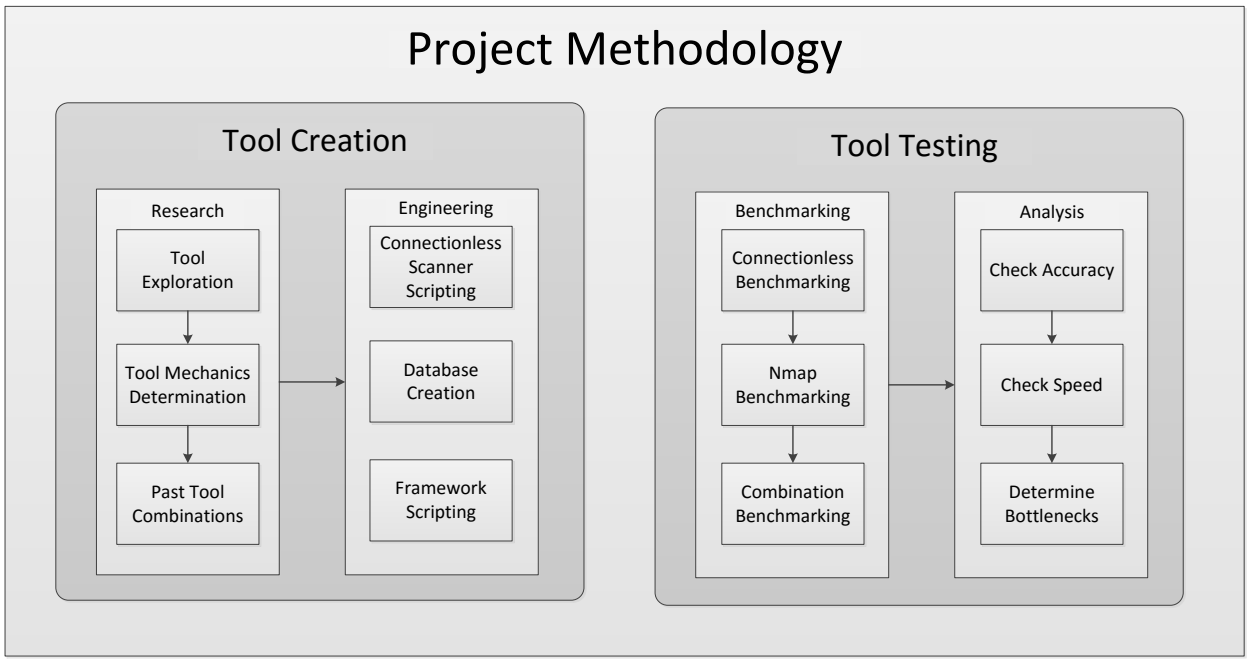


Figure 3 - Project Methodology

4.2.1 Research

The task of researching tools took on three parts. First, several tools needed to be discovered. Through the aforementioned literature, Amap and Nmap were identified as viable connection-oriented scanners; though after analysis of Ghanem's research, Nmap outperforms Amap as a standalone version detection tool (Ghanem and Belaton). For connectionless scanners, Zmap, Masscan, and UnicornScan were compared. Unicornscan was immediately taken off of this list, despite having a variety of useful features, due to the lack of development; the website, www.unicornscan.org is no longer active. This left Zmap and Masscan, both of which would be tested in the tool. The final stage of research was looking at past tool combinations to determine what tools work effectively together. The only literature referencing combining tools was by

Ghanem, which involved combining Amap, Nmap, and Ettercap. No other literature utilized multiple tools to improve the speed of scanning a large network.

4.2.2 Engineering

First, the steps taken to identify devices needed to be planned. The framework would start by selecting a "profile" for which to scan. If searching for web servers, devices running ports 80, 443, 8080, and 8443 are a good place to start. In order to store these profiles and pull them, a MySQL database seems most ideal.

Once this profile is pulled from the database, the connectionless scanner can scan the designated range of IP addresses for those ports, regardless of whether the target is a small network, a large network, or the entire IPv4 address space. These scan results should then be saved in the MySQL database for processing to determine devices that fit the profile we are scanning for. To assist with identifying devices that partially fit the profile, weights should be added to the table storing the port profiles and a weight threshold should be set by the user. With the results from the connectionless scanner, weights can be added among the ports open by IP addresses; if the cumulative weight is greater than the threshold, that device should be considered a potential device and can be added to a list to then be scanned by the connection-oriented scanner.

The final step of the framework's basic functionality is to use a connection-oriented scanner to verify that the devices are running the expected services on the open ports found. Again, these results should be saved to the MySQL database, including the service detection and operating system details from the scan. These results can then be queried to determine which devices truly fit the profile being scanned for. Figure 4 shows the logical design of the framework.

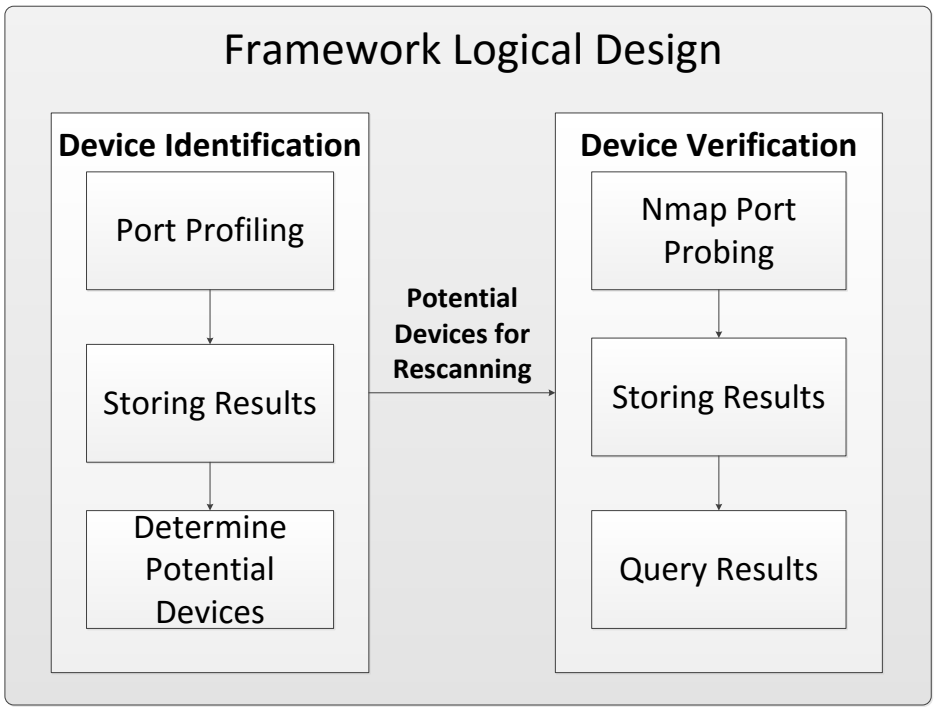


Figure 4 - Framework Logical Design

4.2.3 Benchmarking

The completed framework should then be tested in a controlled environment and compared to using Nmap as a standalone tool. First, the connectionless scanner should be tested for speed and accuracy. Next, Nmap should be tested for speed and accuracy using a scan that should return only the open ports with no service or operating system detection. Then, Nmap's service identification and operating system detection scan should be tested for speed and accuracy. Finally, the framework should be tested as a whole for speed and accuracy.

4.2.4 Test Analysis

With the four tests mentioned in the Benchmarking section, there are two result pairs that should be compared. The results from the connectionless scanner can be compared to using Nmap to get the open ports. The second pair is the Nmap service identification for the entire test environment compared to the framework as a whole. The prior will rely on Nmap to determine if ports are

open before testing for service and operating system, while the latter will rely on the connectionless scanner to inform Nmap of the exact ports that are open for service and operating system detection.

4.3 The Framework

4.3.1 Tool Selection

Nmap was the only clear choice for the connection oriented scanner. It provided much better results for Ghanem as a standalone scanner (Ghanem 2013). For the connectionless scanners, Zmap and Masscan were the two viable options. See Table 7 for a summary of differences between the two scanners.

Zmap	Masscan
Single port scanning	Multiple port scanning
Bitrate limiter	Packet-rate limiter
Used in academic research	Checkpointing

Table 7- Zmap vs Masscan Comparison

Despite Masscan having checkpointing, Zmap was still tested first due to its use in academic research at the University of Michigan. Due to only accepting one port at a time, the framework was scripted to iterate through the list of ports returned by the database when pulling profiles and start Zmap scans in sequence for those ports. Because Zmap's output is a list of IP addresses in a newline separated file, this resulted in as many files as there were ports for the profile being scanned, which resulted in cluttered folders. In addition, issues began arising when calling too many Zmap scans in a row; an error claiming Zmap could not detect the MAC address for the gateway would appear. The error is addressed on zmap's github and the quick fix is to call an arp command prior to running Zmap (Dadrian). This did fix the MAC address issue for the gateway, but introduced a new issue: the eth0 interface would become unavailable randomly after consecutive Zmap scans.

Figure 5 - Masscan Output

At this point, Masscan was tested. Masscan outputs results to a space delimited file. Masscan returns 5 fields: port state, protocol used for the scan, port, IP address, and timestamp. An example of the Masscan output can be seen in Figure 5. Masscan did not encounter any errors during testing, had the same accuracy as Zmap when scanning the test environment, and had similar commands to Nmap; it was therefore chosen over Zmap.

4.3.2 Masscan Speed Limitations

Masscan is limited by three main factors: the operating system, hardware and network bandwidth. Modern operating systems vary greatly in the speed at which they can send packets. Windows and virtualized Linux machines can send packets at a rate of about 300,000 per second (Graham). Standard Linux distributions running on dedicated hardware can reach 1.6 million packets per second (Graham). There is also a driver, PF_Ring, that bypasses the Linux kernel and allows Linux distributions to send packets at the maximum rate the hardware will allow (Graham). Of course, in order to reach these rates, your network interface card and network must be able to handle the traffic.

When scanning the entire IPv4 range, the bandwidth of your target is of very little concern. Masscan randomizes the order of IP addresses and ports that it will scan, which results in very few packets going to a single network at a time. You are much more likely to bring down your own network by scanning too quickly than someone else's network, assuming you are scanning a large IP address range.

With some understanding of Ethernet frames and network transmission and a little math, the number of packets allowed over a specific bandwidth can be solved. First, TCP has 20 bytes of headers. TCP is then encapsulated in IPv4, which is another 20 bytes. These 40 bytes are then encapsulated in an Ethernet frame; however, in order for an Ethernet frame to be valid, it must have a payload of 46. Because Ethernet only uses 18 bytes for its headers, 6 bytes are added as padding to create a valid frame. This results in a 64 byte frame. In order for network transmission to occur, there is a 7 byte preamble, a 1 byte start-of-frame delimiter, and a 12 byte interpacket gap (Beresovsky). When all of this is totaled, it results in 84 bytes, or 672 bits. To convert from bitrate to packets per second, we can divide our bitrate by our packet size in bits. A 1GbE connection can therefore transfer 1,488,095 packets per second.

4.3.3 Nmap Commands

A couple configurations were used throughout the project for Nmap commands depending on the task. A list of relevant Nmap flags can be found in Table 8. When benchmarking to compare Nmap's speed to the connectionless scanner, the –Pn flag was used. When Nmap scans devices without the –Pn flag, Nmap will send an ICMP request to check if the device is on. In many networks, ICMP packets are automatically dropped, which will cause devices to be missed during scans; therefore, the –Pn flag, though making the scan take longer, is necessary to reach the same level of accuracy as the connectionless scanners. The –Pn flag was also used during the probing scans in the framework; if the device returned that the status of a port was open, it is implied that the device is turned on; therefore, there is no reason to waste time with an ICMP request to every device found from the connectionless scanner. The –O and –sV flags were used in both the framework and also the standalone Nmap probing scan. The –oX and –iL flags were

used in all scans to load the IP addresses to scan and to create parsable XML files to import into the database.

Nmap Flag	Function
-O	Enables operating system detection
-sV	Enables service and version detection
-Pn	Skip ping scanning
-oX <filename>	Output as XML to <filename>
-iL <filename>	Use list of hosts found in <filename> as input

Table 8 - Nmap Flags

4.4 Benchmarking Results

For testing purposes, and to avoid overloading the university and lab networks, Masscan was limited to 50,000 packets per second. Two targets were used for testing the framework. The first target was an ESXi machine running on the local university network. The second device was a honeypot set up by another student at the University of Arizona for research purposes. The devices were first scanned with Nmap to determine how many ports were open on the machines. Various sets of scans were run on each machine for the benchmarking process. All scans assumed we wanted to know the states and services of all 65535 ports. In the case of "Nmap Targeted Probing" in the Framework section, this scan only checked the ports found open on each device from the masscan port scan.

The following benchmark results show time in seconds of each of the individual scans. Each benchmark is split into three sections. The first is ESXi, which is the aforementioned machine on the local University network. The Honeypot is a virtualized device hosted on Amazon's AWS. The final set of scans, Combined, is scanning both the ESXi host and the AWS server at the same time. The ESXi machine had 11 ports open while the Honeypot had 8 ports open.

4.4.1 Nmap

Nmap Benchmark			
Nmap Scan	**Nmap Wide Probing**	**Ports Found by Nmap**	**Accuracy**
ESXi			
2331	1449.3	11	100.00%
6541.98	5009.89	11	100.00%
2296.83	12968.44	11	100.00%
5417.21	6719.25	9	81.82%
8317.29	1434.92	11	100.00%
Honeypot			
10713.12	21063.84	8	100.00%
10625.92	10785.12	8	100.00%
10584.15	10755.17	8	100.00%
20429.33	10867.9	8	100.00%
10622.73	46635.75	8	100.00%
Combined			
15952.04	10782.34	19	100.00%
10819.53	21352.27	19	100.00%
10774.57	10979.6	19	100.00%
24664.93	22049.15	19	100.00%
21324.74	10887.76	19	100.00%

Table 9 - Nmap Benchmark

The Nmap Scan column shows the time taken by Nmap to run a standard scan to show open ports with no service or operating system probing. The Nmap Wide Probing shows the time taken by Nmap to probe all 65535 ports. The accuracy refers to the accuracy of the Nmap scan column, as the Nmap Wide Probing was at 100% accuracy for every scan.

4.4.2 Framework

Framework Benchmark				
Masscan Time	**Nmap Targeted Probing**	**Total Framework Speed**	**Ports Found by Framework**	**Accuracy**
ESXi				
12	166.38	178.38	11	100.00%
13	166.3	179.3	11	100.00%
12	167.75	179.75	11	100.00%
13	166.32	179.32	11	100.00%
14	167.72	181.72	11	100.00%
Honeypot				
13	165.65	178.65	8	100.00%
13	164.33	177.33	7	87.50%
14	164.5	178.5	6	75.00%
13	165.7	178.7	5	62.50%
12	164.01	176.01	5	62.50%
Combined				
12	168.07	180.07	19	100.00%
14	167.78	181.78	18	94.74%
13	167.94	180.94	18	94.74%
14	168.02	182.02	19	100.00%
13	167.88	180.88	19	100.00%

Table 10 - Framework Benchmark

4.5 Discussion

Based on the benchmarks, there is a drastic speed increase when combining Masscan and Nmap. The Framework finished its scans in about 3 minutes in all cases. Nmap on the other hand took between 23 minutes and 13 hours to do the same task that was completed by the framework. The accuracy of the framework is significantly lower than Nmap when scanning the AWS honeypots. While this has not yet been investigated, it is possible that the packets are being dropped by AWS on consecutive scans. When scanning both the ESXi machine and the AWS honeypot, the accuracy is greatly improved; most likely due to the fact that the probes were being randomly sent between the two machines, resulting in each device receiving 25,000 packets per second instead of 50,000 packets per second. Nmap also had an anomaly when scanning the local ESXi machine; in just one of the scans, 2 ports were missed by the scans; this still needs investigated.

The reason behind Masscan taking more than 10 seconds was due to the second process continuing for 10 seconds after the final request is sent by the first process. Based on theoretical speeds of Masscan, attainable with a 1GbE connection, a single port can be scanned across the entire IPv4 range in approximately 48 minutes. With the list of 86 SCADA ports, this results in a scan time of 70 hours for the Masscan portion of the framework. It is impossible to guess the duration of the Nmap portion of the scan, as it is completely dependent on the number of ports that are found to be open and the latency between the scanner and those hosts.

By running the scan across the entire IPv4 Range, it also makes the scans much less obvious. Assuming you are scanning a single device at 1.5 million packets per second, that one device would be bombarded by all of those packets. If you split this number of packets randomly among all 4.3 billion devices in the IPv4 space, you are sending .00035 packets per second to each device being scanned. This is much less visible than checking all ports on each device in series.

Using the framework on the University's private /8 IP address space using the SCADA device port profile, several devices were found. After scanning these devices with Nessus, they were confirmed to all be SCADA. Nessus also has an API that would allow the framework to forward IP addresses to Nessus for further verification and vulnerability analysis.

4.6 Future Improvements

To improve upon this framework, bottlenecks need to be addressed. The most notable bottleneck is the Nmap operating system and version detection scans. The framework spends 90% of the time running Nmap scans. There are multiple ways this bottleneck can be addressed. The Nmap scans can be distributed among separate machines using a program such as Dnmap which allows a head node to distribute Nmap queries to child nodes. Another way is finding a way to send Nmap probe requests as connectionless scans similar to how Masscan works. While this would result in lower accuracy, a system could be put in place to attempt probing scans multiple times if no response is received from ports that were found open.

A quick area of improvement for the framework is in the database. In Appendix A, the three tables corresponding to the Nmap scans all contain the same fields regarding CPEs. By normalizing these tables and creating an additional CPE table, redundancy in the data can be avoided.

The purpose of this project was to create a base level framework that others can build on. Some of the modules currently being worked on include a default password testing module for services compatible with Hydra, a Burp Suite module for testing web servers for vulnerabilities, and a banner grabbing module that will parse banners for additional information that can be used for

device identification. Based on Ghanem's findings, Amap could also be incorporated into this framework to improve the service identification.

5 CONCLUSION

By combining scanners, the speed of scanning extremely large networks can be greatly reduced. By looking for devices corresponding to the SCADA device-type profile, SCADA devices were found on a live network in a reasonable amount of time, as opposed to scanning the entire network. Of course, in order to verify that the SCADA devices found were the only SCADA devices running on the network, additional information would be needed; however, when scanning the entire IPv4 range, sacrificing a relatively small amount of accuracy for the ability to scan the entire IPv4 space rather than a very small subset of IP addresses gives a much better representation of the state of the IPv4 space. After some modules have been added to the framework, it can become an invaluable tool for researchers in the cyber security community.

6 REFERENCES

Beresovsky, Eugene. “Size of Empty UDP and TCP Packet.” Stack Overflow. June 1, 2015. http://stackoverflow.com/questions/1846077/size-of-empty-udp-and-tcp-packet. Accessed April 29, 2016

Dadrian. “ARP Issue #19”. Github. August 18, 2013. https://github.com/zmap/zmap/issues/19. Accessed April 29, 2016

“Default SCADA Ports.” Pastebin. http://pastebin.com/EwCibKgc Accessed April 29, 2016

Durumeric et al. “The Matter of Heartbleed.” University of Michigan. 2014

Durumeric, Zakir, Eric Wustrow, and Alex Halderman. “ZMap: Fast Internet-Wide Scanning and its Security Applications.” Proceedings of the 22nd USENIX Security Symposium. August 2013.

Forbis, Samantha. “Integration of ZMap with Shodan for Comprehensive Internet of Things Research.” University of Arizona. 2015.

Ghanem, Waheed and Bahari Belaton. “Improving Accuracy of Applications Fingerprinting on Local Networks Using NMAP-AMAP-ETTERCAP as a Hybrid Framework.” IEEE International Conference on Control System, Computing and Engineering. 2013.

Graham, Robert. “Masscan.” Github. April 19, 2016. https://github.com/robertdavidgraham/masscan.

Hewlett Packard. “Internet of Things Research Study.” November 2015. http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf. Accessed April 29, 2016

Lyon, Gordon. “Nmap Reference Guide”. https://nmap.org/book/man.html. Accessed April 29, 2016

Markowsky, Linda and George Markowsky. “Scanning for Vulnerable Devices in the Internet of Things.” The 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems. September 2015.

Matherly, John. “REST API Dcoumentation”. 2016. https://developer.shodan.io/api. Accessed April 29, 2016

OWASP. “Internet of Things Top Ten”. 2014. https://www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf Accessed April 29, 2016

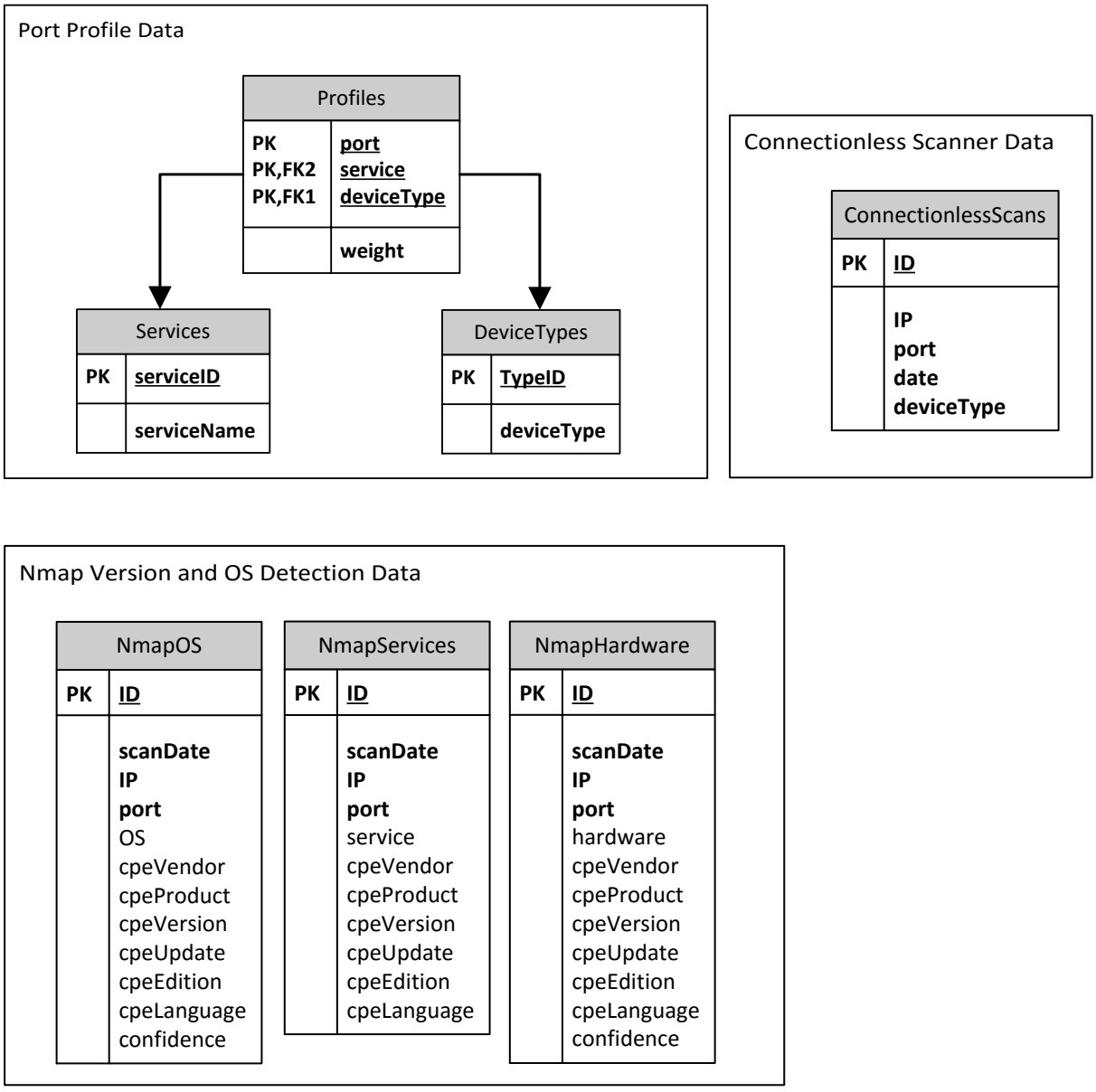
Patton, Mark et al. “Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things”. IEEE Joint Intelligence and Security Informatics Conference. 2014.

Middleton, Raymond, David Day, and Harjinder Lallie. “Global Network Security: A Vulnerability Assessment of Seven Popular Outsourcing Countries”. IEEE International Conference on Green Computing and Communications, Conference on Internet of Things, and Conference on Cyber, Physical and Social Computing. 2012.

Samsung. “Samsung Privacy Statement.” http://www.samsung.com/us/common/privacy.html. Accessed April 29, 2016

Schloesser, Mark. “Internet Scanning: Current State and Lessons Learned”. Blackhat USA Presentation. August 6th, 2014.

7 Appendix A – Database Design



8 Appendix B – Sample Nmap XML Output

```
<host starttime="1460720500" endtime="1460720639">
    <status state="up" reason="user-set" reason_ttl="0"/>
    <address addr="                    addrtype="ipv4"/>
    <hostnames>
        <hostname name='                                    type="PTR"/>
    </hostnames>
    <ports>
        <port protocol="tcp" portid="22">
            <state state="open" reason="syn-ack" reason_ttl="128"/>
            <service name="ssh" product="OpenSSH" version="5.6" extrainfo="protocol 2.0" method="probed" conf="10">
                <cpe>cpe:/a:openbsd:openssh:5.6</cpe>
            </service>
        </port>
        <port protocol="tcp" portid="23">
            <state state="filtered" reason="no-response" reason_ttl="0"/>
            <service name="telnet" method="table" conf="3"/>
        </port>
        <port protocol="tcp" portid="80">
            <state state="open" reason="syn-ack" reason_ttl="128"/>
            <service name="http" product="VMware ESXi Server httpd" hostname=                    method="probed" conf="10"/>
        </port>
        <port protocol="tcp" portid="443">
            <state state="open" reason="syn-ack" reason_ttl="128"/>
            <service name="http" product="VMware ESXi Server httpd" tunnel="ssl" method="probed" conf="10"/>
        </port>
        <port protocol="tcp" portid="8000">
            <state state="open" reason="syn-ack" reason_ttl="128"/>
            <service name="http-alt" method="table" conf="3"/>
        </port>
    </ports>
    <os>
        <portused state="open" proto="tcp" portid="22"/>
        <osmatch name="DD-WRT v24-sp2 (Linux 2.4.37)" accuracy="100" line="34100">
            <osclass type="general purpose" vendor="Linux" osfamily="Linux" osgen="2.4.X" accuracy="100">
                <cpe>cpe:/o:linux:linux_kernel:2.4</cpe>
            </osclass>
        </osmatch>
        <osmatch name="Linux 3.2" accuracy="100" line="77448">
            <osclass type="general purpose" vendor="Linux" osfamily="Linux" osgen="3.X" accuracy="100">
                <cpe>cpe:/o:linux:linux_kernel:3</cpe>
            </osclass>
        </osmatch>
    </os>
    <tcpsequence index="261" difficulty="Good luck!" values="14AF261B,68FD020F,7C223EC1,224F7B93,11712289,101B5A03"/>
    <ipidsequence class="Incremental" values="5515,5516,5517,5518,5519,551A"/>
    <tcptssequence class="none returned (unsupported)"/>
    <times srtt="34940" rttvar="13679" to="100000"/>
```