NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

PASSIVE FINGERPRINTING OF COMPUTER NETWORK RECONNAISSANCE TOOLS

by

Alexander J. Beecroft

September 2009

Thesis Advisor:	James B. Michael
Second Reader:	Raymond R. Buettner

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Passive Fingerprinting of Computer Network Reconnaissance Tools			**5. FUNDING NUMBERS**
6. AUTHOR(S) Alexander J. Beecroft			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words) .

This thesis examines the feasibility of passively fingerprinting network reconnaissance tools. Detecting reconnaissance is a key early indication and warning of an adversary's impending attack or intelligence gathering effort against a network. Current network defense tools provide little capability to detect, and much less specifically identify, network reconnaissance. This thesis introduces a methodology for identifying a network reconnaissance tool's unique fingerprint. The methodology confirmed the utility of previous research on visual fingerprints, produced characteristic summary tables, and introduced the application of TCP sequence number analysis to reconnaissance tool fingerprinting. We demonstrate the use of these methods to fingerprint network reconnaissance tools used in a real-world Cyber Defense Exercise scenario.

14. SUBJECT TERMS Cyberspace defense, network defense, passive fingerprinting, computer network reconnaissance, network scanning, port scanning			**15. NUMBER OF PAGES** 89
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

PASSIVE FINGERPRINTING OF COMPUTER NETWORK RECONNAISSANCE TOOLS

Alexander J. Beecroft
Lieutenant, United States Navy
B.S., U.S. Naval Academy, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION WARFARE SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author:	Alexander J. Beecroft
Approved by:	Professor James B. Michael Thesis Advisor
	Professor Raymond R. Buettner Second Reader
	Dan C. Boger Chairman, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis examines the feasibility of passively fingerprinting network reconnaissance tools. Detecting reconnaissance is a key early indication and warning of an adversary's impending attack or intelligence gathering effort against a network. Current network defense tools provide little capability to detect, and much less specifically identify, network reconnaissance. This thesis introduces a methodology for identifying a network reconnaissance tool's unique fingerprint. The methodology confirmed the utility of previous research on visual fingerprints, produced characteristic summary tables, and introduced the application of TCP sequence number analysis to reconnaissance tool fingerprinting. We demonstrate the use of these methods to fingerprint network reconnaissance tools used in a real-world Cyber Defense Exercise scenario.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ACK:	Acknowledgement Field Significant, TCP Control Flag
AFIT:	Air Force Institute of Technology
CDX:	Cyber Defense Exercise
CND:	Computer Network Defense
DF:	Don’t Fragment Flag
DHCP:	Dynamic Host Configuration Protocol
DNS:	Domain Name System
DoD:	U.S. Department of Defense
DoS:	Denial of Service Attack
ES:	Electronic Warfare Support
FIN:	Finish, No More Data From Sender, TCP Control Flag
GUI:	Graphical User Interface
HTTP:	Hyper Text Transfer Protocol
IA:	Information Assurance
ICMP:	Internet Control Message Protocol
ID:	Identification
IDS:	Intrusion Detection System
IP:	Internet Protocol, v4 for version 4, v6 for version 6
MITM:	Man-In-The-Middle Attack
MSS:	Maximum Segment Size
NOP:	No Operation Performed
NPS:	Naval Postgraduate School
OS:	Operating System
PMTUD:	Path Maximum Transmission Unit Discovery
PSH:	Push Function, TCP Control Flag
RST:	Reset the Connection, TCP Control Flag
SA:	Situational Awareness
SAM:	Surface to Air Missile
SMTP:	Simple Mail Transfer Protocol
RUMINT:	Rumor Intelligence, but the acronym is the name of the program

SP:	Service Pack
SYN:	Synchronize Sequence Numbers, TCP Control Flag
TCP:	Transmission Control Protocol
TTL:	Time To Live
UDP:	User Datagram Protocol
URG:	Urgent Point Field Significant, TCP Control Flag

ACKNOWLEDGMENTS

The author would like to especially thank Professor Bret Michael for his patience, wisdom, guidance, and support throughout this thesis writing process. Thank you to Professor Ray Buettner for guidance as second reader. Thanks to Professor Albert Barreto for providing some critical network equipment to enable the performance of this research. Thank you to Professor Scott Coté and Mr. Andy Shaw for providing NPS data and information on the Cyber Defense Exercise. Thanks to Professor Dave Dittrich for some initial pointers. Thank you very much to LTC Greg Conti for your encouragement and to the SANS Institute for inspiration to pursue a cyber security field. The author would also like to thank his family for their perpetual support and love. Most importantly, the author is grateful to his wife for her steadfast encouragement, support, patience, and love.

Lastly, any errors or deficiencies are solely my own responsibility.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

At the onset of the Vietnam War, an American pilot’s first indication that a Surface-to-Air Missile (SAM) targeted him was to physically see the missile racing toward him. Mounting losses due to SAMs led the United States to develop a plan to incorporate the latest in Electronic Warfare Support (ES) technology into specialized fighter aircraft to identify and target the enemy SAM sites. That program, which came to be known as “Wild Weasel,” underwent incremental development throughout the war. The Wild Weasel aircraft introduced the capability to detect the SAM’s search and track radars, their reconnaissance, directly within the strike group [1]. The Wild Weasels were successful at detecting, classifying, and rapidly engaging the enemy SAM sites. The continued advancement of Wild Weasel and ES technologies throughout the war enabled American fighters to know and react to key information about their enemy’s behavior. Was the enemy radar conducting a general area search? Was it in tracking mode? Did the radar just shift to weapons guidance mode, signaling a missile was about to be fired [2]? This kind of information gave American pilots the edge they needed to turn the tide in the skies over Vietnam and in every air-ground conflict since.

A similar progression of search, track, classify, and attack exists in cyberspace. We can also develop analogous tools to detect and specifically identify, or fingerprint, the tools adversaries use to conduct their reconnaissance leading up to an attack. Just as in the example above, this information could enable computer network defenders to react more quickly and effectively. Detecting reconnaissance is a key early indication and warning of an adversary’s impending attack or intelligence gathering effort against a network. Fingerprinting the reconnaissance tools used can provide important information to create a more robust defense-in-depth posture, creating the opportunity to learn more about an adversary’s behaviors and enable defensive cyber maneuver or counter attack. It also provides for informed decision-making about the appropriate response to initiate. For instance, one could reason about the appropriateness of a response in terms of *jus ad bellum* or *jus in bello*.

This thesis examines the feasibility of passively fingerprinting computer network reconnaissance tools. Several common scanning reconnaissance tools are analyzed, demonstrating a methodology to identify the unique fingerprint characteristics of the tools and describing these fingerprints for each tool. To provide a practical real-world example of using these fingerprints, we analyzed data captured from the Naval Postgraduate School's (NPS) participation in the 2008 Cyber Defense Exercise (CDX) to uniquely identify a network reconnaissance tool used by the attackers against the NPS CDX network. The thesis concludes with recommendations for follow-on research.

A. OVERVIEW

This thesis is organized into four main chapters that provide background information, describe methods, and demonstrate passively fingerprinting computer network reconnaissance tools. This thesis makes extensive use of computer networking protocols and computer security. For the reader desiring additional information on computer networking, the author recommends [3]. For a thorough introduction to computer security, the author recommends [4].

The second chapter is further subdivided into two sections. The first describes the phases of a typical cyber attack. The second identifies where many current reconnaissance detection tools fall short.

The third chapter describes the author's experimentation methodology from setup to data collection and analysis.

The fourth chapter details the results of the experiments for each reconnaissance tool examined. Detailed fingerprint information is shown in the form of visualization illustrations and summary tables.

The fifth chapter provides a demonstration identifying a network reconnaissance tool from data captured by the NPS team in the 2008 Cyber Defense Exercise. The captured data was analyzed with the methodology described in this thesis, which resulted in the specific identification of a reconnaissance tool used during the Cyber Defense Exercise.

The final chapter summarizes the techniques developed and identifies specific areas for further research.

II. THE STRUGGLE FOR CYBERSPACE DOMINANCE

This chapter is divided into two main sections. The first defines the phases of a typical cyber attack. The second describes some of the shortcomings of current computer network defense tools.

A. ANATOMY OF A CYBER ATTACK

Many different authors have described the phases of cyber attacks in different ways. The specific terminology may be slightly different, but categorizing the phases of attack by the physical actions taking place on the network for each phase provides the most universal applicability. We adopt the following three phase description from [4], [5], and [6].

1. Reconnaissance

The first phase of a cyber attack is reconnaissance. It can be further sub-divided into three incremental stages: casing, scanning, and enumeration. For casing, the attacker need only use a web browser to research openly available information. For scanning, the attacker begins to use specialized tools to send packets to identify live hosts and discover information about ports on those hosts. For enumeration, the attacker uses tools to send specifically crafted packets to determine specific services being run on the target machines. Throughout the reconnaissance phase, the attacker refrains from breaking into the potential target. The goal here is to narrow down the target list and query to gain more specific information about those targets.

a. Casing

At the very start of attack planning, the attacker identifies and sets the scope of the target areas to attack. The terms “casing” and “footprinting” are commonly used to describe this stage of reconnaissance. This references an analogy to bank robbers “casing” a target bank by taking note of readily observable information about the bank’s security posture. Similarly, in cyberspace, the attacker does not yet send active probes against the target while casing. Rather, the attacker may use an ordinary web browser to research openly available information that can be useful in the subsequent phases of planning and attack. This includes news articles, browsing the target's Web site, phone numbers, and Internet registry "Who-is" searches to identify the target's Internet Protocol (IP) address space. These searches may reveal the names and phone numbers of key personnel that may be later targeted for phishing schemes, information on the Web site developer and hosting company, physical addresses and specific IP address to target or the limits of the target's IP space.

b. Scanning

Scanning is active reconnaissance. At this stage, the attacker begins overtly sending packets to the target IP address or range of IP addresses (the target's network). The attacker's goal is to determine what machines, or "hosts," are present and reachable on the target network. These scanning probes are akin to search radar scanning the skies, sending out radio energy and listening for the characteristic radio echo return from target aircraft. Just as detecting and recognizing the characteristic patterns of radars provides critical early indication and warning that an adversary is searching for aircraft, the same can be said of cyberspace reconnaissance scanning. Detecting the reconnaissance scans provides the first indication to alert the network defender of a potential impending attack. Observing and classifying the unique fingerprints of these reconnaissance-scanning tools is the purpose of this thesis.

The attacker may perform scans that utilize a number of different protocols. The two most common are Internet Control Message Protocol (ICMP) "pings" and Transmission Control Protocol (TCP) "SYN scans." The specialized reconnaissance tools that the attacker may use can also perform scans using more obscure protocol parameters such as the ICMP timestamp request, User Datagram Protocol (UDP), and TCP "ACK scans," to name a few of the possibilities. Every possible combination of ICMP, UDP, and TCP parameters may be used by the attacker to sneak past network defenses like firewalls and Intrusion Detection Systems (IDS) to minimize the possibility that the defender becomes alerted to the reconnaissance scans.

There are several different kinds of scans based on where the scans originate from and the target destination. A one-to-one scan of multiple UDP or TCP ports is known as a vertical scan. A one-to-many scan of a single UDP or TCP port is known as a horizontal scan. Combining the multiple port aspect of the vertical scan across the numerous targets of the horizontal scan constitutes a scan sweep. See Figure 1 for an illustration of these scanning concepts. More advanced attackers may distribute the origins of their scan across multiple machines under their control; this type of distributed scan makes it challenging for the network defender to identify the true source of the reconnaissance scans. Once the attacker knows the status of the ports, especially the open ports, on the target machines, the next type of reconnaissance begins.

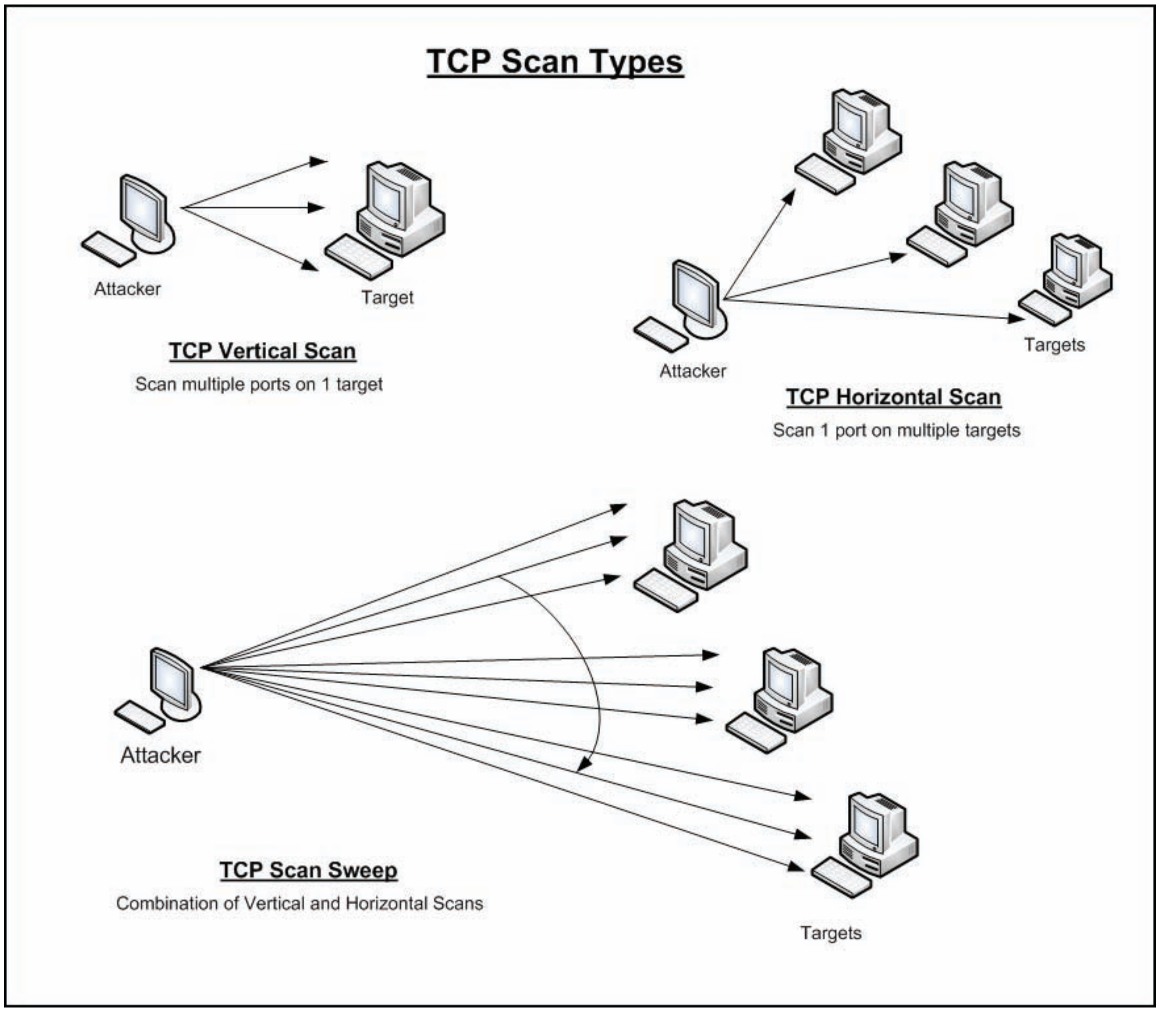


Figure 1. Illustration of Major TCP Scan Types

c. Enumeration

Enumeration is the process of identifying specific services running on the target hosts. It may be performed by the same tool and at the same time as the reconnaissance scan or by a specialized enumeration tool. For the most part, scanning involves sending packets that still properly adhere to the various protocol specifications. For enumeration, the attacker begins sending packets that are intentionally outside the protocol specifications, or mal-formed, in order to test the response from the target. The unique characteristics of each target system's hardware and software implementations enable the attacker to specifically identify these characteristics based on the response from the target to the enumeration packets. Suppose a network administrator tries to be sneaky by hiding a service like a telnet server, which normally resides at TCP port 23, at a different port number, such as 49150. The attacker can still uncover this telnet server through enumeration. Thus, by this point in the reconnaissance process, the attacker knows which target machines can be reached, what ports are open, and specifics such as what operating system (OS) the target is running and what service is running on each port. With this information, the attacker can match the target's characteristics to specific vulnerabilities to attack.

2. Attack

The second phase, the actual attack, involves gaining privileged access (root) on the target machine and then maintaining that access.

a. Gaining Access

The goal of the attack is to gain privileged (root level) access to the target machine so that the attacker usurps control. The attacker may use one or more types of attack, such as buffer overflow, user-password, application-specific, or man-in-the-middle (MITM) attacks. Depending on the attacker's position to access the targeted network, the attacker may be able to simply intercept user authentication credentials passing over the network to log in masquerading as an authorized user. Novice "script kiddies” may use plug-and-play exploitation tools to gain access. More advanced hackers may deploy specifically engineered zero-day exploits to take command of their target while evading network defenses.

b. Maintaining Access

After expending significant effort to find, identify, and infiltrate a target machine, the attacker sets up a means to revisit the target machine and remotely execute code. This can be accomplished via Trojan horses, backdoors, or root kits. The attacker may also employ anti-forensics tools to erase logs and other evidence that the target machine has been compromised.

3. Plunder

With the target machine fully under control of the attacker, the attacker is free to plunder the target, to complete their subversion objective. The attacker may exfiltrate data from the compromised machine, use the machine as a “bot” for denial-of-service (DoS) attacks, use the machine as a springboard to launch attacks further into the target network, or for other purposes.

B. THE CURRENT STATE OF ATTACK DETECTION

1. Intrusion Detection Systems (IDS)

Intrusion detection systems (IDS) are designed to detect unwanted traffic on a computer network. IDS may be deployed to observe, or “sniff,” network traffic across a specific section of the network, the entire network, or on individual host machines. The IDS determines whether or not packets traversing the network are legitimate by comparing them against the signatures of known attack vectors and anomalies, or departures from normally observed baseline traffic [7]. Since the primary purpose of an IDS is to identify malicious packets, block them, and issue alerts, the IDS may be constrained by resource limitations on processing the enormous amount of network traffic data. Consequently, many IDS do not provide extensive information regarding the apparently benign attributes of network reconnaissance scans [5].

SNORT, the most popular open-source IDS and de-facto industry standard for capabilities, provides a functional module to detect reconnaissance port scans called `sfPortscan`. The module generates an alert to notify administrators of a reconnaissance scan by detecting the "negative responses" sent back to the scanner by the target machines on the administrator's network. When the number of these negative responses crosses a certain threshold over a specified time period, the reconnaissance alert is generated. The alert includes basic information such as scan protocol (ICMP, TCP SYN) and type (portscan or portsweep) [8]. The SNORT module does not uniquely identify the scanning tool that was utilized.

2. Passive Fingerprinting

Just as a fingerprint can effectively distinguish an individual from everyone else, the multitude of parameters within all of the protocols that make the Internet work can be used to provide specific information about the systems communicating on the Internet. Passive fingerprinting techniques can be difficult because they cannot rely on "active" measures, like sending out packets to query the target. The principal attributes that must be used to identify the fingerprints reside in the TCP/IP model Layer 3 (Network) and Layer 4 (Transport) headers [9].

Initial efforts to develop a method of passively fingerprinting focused on identifying the operating system (OS) of the target machine. Michal Zalewski's program, p0f, has remained the most successful passive OS fingerprinting program. Based on extensive empirical tests, p0f compares incoming packets against its database of known fingerprints to classify the OS of the computer that sent the packets. Zalewski's work narrowed down the possible combinations of differences in the IP and TCP header fields to the most effective fields to examine to make the OS determination [9]. The fingerprint determination is made based on static data from each connection to the machine running p0f. The fingerprint entry format from p0f version 2.0.8 is as follows [10]:

```
# Fingerprint entry format:
#
# wwww:ttt:D:ss:OOO…:QQ:OS:Details
#
# wwww      –window size (can be * or %nn or Sxx or Txx)
# “Snn” (multiple of MSS) and “Tnn” (multiple of MTU) are allowed.
# ttt       –initial TTL
# D         –don’t fragment bit (0 – not set, 1 – set)
# ss        –overal SYN packet size (* has a special meaning)
# OOO       –option value and order specification (see below)
# QQ        –quirks list (see below)
# OS        –OS genre (Linux, Solaris, Windows)
# details   –OS description (2.0.27 on x86, etc)
```

Tools like p0f can be very effective at examining socket data to make an accurate OS determination. However, it has been shown that a tool like p0f can be fooled by altering these fingerprint parameters in an OS kernel [11]. Consequently, it is important to include dynamic behavioral parameters to the fingerprints. This thesis moves beyond OS fingerprinting to determine evidence of static and dynamic fingerprint parameters for network reconnaissance scanning tools. It is not the primary function of p0f, but this particular tool provides a rudimentary ability to identify a reconnaissance scan performed using Nmap. The passive fingerprinting techniques from [9] and [10] formed the starting point for the analysis developed in this thesis.

3. Network Data Visualization

Most captured network data is still displayed in a textual format, which can make it difficult to rapidly analyze or acquire situational awareness (SA) on the network. Visualization techniques show great promise to improve the network defender’s ability to quickly gain and maintain SA on their network [12]. It is therefore possible to develop visual fingerprints that could alone provide unique fingerprints of network reconnaissance tools [13]. The research conducted in [13] provided the second main approach to analysis that is further developed in this thesis. While these visualizations are a powerful additional tool for network defenders, they are vulnerable to occlusion attacks that force the network defender into the same tedious analysis process as the current textual based systems [14]. This thesis pairs some visualization techniques with an analysis of the specific textual hard network data to present a more descriptive fingerprint.

4. The "Advanced Persistent Threat"

Expert attackers, such as state-sponsored information warriors, have progressed to the point that sometimes the first indication that a network has been compromised is detecting the exfiltration or corruption of data. An organization may be suddenly undercut by a competing firm utilizing technology that looks suspiciously exactly like the defender's own proprietary technology [15]. Expert attackers can infiltrate networks and establish a pervasive presence capable of rapidly reacting to defensive tactics to continue exfiltrating data; aptly termed by one industry leading company as the "Advanced Persistent Threat" [16]. Consequently, it is of vital importance to develop the best early indication and warning that a network is being targeted by detecting and identifying the adversary while the attacker is still in the reconnaissance phase of its attack.

III. EXPERIMENTATION METHODOLOGY

A. ISOLATED LABORATORY TEST NETWORK SETUP

An isolated laboratory test network was set up to conduct experiments and collect data on the characteristics and behavior of several common network reconnaissance tools. The experimental network consists of dedicated machines for simulating the attacker, the target, and data-capture machine. These computers are linked via standard Category 5e Ethernet cable through a commercial grade 3Com 3300 network switch. Both the attacker and target were loaded to dual-boot Microsoft Windows XP Service Pack 3 (XP SP3) and Ubuntu Linux 9.04 (32-bit version, Linux kernel 2.6.28). A dedicated Dynamic Host Configuration Protocol (DHCP) server was installed to assign IP addresses and to simulate a typical enterprise network configuration. This DHCP server also provided a backup network data-capture capability and allowed for expansion of the test network to include additional target hosts.

A diagram of the isolated laboratory test network can be seen in Figure 2.

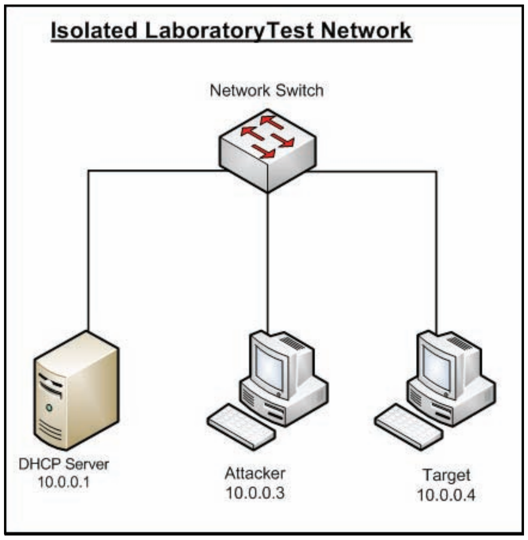


Figure 2. Experimentation Network Diagram

B. DATA COLLECTION

1. Selection and Configuration of Software Tools

The different tools to be analyzed were selected from a number of different sources, including network security texts, such as [4]–[6], [13], [14], [17]–[19], and a Web site [20].

Once installed and initial tests of each tool were attempted, the group of test programs was narrowed down to focus on the default patterns of TCP reconnaissance scanning tools. It was discovered that by default, Windows XP SP3 does not reply to ICMP messages. Many of the reconnaissance tools, also by default, verify which target hosts are reachable by starting scans with an ICMP ping to the target; and only proceeding with the TCP port scan against the target machines that replied to the ICMP ping. Without a response from the target Windows XP SP3 machine, the follow-on TCP port scans would not occur. To keep the evaluation of the reconnaissance tools limited to default behavior as much as possible, the target machine's Windows XP firewall was set to allow ICMP responses. Some reconnaissance tools can be configured to conduct scans without a preparatory ICMP ping. However, the goal was to observe each tool's default behavior to ascertain its unique fingerprint. Furthermore, many networks have been configured to drop all ICMP messages from outside the network as a defensive tactic to prevent an ICMP DoS Attack caused by a flood (i.e., large volume) of ICMP packets [4], [19].

For Windows XP Service Pack 2 and beyond, Windows no longer permits default raw socket support for programs, even when running as an administrator. Thus, many of the venerable older Windows scanning tools do not function properly with just the default installation. These Windows scanning tools, like Superscan by Foundstone, date back as far as 2002, with little or no continued development on the tools since that time [21]. Consequently, the scanning tools evaluated in this thesis are all Linux based with the exception of Zenmap.

An additional setting modification was made to the Windows XP firewall to permit connections to TCP ports 25 and 80. These two ports correspond to Simple Mail

Transfer Protocol (SMTP, better known as email) and HyperText Transfer Protocol (HTTP, better known as the application backbone of the World Wide Web) [22]. Thus, the reconnaissance tools could scan the target in their default mode and be assured of discovering some information about the target's TCP ports.

2. Data Capture

The target machine maintained a Windows XP SP3 environment to simultaneously capture the reconnaissance scan packets with Wireshark [23] and display real-time network data visualization with RUMINT [24]. New packet capture ".pcap" data files were created in Wireshark for each test run of each reconnaissance tool. At least three test runs were conducted for each tool in the default configuration. An additional three test runs were conducted for each tool to specifically target TCP Ports 80, 139, and 65000. These ports were selected to create a narrow data set showing a listening but closed port, an open port, and a non-listening closed port. Lastly, a test was conducted to observe each tool perform a scan against the entire TCP Port range (i.e., from 0 to 65,535). However, the principle objective was to observe the default TCP scan. The limited and complete port range tests served to verify that the reconnaissance tool acted in a similar manner regardless of the number of ports being scanned.

C. DATA ANALYSIS

The purpose of analyzing the data captured from these reconnaissance scans is not to reverse engineer the tools. Rather, the captured data is analyzed to observe the specific static and dynamic characteristics of the tools to create unique fingerprints to identify each reconnaissance tool. For the purpose of developing these unique fingerprints, it is not necessary to fully explain all of the behaviors of the different reconnaissance tools. As will be shown, the deeper the analysis delves, the more questions arise that complicate the process. It is adequate to identify the static and dynamic characteristics to develop the fingerprint; any further information and analysis may be beneficial, but is beyond the scope of this research.

The next section describes the particular TCP/IP packet header fields that are of interest to developing the fingerprints.

1. Packet Header Fields

The following fields from the IP and TCP headers form the basic parameters to evaluate to identify a particular tool's unique fingerprint. Additional information regarding the IP and TCP header fields can be found in [25], [26], [18], and [9].

a. IP Total Length

Total Length is a 16-bit IP field stating the entire length of the datagram, measured in octets. This is a simple value, but it does help specify which reconnaissance tool or OS sent the packet.

b. IP Identification

The IP Identification field is a randomized 16-bit value assigned by the sender to aid in assembling fragments of a datagram. It should uniquely identify the particular datagram from all others originated by the sender.

c. IP Flags

The IP Flags consist of 3 bits. The first bit is reserved and should always be zero. The second bit is the Don't Fragment (DF) bit, specifying whether or not a datagram should be fragmented or just dropped if it reaches a network segment that cannot handle the length of the datagram. Most OSs set DF to 1, meaning Don't Fragment, in order to implement a form of Path Maximum Transmission Unit Discovery (PMTUD). This scheme has the sender adjust the maximum size of the packets it sends to minimize the possibility of the packets getting fragmented. The last bit is the More Fragments flag, signifying there are additional fragments to be reassembled following the current packet fragment. The first and last IP flag bits do not play a large role in scanning, but may be used extensively in enumeration.

d. IP Time to Live (TTL)

The TTL field is 8 bits, allowing a range from 0–255. It prevents packets from endlessly circulating the Internet. Each time a packet gets passed on by a router, the TTL value is decremented. When the TTL value reaches zero, the packet is dropped.

Different OS manufacturers set this value differently, providing a good indicator of the source machine’s OS. For example, UNIX variants have an initial TTL=255, Windows machines have an initial TTL=128, and Linux variants have an initial TTL=64. Due to the efficiency of the Internet’s routing, it is rarely necessary for a packet to require more than 15 hops to reach its destination. This allows an estimation of the sender’s distance in router hops from the recipient.

e. IP Source Address

The IP Source Address is the 32-bit IP address of the source of the datagram, normally displayed in a dotted decimal format. For the experiment’s reconnaissance scans, the Source IP is 10.0.0.3.

f. IP Destination Address

The IP Destination Address is the 32-bit IP address of the destination of the datagram, also displayed in a dotted decimal format. For the experiment’s reconnaissance scans, the Destination IP is 10.0.0.4.

g. TCP Source Port

The TCP Source Port is a 16-bit value offering a range between 0 and 65,535. The TCP Port numbers are divided into three ranges, ports 1–1023 are “well-known,” ports 1024–49151 are “registered,” and ports 49152–65535 are “dynamic.” The well-known ports offer a common frame of reference for popular services like email at port 25 and http at port 80. Since each vendor’s software implements the TCP/IP stack slightly differently, the selection of the TCP Source Port number can also provide some clues for identifying the OS of the sender.

h. TCP Destination Port

The TCP Destination Port is also a 16-bit value, sharing similar properties as the Source Port. Since the most common services are running on a well-known port, most of the reconnaissance tools send the majority of their scanning packets to the well-known ports.

i. TCP Sequence Number

The TCP Sequence Number is 32 bits long with possible values between 0–4,294,967,295. It plays a role similar to the IP ID field to uniquely identify a connection, but more importantly it also forms a method of synchronizing and accounting for the data transferred between two machines in a TCP conversation along with the TCP Acknowledgement Number [26]. The original TCP specification contained a weakness in TCP Sequence Number implementation that allowed an attacker to hijack TCP connections. Corrections were made to OS kernels to introduce pseudo-randomness into the TCP Sequence Number generation, with varying results [27], [28]. Variations in TCP Sequence Number behavior were revealed to be one of the best ways to determine the dynamic unique fingerprint of the scanning tools. It could also be very useful to uncover reconnaissance scans that are conducted slowly over a longer period of time in an attempt to evade IDSs. Unfortunately, TCP Sequence Number analysis also proved to be one of the most time-consuming parameters to analyze.

j. TCP Acknowledgement Number

The TCP Acknowledgement Number is also 32 bits long. If the "ACK" Control Flag is set, this field contains the value of the next sequence number the sender of the segment is expecting to receive. It completes the method of accounting for the amount of data being exchanged between the two computers at each step in the communication process as described above. By default, all of the reconnaissance tools tested performed "SYN" scans. Consequently, the TCP Acknowledgement Number field was always zero and inconsequential to fingerprint analysis. However, many of the tools have the functionality to send "ACK" scans, at which point monitoring the TCP Acknowledgement Number becomes pertinent.

k. TCP Control Flags

The TCP Control Flags are a 6-bit field, from left to right: URG (urgent pointer field significant), ACK (acknowledgement field significant), PSH (push function), RST (reset the connection), SYN (synchronize sequence numbers), and FIN (no more data from sender). As mentioned above, by default, all the reconnaissance scanners tested performed “SYN” scans. The incoming SYN packets are treated like legitimate attempts to initiate a connection by the receiving machine. However, the scanner has no intention of completing the synchronization of the “Three-way TCP handshake” to establish normal communications between the scanner and target. Consequently, if the target TCP port is open, the target still responds with the proper “SYN+ACK” packet, but the scanning program has moved on to the next target port. The scanner remembers that the port that replied with “SYN+ACK” is open to provide this important information to the attacker when the scan is complete. However, since the scanning program is often operating independently of its native OS kernel’s TCP/IP stack, when the scanner receives the “SYN+ACK” packet, the kernel responds with a “RST” packet. The kernel had no knowledge of the “SYN” packet being sent, so from the kernel’s perspective, the “SYN+ACK” it received does not belong to any connection and is therefore “RST.”

If the target port is closed, with no service listening on it, the target machine responds to the “SYN” packet with a “RST” packet. This still provides valuable information to the scanner. Consequently, many systems and firewalls have been configured to simply send no response when a “SYN” packet is received at a closed port [22]. When the attacker conducts a vertical scan as described in the previous chapter, there are considerably more closed ports than open ports. These negative responses are the primary indicators to an IDS of reconnaissance scanning activity [7].

Various combinations of TCP Control Flags are frequently used in the enumeration stage of reconnaissance. Certain combinations may also enable the attacker to pass through a firewall with a weak rule set and still conduct its reconnaissance scan.

l. TCP Window

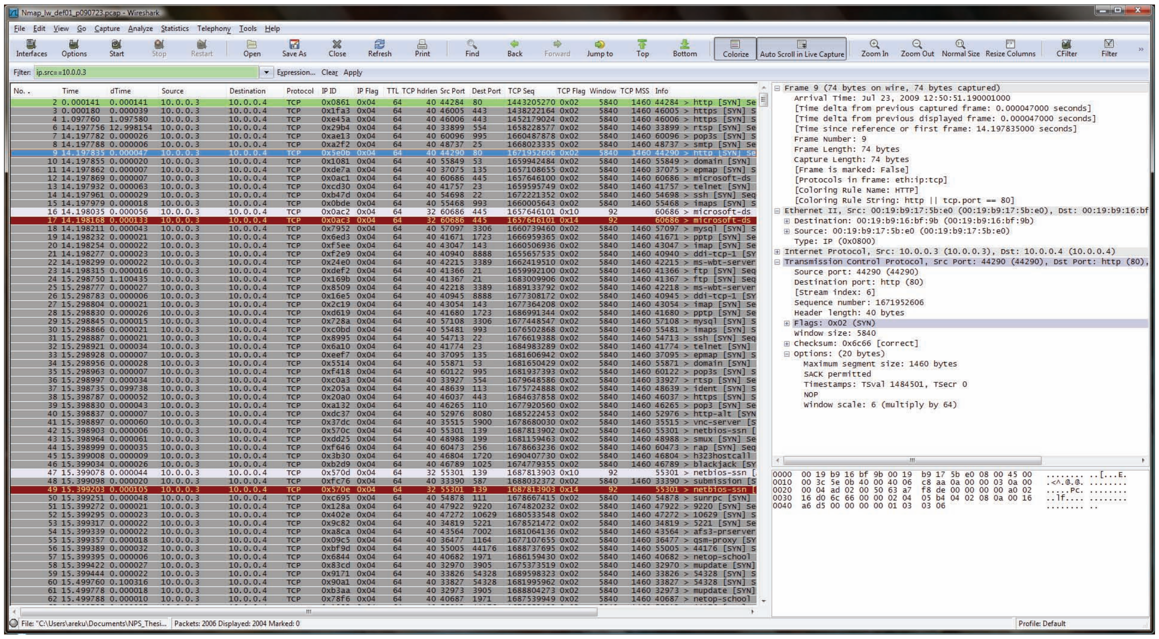
The TCP Window is a 16-bit field that represents the number of data octets, beginning with the one indicated in the acknowledgement field, which the sender of the segment is able to accept. It is significant to fingerprinting whether it is a constant value, a fixed set of values, or a multiple of the Maximum Segment Size (MSS) from the TCP Options field.

m. TCP Options

TCP Options are placed at the end of the TCP header and are multiples of 8 bits in length. The order and selection of information included in the TCP Options also helps specify the sender's OS. Windows machines exhibit the following order: MSS, no operation performed (NOP, a value of 0x01 in the packet), NOP, and Selective ACK Permitted. Linux machines output the following order: MSS, Selective ACK Permitted, Timestamp, NOP, and Window scale [9]. Similar to these OS TCP Options differences, the reconnaissance tools also differ in use of the TCP Options field.

2. Wireshark Filter

Packets on the isolated laboratory test network were captured as ".pcap" data files with the program Wireshark running on the target machine. Separate files were created for each reconnaissance scan and saved so that they could be exported to other machines for follow-on analysis. Wireshark was used to filter the displayed data to only include the relevant packets sent from the attacker machine, 10.0.0.3 [23]. To facilitate data export for further analysis in a spreadsheet, custom column headings in the packet list view pane were created. This enabled the detailed packet header fields to be exported into comma-delimited value fields in ".csv" files. Figure 3 shows an example screenshot from Wireshark. The author found that using a widescreen monitor to display packet information made a more effective use of the visual area than the default Wireshark configuration.


Nmap_lw_def01_p090723.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Interfaces Options Start Stop Restart Open Save As Close Refresh Print Find Back Forward Jump to Top Bottom Colorize Auto Scroll in Live Capture Zoom In Zoom Out Normal Size Resize Columns CFilter Filter
Expression... Clear Apply
Profile: Default

Filter: ip.src==10.0.0.3

```
No.  Time      dTime     Source    Destination Protocol IP ID  IP Flag TTL TCP hdrlen Src Port Dest Port TCP Seq    TCP Flag Window TCP MSS Info
  2 0.000141  0.000141  10.0.0.3  10.0.0.4  TCP  0x0861 0x04  64  40 44284 80     1443205270 0x02  5840  1460 44284 > http [SYN] Se
  3 0.000180  0.000039  10.0.0.3  10.0.0.4  TCP  0x1fa3 0x04  64  40 46005 443    1438222164 0x02  5840  1460 46005 > https [SYN] S
  4 1.097760  1.097580  10.0.0.3  10.0.0.4  TCP  0xe45a 0x04  64  40 46006 443    1452179024 0x02  5840  1460 46006 > https [SYN] S
  6 14.197756 12.998154 10.0.0.3  10.0.0.4  TCP  0x29b4 0x04  64  40 33899 554    1658228577 0x02  5840  1460 33899 > rtsp [SYN] Se
  7 14.197782 0.000026  10.0.0.3  10.0.0.4  TCP  0xae13 0x04  64  40 60096 995    1660487878 0x02  5840  1460 60096 > pop3s [SYN] S
  8 14.197788 0.000006  10.0.0.3  10.0.0.4  TCP  0xa2f2 0x04  64  40 48737 25     1668023335 0x02  5840  1460 48737 > smtp [SYN] Se
  9 14.197835 0.000047  10.0.0.3  10.0.0.4  TCP  0x5e0b 0x04  64  40 44290 80     1671952606 0x02  5840  1460 44290 > http [SYN] Se
 10 14.197855 0.000020  10.0.0.3  10.0.0.4  TCP  0x1081 0x04  64  40 55849 53     1659942484 0x02  5840  1460 55849 > domain [SYN]
 11 14.197862 0.000007  10.0.0.3  10.0.0.4  TCP  0xde7a 0x04  64  40 37075 135    1657108655 0x02  5840  1460 37075 > epmap [SYN] S
 12 14.197869 0.000007  10.0.0.3  10.0.0.4  TCP  0x0ac1 0x04  64  40 60686 445    1657646100 0x02  5840  1460 60686 > microsoft-ds
 13 14.197932 0.000063  10.0.0.3  10.0.0.4  TCP  0xcd30 0x04  64  40 41757 23     1659595749 0x02  5840  1460 41757 > telnet [SYN]
 14 14.197961 0.000029  10.0.0.3  10.0.0.4  TCP  0xb47d 0x04  64  40 54698 22     1672221352 0x02  5840  1460 54698 > ssh [SYN] Seq
 15 14.197979 0.000018  10.0.0.3  10.0.0.4  TCP  0x0bde 0x04  64  40 55468 993    1660005643 0x02  5840  1460 55468 > imaps [SYN] S
 16 14.198035 0.000056  10.0.0.3  10.0.0.4  TCP  0x0ac2 0x04  64  32 60686 445    1657646101 0x10    92       60686 > microsoft-ds
 17 14.198168 0.000133  10.0.0.3  10.0.0.4  TCP  0x0ac3 0x04  64  32 60686 445    1657646101 0x14    92       60686 > microsoft-ds
 18 14.198211 0.000043  10.0.0.3  10.0.0.4  TCP  0x7952 0x04  64  40 57097 3306   1660739460 0x02  5840  1460 57097 > mysql [SYN] S
 19 14.198232 0.000021  10.0.0.3  10.0.0.4  TCP  0x6ed3 0x04  64  40 41671 1723   1666959365 0x02  5840  1460 41671 > pptp [SYN] Se
 20 14.198254 0.000022  10.0.0.3  10.0.0.4  TCP  0xf5ee 0x04  64  40 43047 143    1660506936 0x02  5840  1460 43047 > imap [SYN] Se
 21 14.198277 0.000023  10.0.0.3  10.0.0.4  TCP  0xf2e9 0x04  64  40 40940 8888   1655657535 0x02  5840  1460 40940 > ddi-tcp-1 [SY
 22 14.198299 0.000022  10.0.0.3  10.0.0.4  TCP  0x24e0 0x04  64  40 42215 3389   1662419510 0x02  5840  1460 42215 > ms-wbt-server
 23 14.198315 0.000016  10.0.0.3  10.0.0.4  TCP  0xdef2 0x04  64  40 41366 21     1659992100 0x02  5840  1460 41366 > ftp [SYN] Seq
 24 15.298750 1.100435  10.0.0.3  10.0.0.4  TCP  0x169b 0x04  64  40 41367 21     1683009906 0x02  5840  1460 41367 > ftp [SYN] Seq
 25 15.298777 0.000027  10.0.0.3  10.0.0.4  TCP  0x8509 0x04  64  40 42218 3389   1689133792 0x02  5840  1460 42218 > ms-wbt-server
 26 15.298783 0.000006  10.0.0.3  10.0.0.4  TCP  0x16e5 0x04  64  40 40945 8888   1677308172 0x02  5840  1460 40945 > ddi-tcp-1 [SY
 27 15.298804 0.000021  10.0.0.3  10.0.0.4  TCP  0x2c19 0x04  64  40 43054 143    1677364208 0x02  5840  1460 43054 > imap [SYN] Se
 28 15.298830 0.000026  10.0.0.3  10.0.0.4  TCP  0xd619 0x04  64  40 41680 1723   1686991344 0x02  5840  1460 41680 > pptp [SYN] Se
 29 15.298845 0.000015  10.0.0.3  10.0.0.4  TCP  0x728a 0x04  64  40 57108 3306   1677448547 0x02  5840  1460 57108 > mysql [SYN] S
 30 15.298866 0.000021  10.0.0.3  10.0.0.4  TCP  0xc0bd 0x04  64  40 55481 993    1676502868 0x02  5840  1460 55481 > imaps [SYN] S
 31 15.298887 0.000021  10.0.0.3  10.0.0.4  TCP  0x8995 0x04  64  40 54713 22     1676619388 0x02  5840  1460 54713 > ssh [SYN] Seq
 32 15.298921 0.000034  10.0.0.3  10.0.0.4  TCP  0x6a10 0x04  64  40 41774 23     1684983289 0x02  5840  1460 41774 > telnet [SYN]
 33 15.298928 0.000007  10.0.0.3  10.0.0.4  TCP  0xeef7 0x04  64  40 37095 135    1681606942 0x02  5840  1460 37095 > epmap [SYN] S
 34 15.298956 0.000028  10.0.0.3  10.0.0.4  TCP  0x5514 0x04  64  40 55871 53     1681650429 0x02  5840  1460 55871 > domain [SYN]
 35 15.298963 0.000007  10.0.0.3  10.0.0.4  TCP  0xf418 0x04  64  40 60122 995    1681937393 0x02  5840  1460 60122 > pop3s [SYN] S
 36 15.298997 0.000034  10.0.0.3  10.0.0.4  TCP  0xc0a3 0x04  64  40 33927 554    1679648586 0x02  5840  1460 33927 > rtsp [SYN] Se
 37 15.398735 0.099738  10.0.0.3  10.0.0.4  TCP  0x205a 0x04  64  40 48639 113    1675724888 0x02  5840  1460 48639 > ident [SYN] S
 38 15.398787 0.000052  10.0.0.3  10.0.0.4  TCP  0x20a0 0x04  64  40 46037 443    1684637858 0x02  5840  1460 46037 > https [SYN] S
 39 15.398830 0.000043  10.0.0.3  10.0.0.4  TCP  0xa132 0x04  64  40 46265 110    1677920560 0x02  5840  1460 46265 > pop3 [SYN] Se
 40 15.398837 0.000007  10.0.0.3  10.0.0.4  TCP  0xdc37 0x04  64  40 52976 8080   1685222453 0x02  5840  1460 52976 > http-alt [SYN
 41 15.398897 0.000060  10.0.0.3  10.0.0.4  TCP  0x37dc 0x04  64  40 35515 5900   1678680030 0x02  5840  1460 35515 > vnc-server [S
 42 15.398903 0.000006  10.0.0.3  10.0.0.4  TCP  0x570c 0x04  64  40 55301 139    1687813902 0x02  5840  1460 55301 > netbios-ssn [
 43 15.398964 0.000061  10.0.0.3  10.0.0.4  TCP  0xdd25 0x04  64  40 48988 199    1681159463 0x02  5840  1460 48988 > smux [SYN] Se
 44 15.398999 0.000035  10.0.0.3  10.0.0.4  TCP  0xf646 0x04  64  40 60473 256    1678663236 0x02  5840  1460 60473 > rap [SYN] Seq
 45 15.399008 0.000009  10.0.0.3  10.0.0.4  TCP  0x3b30 0x04  64  40 46804 1720   1690407730 0x02  5840  1460 46804 > h323hostcall
 46 15.399034 0.000026  10.0.0.3  10.0.0.4  TCP  0xb2d9 0x04  64  40 46789 1025   1674779355 0x02  5840  1460 46789 > blackjack [SY
 47 15.399078 0.000044  10.0.0.3  10.0.0.4  TCP  0x570d 0x04  64  32 55301 139    1687813903 0x10    92       55301 > netbios-ssn [
 48 15.399098 0.000020  10.0.0.3  10.0.0.4  TCP  0xfc76 0x04  64  40 33390 587    1688032372 0x02  5840  1460 33390 > submission [S
 49 15.399203 0.000105  10.0.0.3  10.0.0.4  TCP  0x570e 0x04  64  32 55301 139    1687813903 0x14    92       55301 > netbios-ssn [
 50 15.399251 0.000048  10.0.0.3  10.0.0.4  TCP  0xc695 0x04  64  40 54878 111    1678667415 0x02  5840  1460 54878 > sunrpc [SYN]
 51 15.399272 0.000021  10.0.0.3  10.0.0.4  TCP  0x128a 0x04  64  40 47922 9220   1674820232 0x02  5840  1460 47922 > 9220 [SYN] Se
 52 15.399295 0.000023  10.0.0.3  10.0.0.4  TCP  0x402e 0x04  64  40 47272 10629  1680533548 0x02  5840  1460 47272 > 10629 [SYN] S
 53 15.399317 0.000022  10.0.0.3  10.0.0.4  TCP  0x9c82 0x04  64  40 34819 5221   1678521472 0x02  5840  1460 34819 > 5221 [SYN] Se
 54 15.399339 0.000022  10.0.0.3  10.0.0.4  TCP  0xa8ca 0x04  64  40 43564 7002   1681064136 0x02  5840  1460 43564 > afs3-prserver
 55 15.399357 0.000018  10.0.0.3  10.0.0.4  TCP  0x09c5 0x04  64  40 36477 1164   1677107655 0x02  5840  1460 36477 > qsm-proxy [SY
 56 15.399389 0.000032  10.0.0.3  10.0.0.4  TCP  0xbf9d 0x04  64  40 55005 44176  1688737695 0x02  5840  1460 55005 > 44176 [SYN] S
 57 15.399395 0.000006  10.0.0.3  10.0.0.4  TCP  0x6844 0x04  64  40 40682 1971   1686159430 0x02  5840  1460 40682 > netop-school
 58 15.399422 0.000027  10.0.0.3  10.0.0.4  TCP  0x83cd 0x04  64  40 32970 3905   1675373519 0x02  5840  1460 32970 > mupdate [SYN]
 59 15.399444 0.000022  10.0.0.3  10.0.0.4  TCP  0x9171 0x04  64  40 33826 54328  1689598323 0x02  5840  1460 33826 > 54328 [SYN] S
 60 15.499760 0.100316  10.0.0.3  10.0.0.4  TCP  0x90a1 0x04  64  40 33827 54328  1681995962 0x02  5840  1460 33827 > 54328 [SYN] S
 61 15.499778 0.000018  10.0.0.3  10.0.0.4  TCP  0xb3aa 0x04  64  40 32973 3905   1688804273 0x02  5840  1460 32973 > mupdate [SYN]
 62 15.499788 0.000010  10.0.0.3  10.0.0.4  TCP  0x78f6 0x04  64  40 40687 1971   1687539949 0x02  5840  1460 40687 > netop-school
```

```
⊟ Frame 9 (74 bytes on wire, 74 bytes captured)
    Arrival Time: Jul 23, 2009 12:50:51.190001000
    [Time delta from previous captured frame: 0.000047000 seconds]
    [Time delta from previous displayed frame: 0.000047000 seconds]
    [Time since reference or first frame: 14.197835000 seconds]
    Frame Number: 9
    Frame Length: 74 bytes
    Capture Length: 74 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:tcp]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80]
⊟ Ethernet II, Src: 00:19:b9:17:5b:e0 (00:19:b9:17:5b:e0), Dst: 00:19:b9:16:bf
  ⊞ Destination: 00:19:b9:16:bf:9b (00:19:b9:16:bf:9b)
  ⊞ Source: 00:19:b9:17:5b:e0 (00:19:b9:17:5b:e0)
    Type: IP (0x0800)
⊞ Internet Protocol, Src: 10.0.0.3 (10.0.0.3), Dst: 10.0.0.4 (10.0.0.4)
⊟ Transmission Control Protocol, Src Port: 44290 (44290), Dst Port: http (80),
    Source port: 44290 (44290)
    Destination port: http (80)
    [Stream index: 6]
    Sequence number: 1671952606
    Header length: 40 bytes
  ⊞ Flags: 0x02 (SYN)
    Window size: 5840
  ⊞ Checksum: 0x6c66 [correct]
  ⊟ Options: (20 bytes)
      Maximum segment size: 1460 bytes
      SACK permitted
      Timestamps: TSval 1484501, TSecr 0
      NOP
      Window scale: 6 (multiply by 64)
```

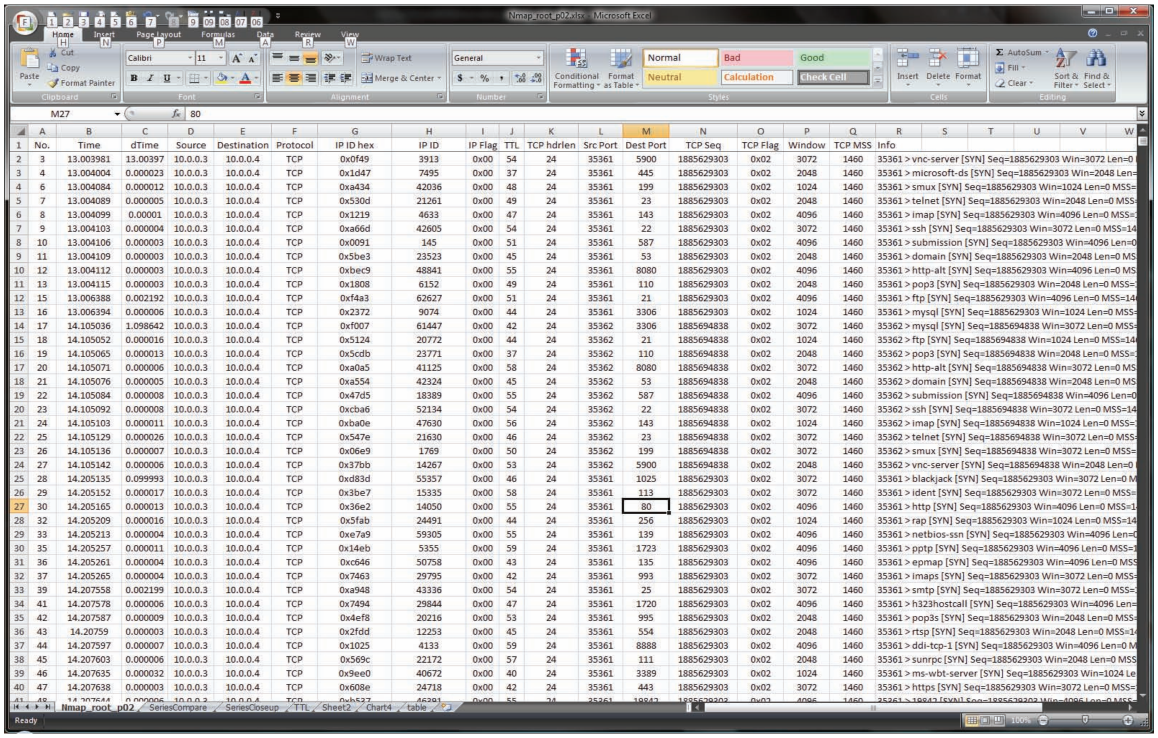
```
0000  00 19 b9 16 bf 9b 00 19  b9 17 5b e0 08 00 45 00   ........ ..[...E.
0010  00 3c 5e 0b 40 00 40 06  c8 aa 0a 00 00 03 0a 00   .<^.@.@. ........
0020  00 04 ad 02 00 50 63 a7  f8 de 00 00 00 00 a0 02   .....Pc. ........
0030  16 d0 6c 66 00 00 02 04  05 b4 04 02 08 0a 00 16   ..lf.... ........
0040  a6 d5 00 00 00 00 01 03  03 06                     ........ ..
```

File: "C:\Users\areku\Documents\NPS_Thesi... Packets: 2006 Displayed: 2004 Marked: 0

Figure 3. Wireshark network data analysis and parsing screen capture

3. Spreadsheet Analysis

The comma-delimited value field ".csv" files exported from Wireshark were imported into Microsoft Excel and converted into Excel spreadsheet files. This provided the experimenter with the ability to extract additional data, for example, charts of TCP Sequence Number trends. Figure 4 shows a typical screenshot. Note how by itself, the rows and columns of numerical data are difficult to use to interpret the network activity the data represents.



	No.	Time	dTime	Source	Destination	Protocol	IP ID hex	IP ID	IP Flag	TTL	TCP hdrlen	Src Port	Dest Port	TCP Seq	TCP Flag	Window	TCP MSS	Info
2	3	13.003981	13.00397	10.0.0.3	10.0.0.4	TCP	0x0f49	3913	0x00	54	24	35361	5900	1885629303	0x02	3072	1460	35361 > vnc-server [SYN] Seq=1885629303 Win=3072 Len=0
3	4	13.004004	0.000023	10.0.0.3	10.0.0.4	TCP	0x1d47	7495	0x00	37	24	35361	445	1885629303	0x02	2048	1460	35361 > microsoft-ds [SYN] Seq=1885629303 Win=2048 Len=
4	6	13.004084	0.000012	10.0.0.3	10.0.0.4	TCP	0xa434	42036	0x00	48	24	35361	199	1885629303	0x02	1024	1460	35361 > smux [SYN] Seq=1885629303 Win=1024 Len=0 MSS=
5	7	13.004089	0.000005	10.0.0.3	10.0.0.4	TCP	0x530d	21261	0x00	49	24	35361	23	1885629303	0x02	2048	1460	35361 > telnet [SYN] Seq=1885629303 Win=2048 Len=0 MSS
6	8	13.004099	0.00001	10.0.0.3	10.0.0.4	TCP	0x1219	4633	0x00	47	24	35361	143	1885629303	0x02	4096	1460	35361 > imap [SYN] Seq=1885629303 Win=4096 Len=0 MSS=
7	9	13.004103	0.000004	10.0.0.3	10.0.0.4	TCP	0xa66d	42605	0x00	54	24	35361	22	1885629303	0x02	3072	1460	35361 > ssh [SYN] Seq=1885629303 Win=3072 Len=0 MSS=14
8	10	13.004106	0.000003	10.0.0.3	10.0.0.4	TCP	0x0091	145	0x00	51	24	35361	587	1885629303	0x02	4096	1460	35361 > submission [SYN] Seq=1885629303 Win=4096 Len=0
9	11	13.004109	0.000003	10.0.0.3	10.0.0.4	TCP	0x5be3	23523	0x00	45	24	35361	53	1885629303	0x02	2048	1460	35361 > domain [SYN] Seq=1885629303 Win=2048 Len=0 MS
10	12	13.004112	0.000003	10.0.0.3	10.0.0.4	TCP	0xbec9	48841	0x00	55	24	35361	8080	1885629303	0x02	4096	1460	35361 > http-alt [SYN] Seq=1885629303 Win=4096 Len=0 MS
11	13	13.004115	0.000003	10.0.0.3	10.0.0.4	TCP	0x1808	6152	0x00	49	24	35361	110	1885629303	0x02	2048	1460	35361 > pop3 [SYN] Seq=1885629303 Win=2048 Len=0 MSS=
12	15	13.006388	0.002192	10.0.0.3	10.0.0.4	TCP	0xf4a3	62627	0x00	51	24	35361	21	1885629303	0x02	4096	1460	35361 > ftp [SYN] Seq=1885629303 Win=4096 Len=0 MSS=14
13	16	13.006394	0.000006	10.0.0.3	10.0.0.4	TCP	0x2372	9074	0x00	44	24	35361	3306	1885629303	0x02	1024	1460	35361 > mysql [SYN] Seq=1885629303 Win=1024 Len=0 MSS=
14	17	14.105036	1.098642	10.0.0.3	10.0.0.4	TCP	0xf007	61447	0x00	42	24	35362	3306	1885694838	0x02	3072	1460	35362 > mysql [SYN] Seq=1885694838 Win=3072 Len=0 MSS=
15	18	14.105052	0.000016	10.0.0.3	10.0.0.4	TCP	0x5124	20772	0x00	44	24	35362	21	1885694838	0x02	1024	1460	35362 > ftp [SYN] Seq=1885694838 Win=1024 Len=0 MSS=14
16	19	14.105065	0.000013	10.0.0.3	10.0.0.4	TCP	0x5cdb	23771	0x00	37	24	35362	110	1885694838	0x02	2048	1460	35362 > pop3 [SYN] Seq=1885694838 Win=2048 Len=0 MSS=
17	20	14.105071	0.000006	10.0.0.3	10.0.0.4	TCP	0xa0a5	41125	0x00	58	24	35362	8080	1885694838	0x02	3072	1460	35362 > http-alt [SYN] Seq=1885694838 Win=3072 Len=0 MS
18	21	14.105076	0.000005	10.0.0.3	10.0.0.4	TCP	0xa554	42324	0x00	45	24	35362	53	1885694838	0x02	2048	1460	35362 > domain [SYN] Seq=1885694838 Win=2048 Len=0 MS
19	22	14.105084	0.000008	10.0.0.3	10.0.0.4	TCP	0x47d5	18389	0x00	55	24	35362	587	1885694838	0x02	4096	1460	35362 > submission [SYN] Seq=1885694838 Win=4096 Len=0
20	23	14.105092	0.000008	10.0.0.3	10.0.0.4	TCP	0xcba6	52134	0x00	54	24	35362	22	1885694838	0x02	3072	1460	35362 > ssh [SYN] Seq=1885694838 Win=3072 Len=0 MSS=14
21	24	14.105103	0.000011	10.0.0.3	10.0.0.4	TCP	0xba0e	47630	0x00	56	24	35362	143	1885694838	0x02	1024	1460	35362 > imap [SYN] Seq=1885694838 Win=1024 Len=0 MSS=
22	25	14.105129	0.000026	10.0.0.3	10.0.0.4	TCP	0x547e	21630	0x00	46	24	35362	23	1885694838	0x02	3072	1460	35362 > telnet [SYN] Seq=1885694838 Win=3072 Len=0 MSS
23	26	14.105136	0.000007	10.0.0.3	10.0.0.4	TCP	0x06e9	1769	0x00	50	24	35362	199	1885694838	0x02	3072	1460	35362 > smux [SYN] Seq=1885694838 Win=3072 Len=0 MSS=
24	27	14.105142	0.000006	10.0.0.3	10.0.0.4	TCP	0x37bb	14267	0x00	53	24	35362	5900	1885694838	0x02	2048	1460	35362 > vnc-server [SYN] Seq=1885694838 Win=2048 Len=0
25	28	14.205135	0.099993	10.0.0.3	10.0.0.4	TCP	0xd83d	55357	0x00	46	24	35361	1025	1885629303	0x02	3072	1460	35361 > blackjack [SYN] Seq=1885629303 Win=3072 Len=0 M
26	29	14.205152	0.000017	10.0.0.3	10.0.0.4	TCP	0x3be7	15335	0x00	58	24	35361	113	1885629303	0x02	3072	1460	35361 > ident [SYN] Seq=1885629303 Win=3072 Len=0 MSS=
27	30	14.205165	0.000013	10.0.0.3	10.0.0.4	TCP	0x36e2	14050	0x00	55	24	35361	80	1885629303	0x02	4096	1460	35361 > http [SYN] Seq=1885629303 Win=4096 Len=0 MSS=1
28	32	14.205209	0.000016	10.0.0.3	10.0.0.4	TCP	0x5fab	24491	0x00	44	24	35361	256	1885629303	0x02	1024	1460	35361 > rap [SYN] Seq=1885629303 Win=1024 Len=0 MSS=14
29	33	14.205213	0.000004	10.0.0.3	10.0.0.4	TCP	0xe7a9	59305	0x00	55	24	35361	139	1885629303	0x02	4096	1460	35361 > netbios-ssn [SYN] Seq=1885629303 Win=4096 Len=0
30	35	14.205257	0.000011	10.0.0.3	10.0.0.4	TCP	0x14eb	5355	0x00	59	24	35361	1723	1885629303	0x02	4096	1460	35361 > pptp [SYN] Seq=1885629303 Win=4096 Len=0 MSS=1
31	36	14.205261	0.000004	10.0.0.3	10.0.0.4	TCP	0xc646	50758	0x00	43	24	35361	135	1885629303	0x02	4096	1460	35361 > epmap [SYN] Seq=1885629303 Win=4096 Len=0 MSS
32	37	14.205265	0.000004	10.0.0.3	10.0.0.4	TCP	0x7463	29795	0x00	42	24	35361	993	1885629303	0x02	3072	1460	35361 > imaps [SYN] Seq=1885629303 Win=3072 Len=0 MSS=
33	39	14.207558	0.002199	10.0.0.3	10.0.0.4	TCP	0xa948	43336	0x00	54	24	35361	25	1885629303	0x02	3072	1460	35361 > smtp [SYN] Seq=1885629303 Win=3072 Len=0 MSS=
34	41	14.207578	0.000006	10.0.0.3	10.0.0.4	TCP	0x7494	29844	0x00	47	24	35361	1720	1885629303	0x02	4096	1460	35361 > h323hostcall [SYN] Seq=1885629303 Win=4096 Len=
35	42	14.207587	0.000009	10.0.0.3	10.0.0.4	TCP	0x4ef8	20216	0x00	53	24	35361	995	1885629303	0x02	2048	1460	35361 > pop3s [SYN] Seq=1885629303 Win=2048 Len=0 MSS=
36	43	14.20759	0.000003	10.0.0.3	10.0.0.4	TCP	0x2fdd	12253	0x00	45	24	35361	554	1885629303	0x02	2048	1460	35361 > rtsp [SYN] Seq=1885629303 Win=2048 Len=0 MSS=14
37	44	14.207597	0.000007	10.0.0.3	10.0.0.4	TCP	0x1025	4133	0x00	59	24	35361	8888	1885629303	0x02	4096	1460	35361 > ddi-tcp-1 [SYN] Seq=1885629303 Win=4096 Len=0 M
38	45	14.207603	0.000006	10.0.0.3	10.0.0.4	TCP	0x569c	22172	0x00	57	24	35361	111	1885629303	0x02	2048	1460	35361 > sunrpc [SYN] Seq=1885629303 Win=2048 Len=0 MSS
39	46	14.207635	0.000032	10.0.0.3	10.0.0.4	TCP	0x9ee0	40672	0x00	40	24	35361	3389	1885629303	0x02	1024	1460	35361 > ms-wbt-server [SYN] Seq=1885629303 Win=1024 Le
40	47	14.207638	0.000003	10.0.0.3	10.0.0.4	TCP	0x608e	24718	0x00	42	24	35361	443	1885629303	0x02	3072	1460	35361 > https [SYN] Seq=1885629303 Win=3072 Len=0 MSS=

Figure 4. Excel spreadsheet network data analysis

4. RUMINT Parallel Coordinate Plots

RUMINT is a network data visualization tool created by Greg Conti [14], [24]. Among its most useful features are its VCR-like playback interface and parallel coordinate plot. The program was run both live, as the reconnaissance scans were performed for data capture, and again later to assist in data analysis. The parallel coordinate plot is constructed in the following manner: for each field listed on the horizontal axis along the bottom, there is a corresponding vertical line displaying the linear minimum to maximum potential value for that field. The program allows customized selection of horizontal fields to display and color-codes the packet protocols, with TCP being green, UDP as orange, and ICMP as blue [14], [24]. An example plot is shown in Figure 5. Note how much more rapidly one can develop SA about what activity is occurring on the network compared to looking at the screen captures from Wireshark and Excel.

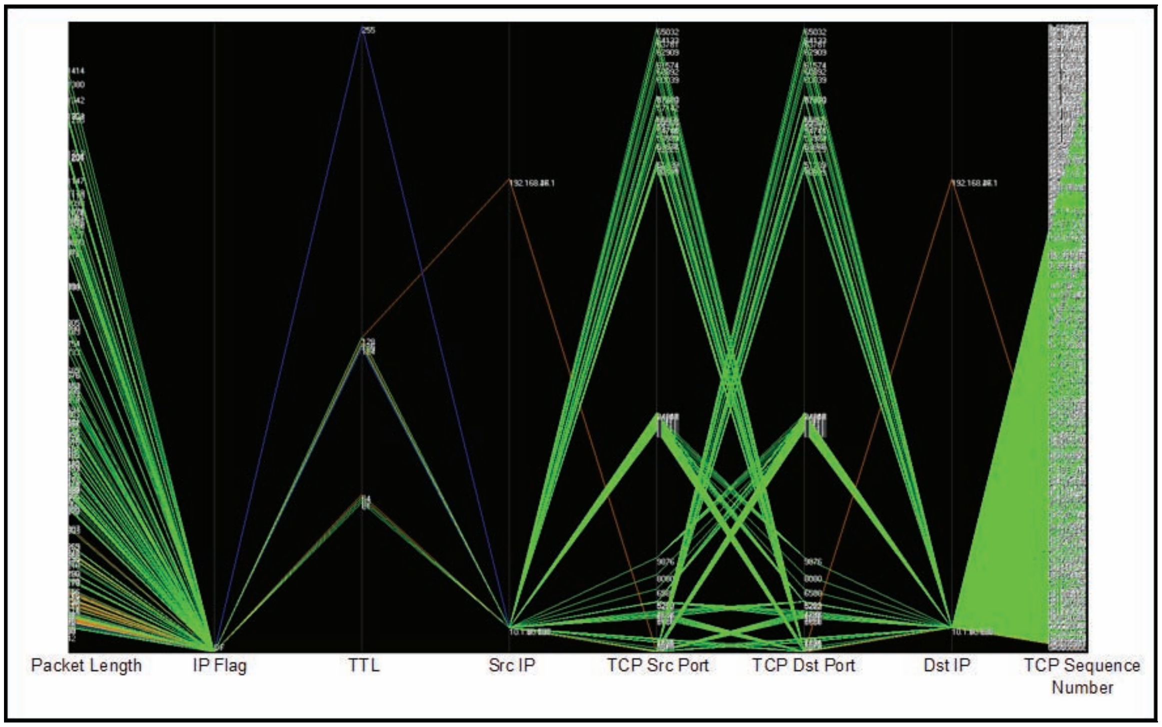


Figure 5. RUMINT Parallel Coordinate Plot Example

THIS PAGE INTENTIONALLY LEFT BLANK

IV. EXPERIMENTATION RESULTS

This chapter reports the result of the methodology discussed in the previous chapter. Each section addresses one of the reconnaissance tools examined.

A. NMAP

Nmap, created by Gordon "Fyodor" Lyon, is the most famous network reconnaissance tool [22], [29]. Nmap has undergone continuous improvement since its initial release in 1997, setting the standard in functional reconnaissance capabilities. Nmap offers a considerable number of advanced scanning features, to include active OS version detection and enumeration. However, this thesis is focused on the default TCP vertical SYN scan.

The test scans were initiated with root-level privileges and default settings by the terminal command `sudo nmap 10.0.0.4`. If necessary, Nmap begins by converting the specified target into an IPv4 address with a Domain Name System (DNS) lookup. Next, it determines if the target host is reachable by sending an ARP request, ICMP echo request ping, or TCP ACK packet to port 80 in iterative attempts. It then performs a reverse-DNS query to convert the IPv4 address back to a name. This reverse-DNS query can be turned off with the "`-n`" switch, but it is included in default behavior. Detecting this reverse-DNS query can be an alert to presence of an unauthorized network scan, i.e., the attacker has already gained direct access on the network. However, if the scan were conducted from outside the defender's network, the defender would not be able to detect these reverse DNS lookups. Nmap then begins its principle effort; a TCP SYN scan against Fyodor's empirically determined 1,000 most used ports. A savvy attacker may modify these 1,000 default scanned ports by changing the list of ports to scan in the "`nmap-services`" program file. Consequently, the simple list of the destination ports is not as reliable a fingerprint parameter as other behavioral characteristics. The tests revealed that a complete default Nmap scan consisted of 1,997 packets. A direct comparison of these destination ports is left to further research. Lastly, Nmap prints its scan results for the attacker [22], [29]. Since this thesis is focused on developing the network defender's ability to passively fingerprint the reconnaissance tools, the accuracy, and usefulness, of the reconnaissance tool's output is inconsequential.

Figure 6 shows the complete RUMINT Parallel Coordinate Plot view of a default Nmap scan. Note how many of the possible values for the IP Identification numbers are used compared to the apparently nearly singular value for the TCP Sequence Number. More analysis of this observation will be made later in this section. The IP Identification Number continued to show similar behavior for every reconnaissance tool tested, with one notable exception. Consequently, the IP ID field and several others that were of little value to developing the fingerprints were filtered out in subsequent RUMINT figures and analysis.

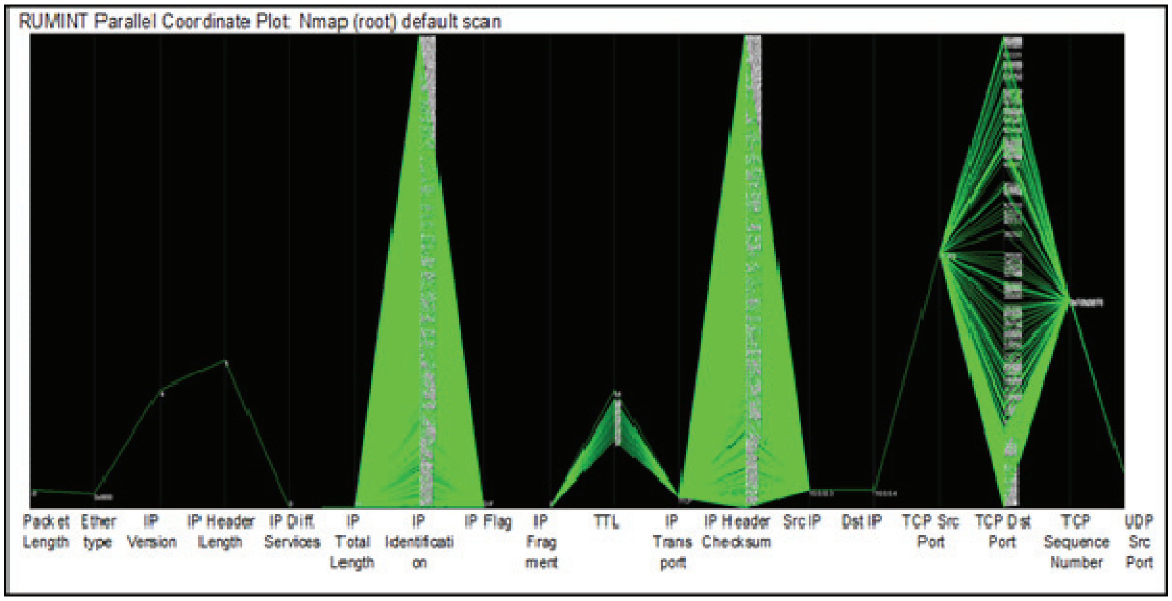


Figure 6. Nmap default scan in RUMINT Parallel Coordinate Plot view

Figure 7 represents the field headers that provide the best concise illustration of each reconnaissance tool's visual fingerprint. This RUMINT header field format will be used for the remainder of the analyses.

For Nmap's default scan, note the following: constant packet length, Don't Fragment Flag not set, a spread TTL distribution, single or very small spread Source Port selection, the high concentration of packets sent to well-known and registered destination ports, and the narrow TCP Sequence Number range.

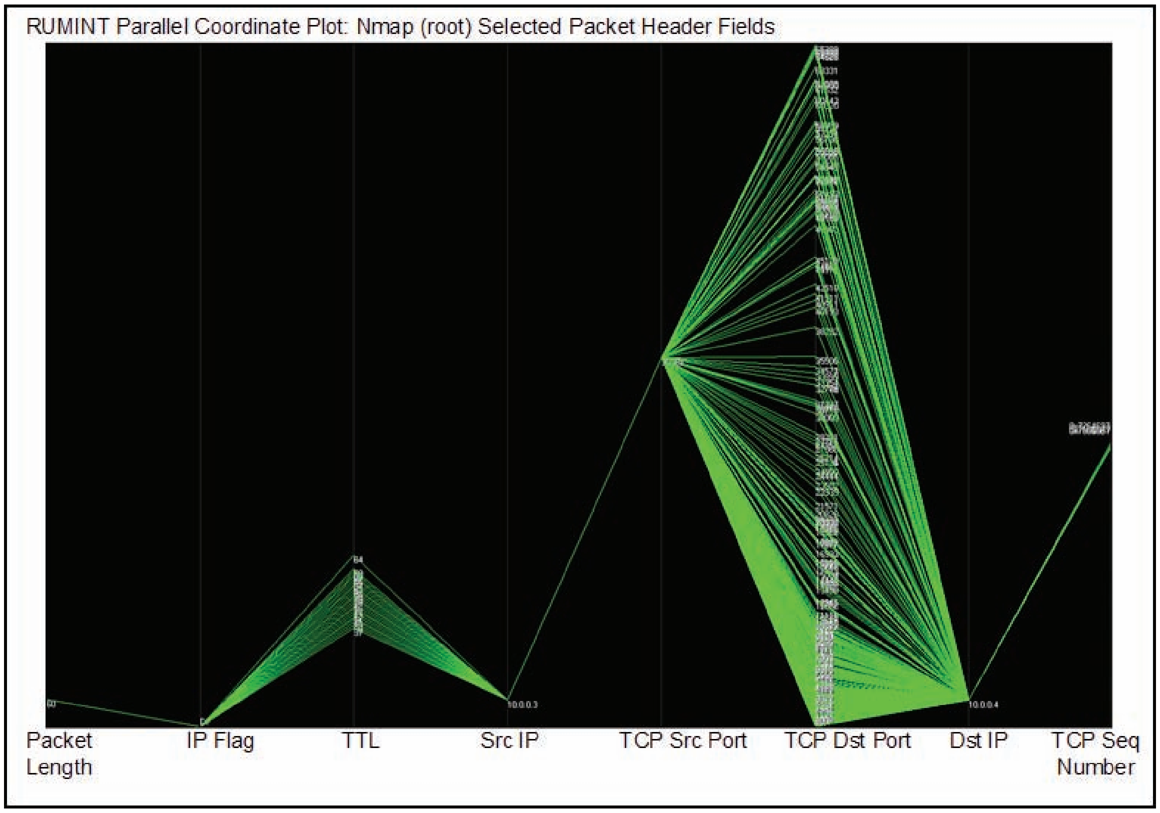


Figure 7. Nmap default scan in RUMINT

Further analysis in Wireshark showed that the packets with a TTL value of 64 were the "RST" packets sent in response to the "SYN+ACK" from the target's open ports. The actual SYN scan activity used an apparently randomized TTL value within a set range. The varying TTL value would make it more difficult for the target to determine the attacker's distance in terms of router hops.

The next figure shows a comparison of the TCP sequence numbers and the IP identification numbers. Both are ordinarily supposed to be initialized as a random number to make it difficult for an attacker to perform an injection MITM attack by guessing the next IP ID or TCP sequence number. Recall that the TCP sequence number of SYN packets serve as the Initial Sequence Number to synchronize follow-on communications between the sender and receiver. The observation of Figure 8 demonstrates that Nmap does not provide a large range of initial TCP sequence numbers; it appears to be very linear. Each independent vertical axis on the figure shows the minimum and maximum values for the TCP sequence number and IP identification fields. This allows a comparison of each field's use of possible values relative to the other.

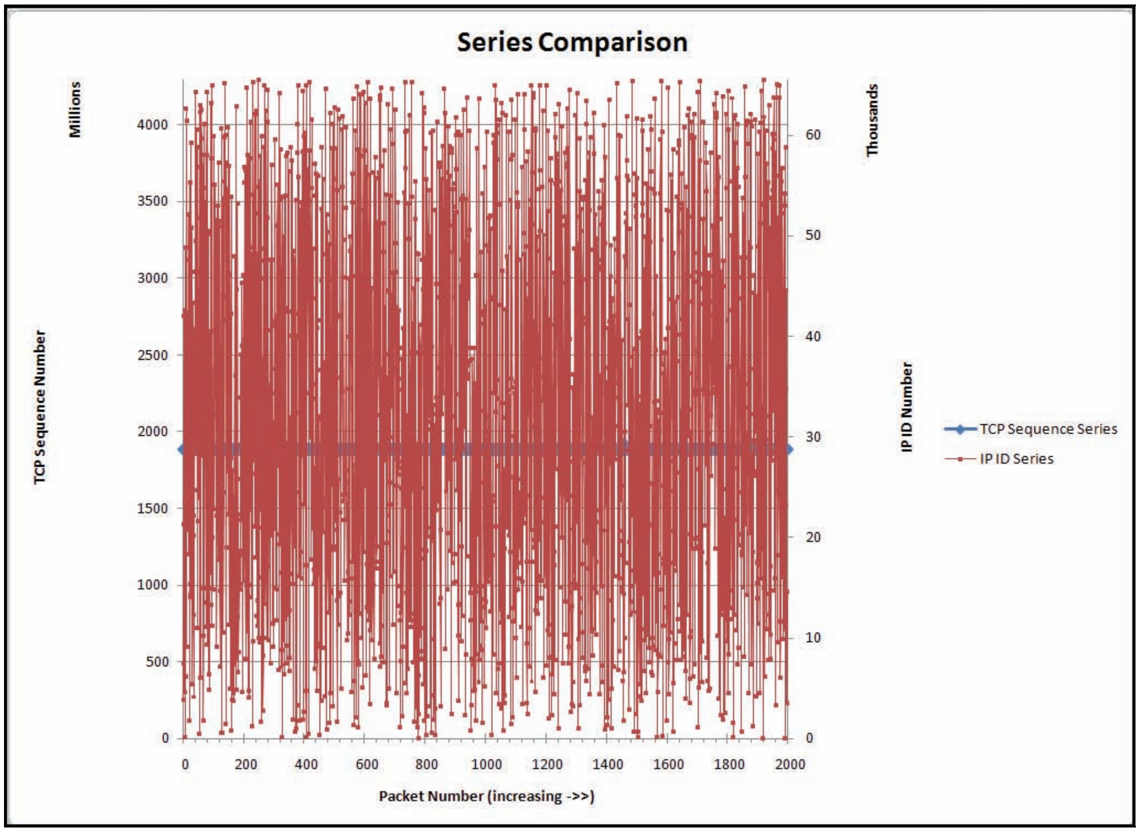


Figure 8. Nmap TCP Sequence Number and IP ID Number Whole Scan Series Comparison

Figure 8 shows the results of the comparison for all 1,997 packets in the default scan. The wide-ranging spread of the IP identification numbers occludes the apparently linear TCP sequence numbers. Figure 9 shows the results for only the first 100 packets. The scale of the TCP sequence number vertical axis was also changed to show the trend in greater detail.

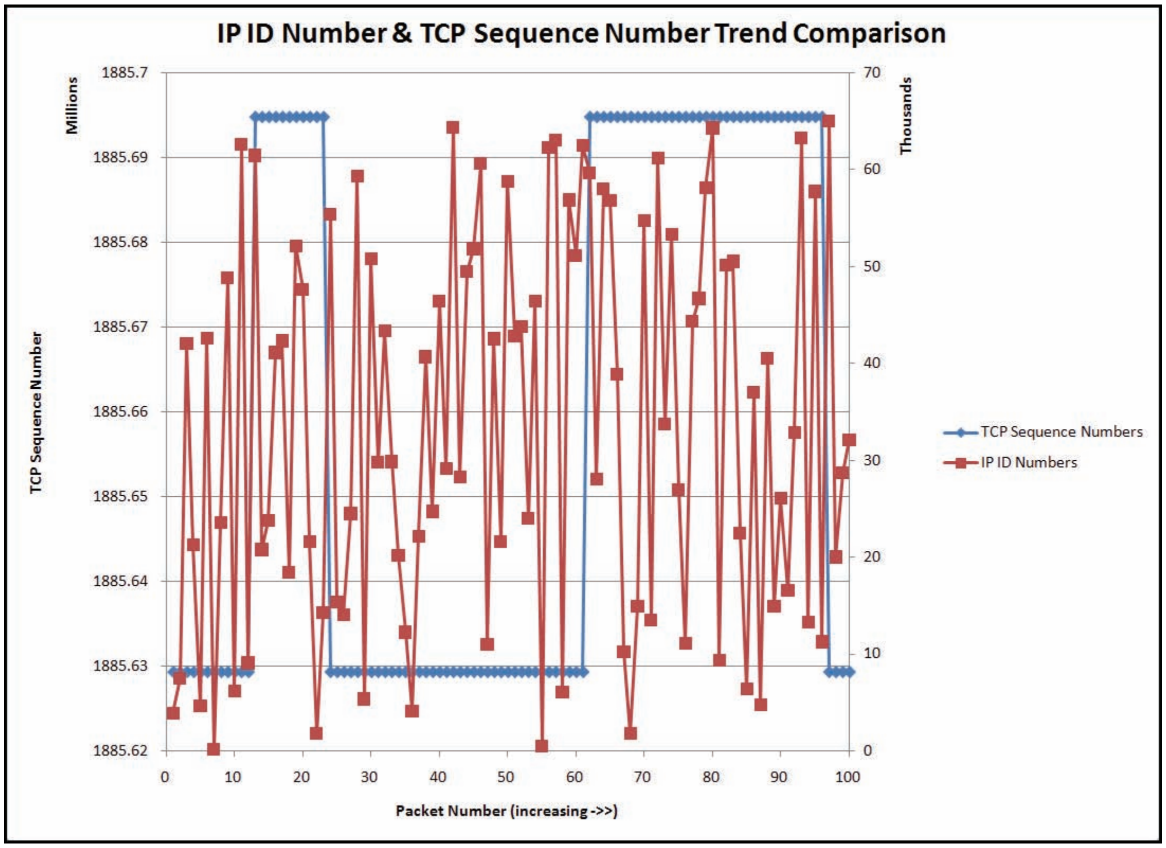


Figure 9. Nmap TCP Sequence Number and IP ID Number First 100 Packets Series Comparison

Suddenly, it is revealed that the TCP sequence number is not wholly linear, but follows a distinct trend alternating between two values. This characteristic is not readily obvious by watching the playback in RUMINT or in the initial observations of the textual display of TCP sequence numbers in Wireshark or Excel data fields.

As mentioned earlier, the IP identification field maintained a similar randomized trend for each test run of every reconnaissance tool tested, with the exception of one. Consequently, the IP identification field was dropped from further analysis. The next component of this line of analysis was to determine if the Nmap TCP sequence number alternating sequence displayed any patterns by itself. Figure 10 shows the TCP sequence number trend of 200 packets, starting with packet 1,400.

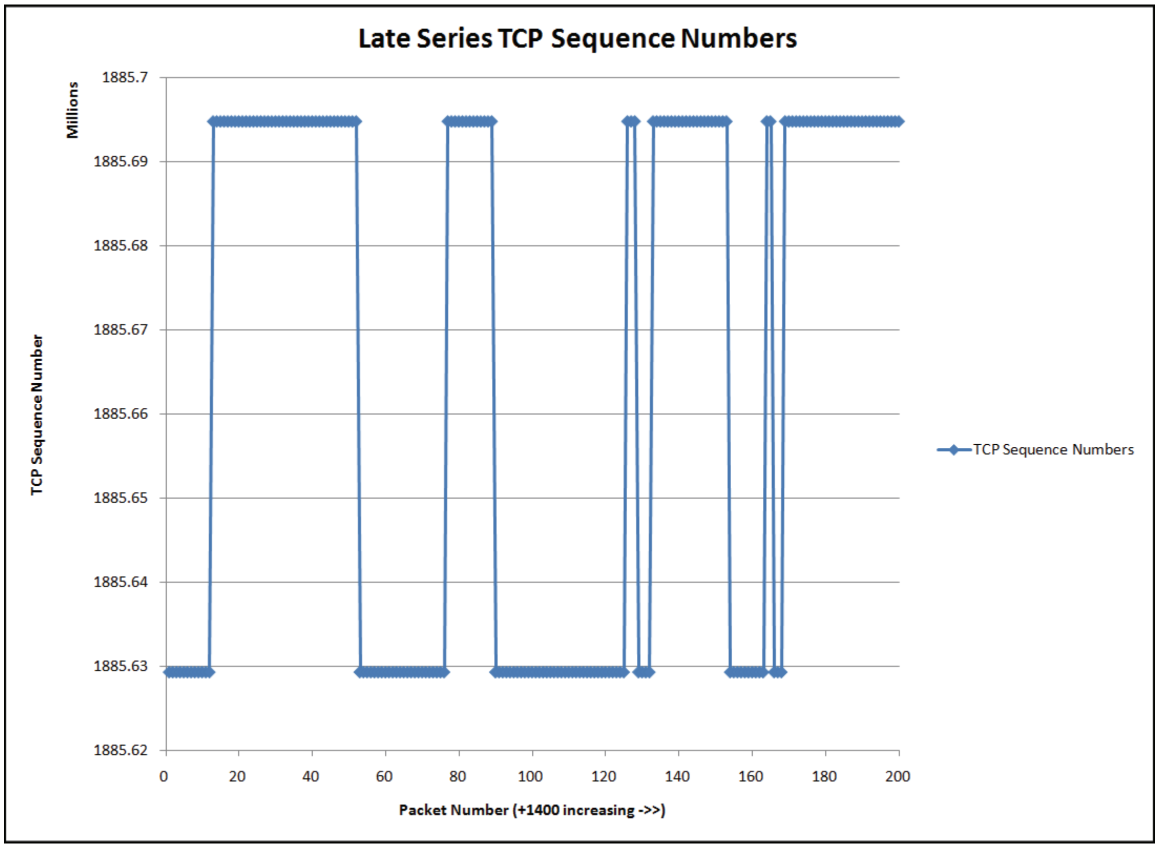


Figure 10. Nmap TCP Sequence Number Trend Near End of Scan

Figure 10 demonstrates that Nmap's TCP sequence numbers to not maintain a constant frequency-alternating pattern. In the test data shown here, the lower TCP sequence number was 1885629303 and the higher alternating number was 1885694838. The difference between the upper and lower numbers was 65,535.

Two other interesting occurrences were noted regarding Nmap's TCP sequence number trend. First, Nmap only sends one packet to each port during the course of its default scan, with the exception of port 80, which receives three packets. For both the second and third packet sent to port 80, the TCP sequence number made a significant jump in value. It then resumed the previous alternating trend. The first jump, the second packet to port 80, was an even 256 times the base alternating difference value of 65,535 above the lower sequence number. The second jump, the third packet to port 80, was an even 512 times the base alternating difference value of 65,535 above the lower sequence number.

Second, we observed that the TCP source port also alternates, by a difference of one, between two values. In this test, the source ports were 35361 and 35362. It was further observed that the source port alternates in the exact same low-high pattern as the TCP sequence numbers. This is an additional distinguishing fingerprint of Nmap. These distinct saw-tooth patterns were each sufficient to uniquely identify Nmap. Therefore, a deeper analysis of these patterns is left to future research.

As observed in Figure 7, Nmap uses a randomized range of different TTL values. These values fall between 37 and 59. If only a few packets were sent, this could make it difficult to estimate the sender's distance away by counting router hops decrementing the TTL values. However, with the 1,997 packets sent in the default scan, the spread of TTL values between 37 and 59 remains apparent. An illustration of the standardized TTL value frequency distribution is shown in Figure 11. The frequency distribution at each TTL value does not remain constant, but always followed a pattern similar to the one shown.

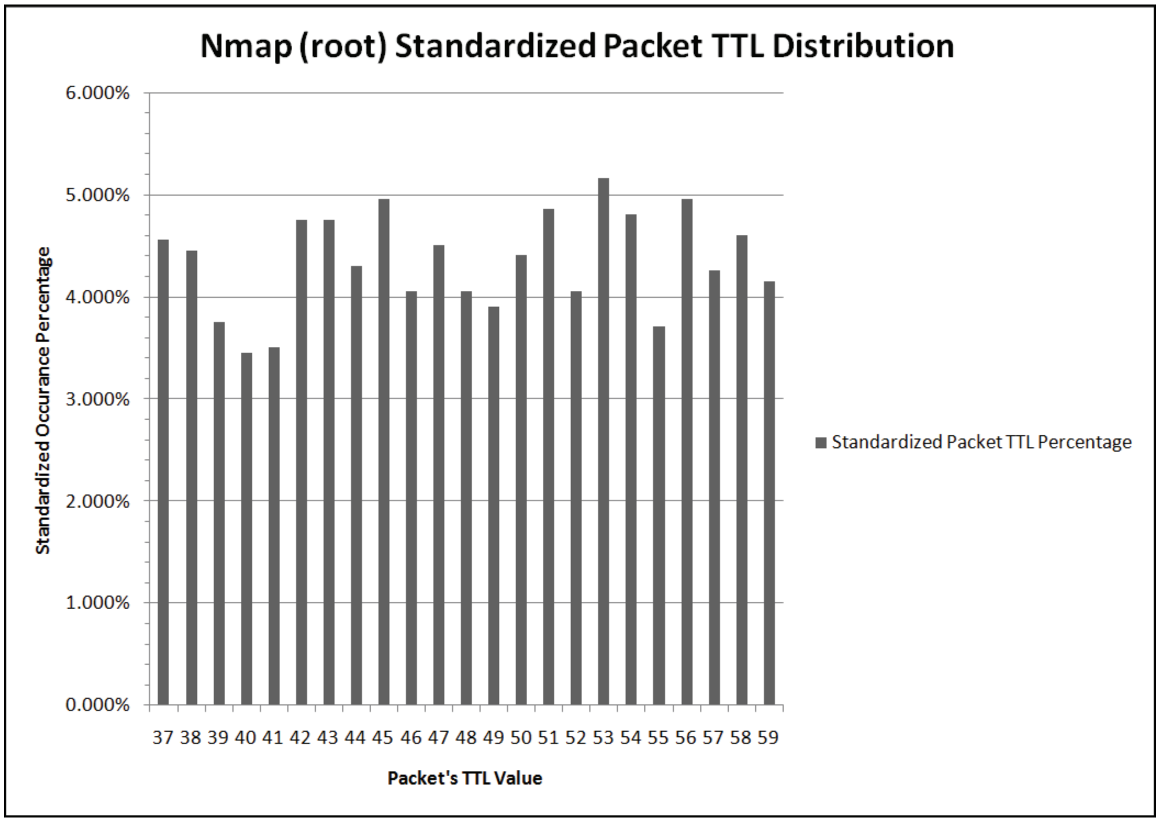
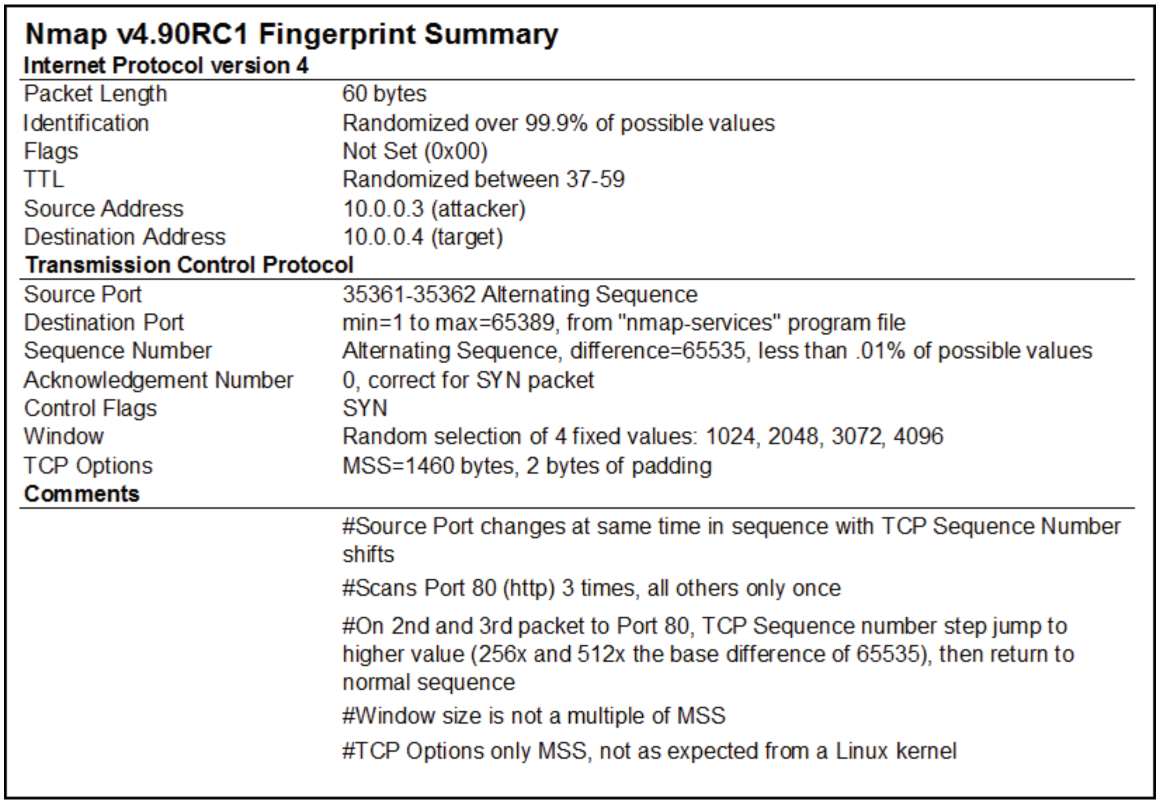


Figure 11. Nmap Standardized Packet TTL Frequency Distribution

An additional unique characteristic of Nmap is the randomized selection of a TCP Window size between one of only four fixed values, 1024, 2048, 3072, and 4096. Since these values are fixed, they provide an effective method to rapidly check a dataset for the possible presence of Nmap by incorporating the values into a Wireshark display filter.

The preceding discussion has outlined a number analysis techniques used to determine a unique fingerprint for Nmap. Table 1 summarizes the results.



Nmap v4.90RC1 Fingerprint Summary

Internet Protocol version 4	
Packet Length	60 bytes
Identification	Randomized over 99.9% of possible values
Flags	Not Set (0x00)
TTL	Randomized between 37-59
Source Address	10.0.0.3 (attacker)
Destination Address	10.0.0.4 (target)
Transmission Control Protocol	
Source Port	35361-35362 Alternating Sequence
Destination Port	min=1 to max=65389, from "nmap-services" program file
Sequence Number	Alternating Sequence, difference=65535, less than .01% of possible values
Acknowledgement Number	0, correct for SYN packet
Control Flags	SYN
Window	Random selection of 4 fixed values: 1024, 2048, 3072, 4096
TCP Options	MSS=1460 bytes, 2 bytes of padding
Comments	
	#Source Port changes at same time in sequence with TCP Sequence Number shifts
	#Scans Port 80 (http) 3 times, all others only once
	#On 2nd and 3rd packet to Port 80, TCP Sequence number step jump to higher value (256x and 512x the base difference of 65535), then return to normal sequence
	#Window size is not a multiple of MSS
	#TCP Options only MSS, not as expected from a Linux kernel

Table 1. Nmap(root) Fingerprint Summary

When reviewing the different data captures for analysis, it was noted that one of the early default Nmap captures had accidentally been performed without root level privileges. There were several notable differences compared to Nmap run as root, each showing that it is likely that the program was relying more on the system kernel to perform certain functions. First, the “Don’t Fragment” flag was set, as expected for a modern OS utilizing PMTUD. Next, the TTL was 64 for all the packets sent, as expected from a Linux kernel. The Window size was 5840; an even multiple of four times the

MSS, which was 1460. The TCP options were also set as expected for Linux, in the following order: MSS, SACK permitted, timestamp, NOP, and window scale. Lastly, the TCP sequence numbers were examined and found to oscillate within a band of approximately 16,500,000 while monotonically increasing. The graph in Figure 12 shows this trend; compare to Figure 10 to see the difference between TCP sequence numbers with user privileges and root privileges.

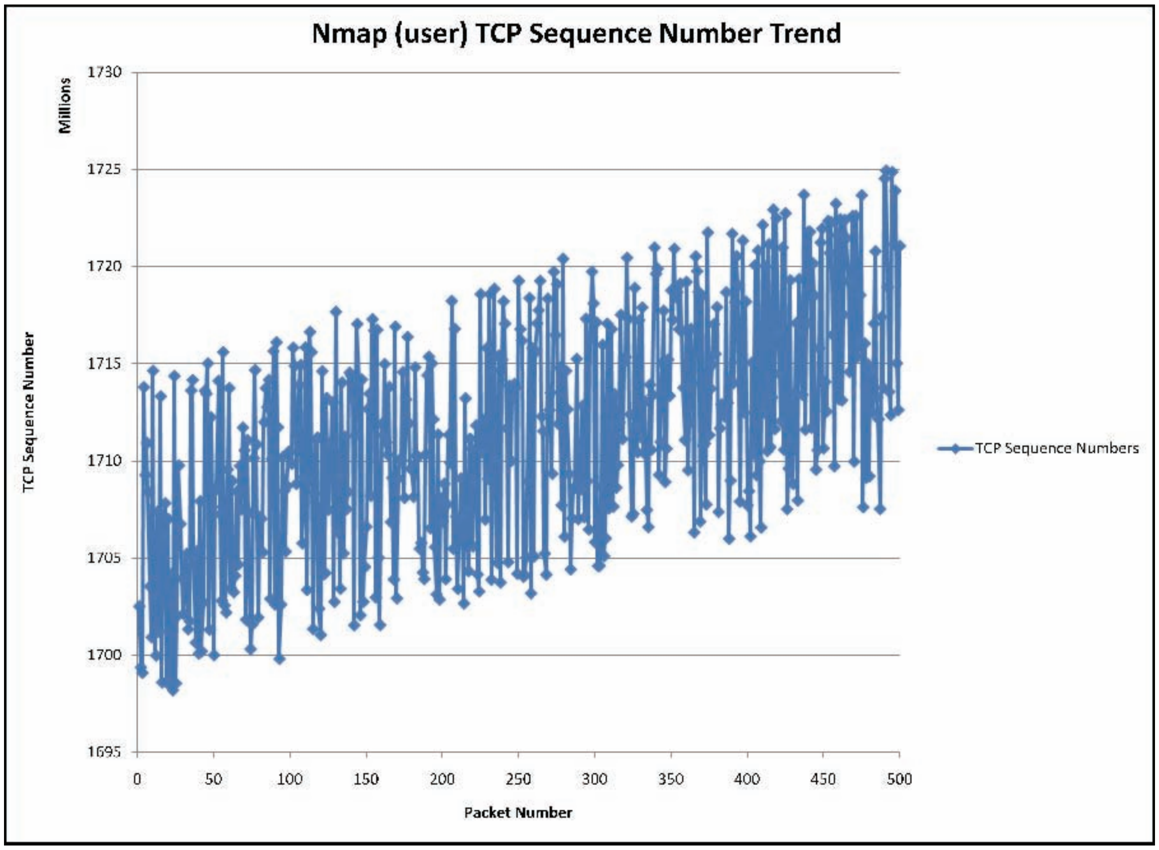


Figure 12. Nmap (user) TCP Sequence Number Trend

The methodologies introduced above were used to examine the remainder of the reconnaissance tools. The following sections detail the notable results.

B. ZENMAP

Zenmap is a Windows Graphical User Interface (GUI) for Nmap [22], [29]. It was included for examination to determine potential differences between the Nmap engine implementation in its native Linux versus Windows. The RUMINT visualization of the default Zenmap scan is shown in Figure 13. Note the following: constant packet length, a range of TTL values (looks similar to Nmap), single or very small spread (and high numbered) source port, high concentration of packets sent to well-known and registered destination ports, and the narrow TCP sequence number range. These characteristics all look nearly indistinguishable from Nmap. Future research with a much larger test sample base would be needed to make an accurate determination based on the selection of source ports. Figure 13 may make it appear that Zenmap frequently chooses a higher source port than Nmap, however, within the tests conducted for this thesis, that distinction was an unreliable indicator.

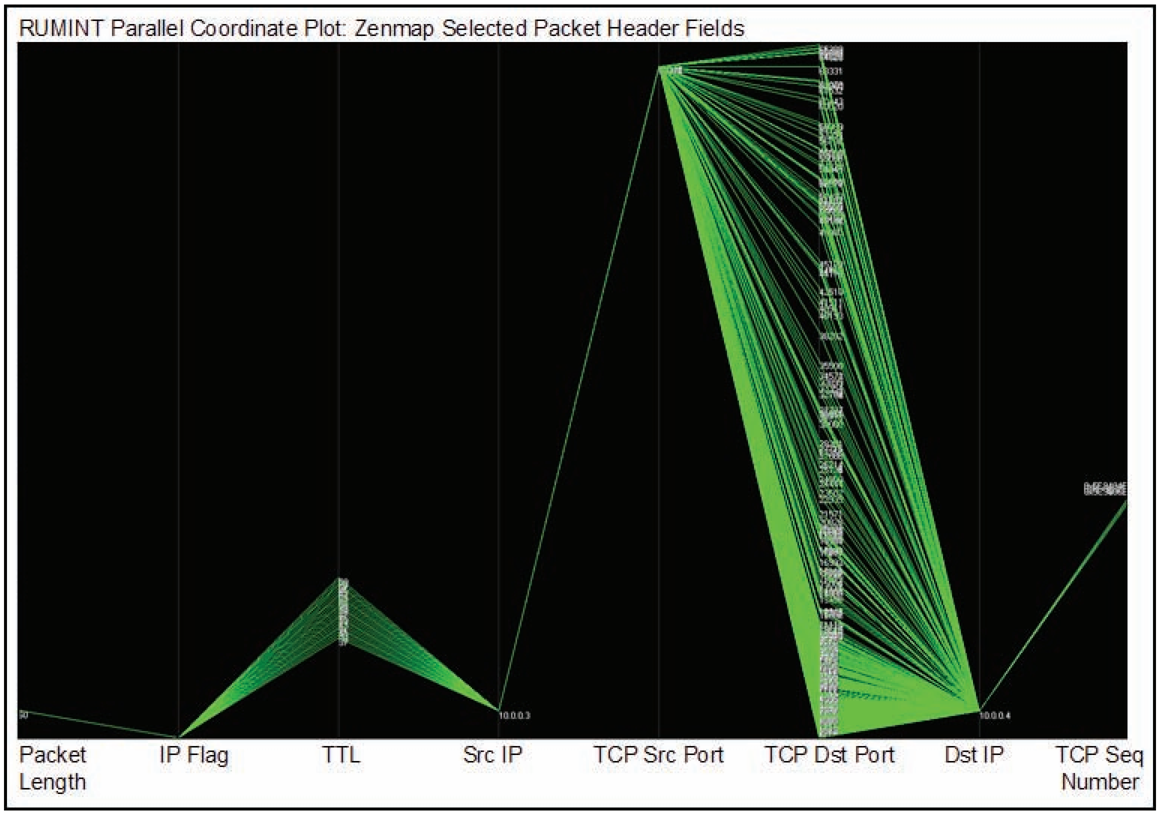


Figure 13. Zenmap default scan in RUMINT

To continue the analysis to find a difference between Nmap and Zenmap, two graphs to compare TCP sequence number trends were made. The first comparison, Figure 14, spans the entire scan, all 1,997 packets in both cases. The spikes that correspond to the second and third packets sent to TCP port 80 occur at relatively the same time in both Nmap and Zenmap. Note that the vertical axes are both measured in the millions, but the scales do not allow a direct comparison. There is an offset due to Nmap and Zenmap initializing the scan sequence on a different TCP sequence number. That was to be expected and was incorporated to enable a better relative comparison on the same graph.

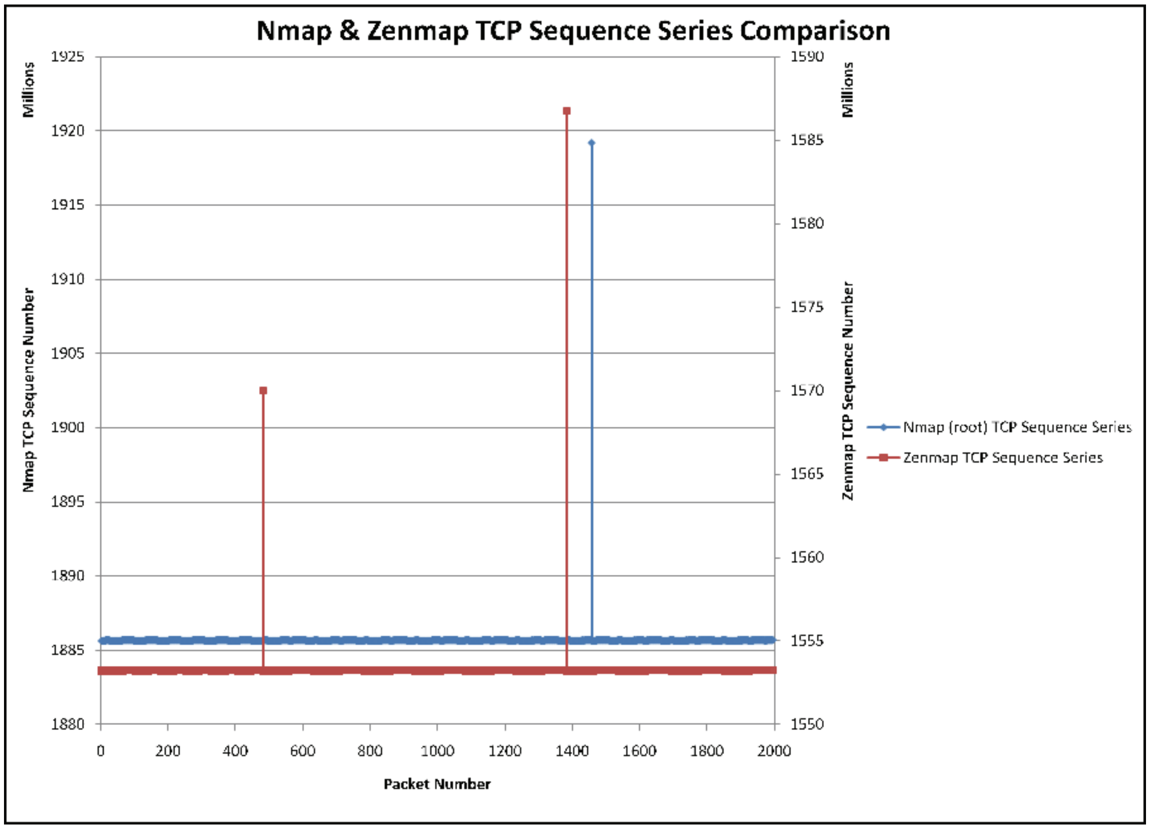


Figure 14. Nmap and Zenmap TCP Sequence Series Comparison

Narrowing the comparison to the first 100 packets of each scan, the basic patterns are still nearly identical. This can be seen in Figure 15. After further analysis comparing the TCP sequence low-high alternation differences, a possible difference was discovered in Zenmap. On occasion, both Nmap and Zenmap alternated TCP sequence numbers by a difference of 65,537, in addition to the more frequently occurring value of 65,535. When the alternating difference was 65,537, Zenmap's second packet to port 80 was still a factor of 256 times the base difference value of 65,537. Zenmap's third packet to port 80 broke the trend by not being the expected even factor 512. Instead, the third packet to port 80 had a TCP sequence number that was slightly less than 512 times the base difference value. The tests conducted for this research showed that the selection of 65,537 might occur more frequently with Zenmap than Nmap. Further research is needed to confirm this finding.

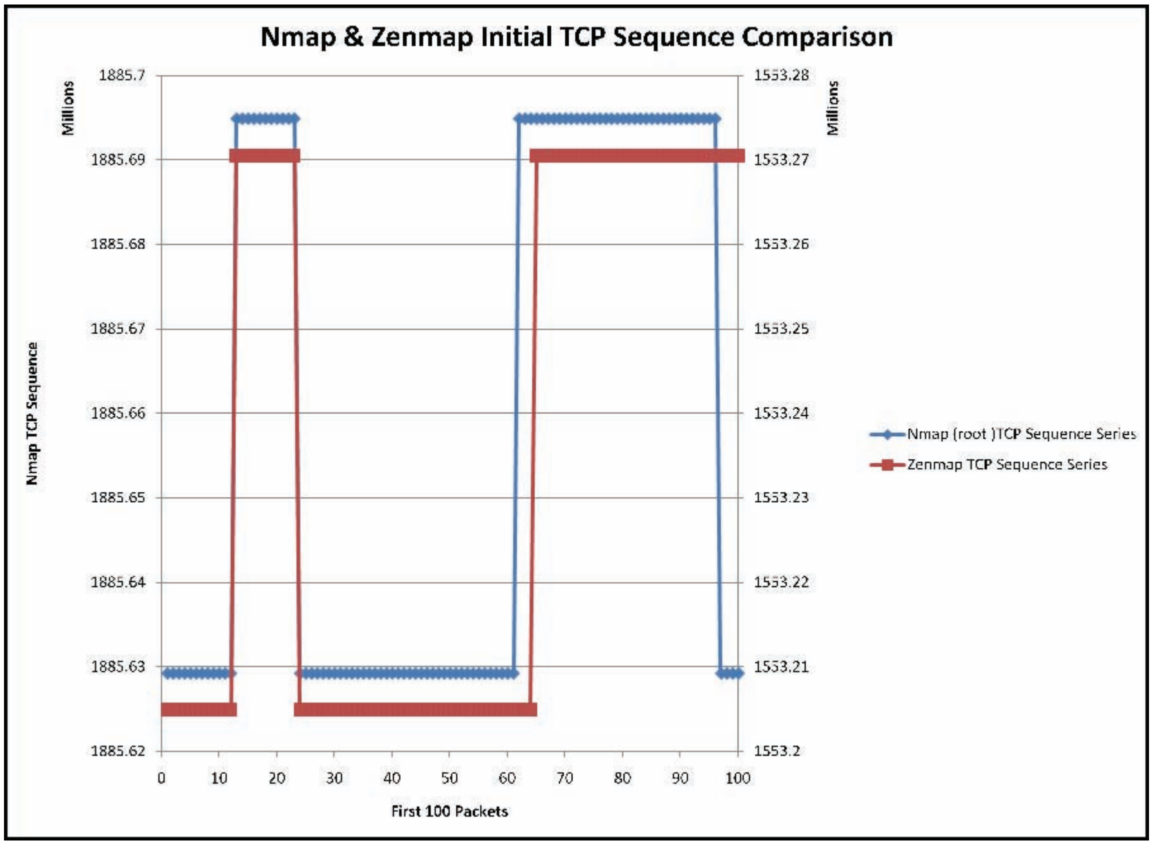
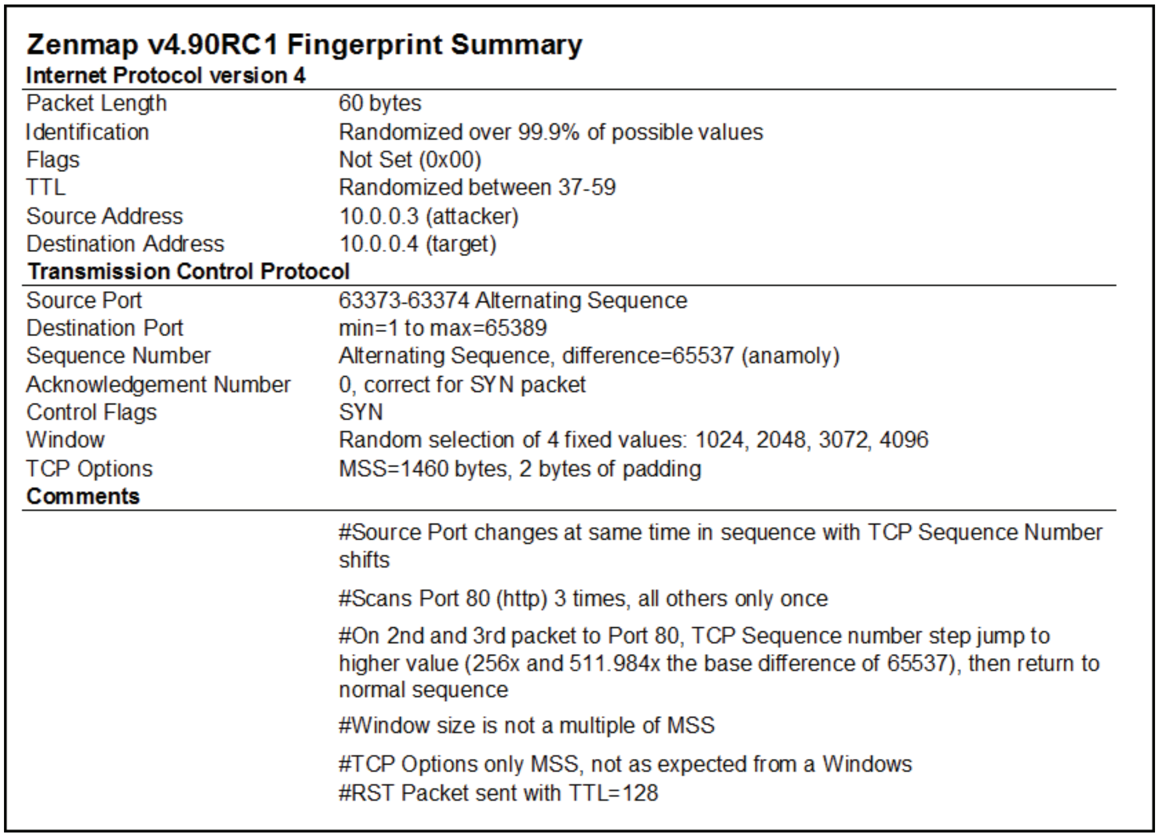


Figure 15. Nmap and Zenmap First 100 Packets TCP Sequence Comparison

As shown in Figure 15, Zenmap can be difficult to distinguish from Nmap. There is one method that can provide the simplest differentiation between Zenmap and Nmap. The “RST” packet sent by the attacker’s host system kernel has that kernel’s default TTL value. However, another implementation of network defense may have been employed in Windows SP3, since it notably violated the original TCP RFC and did not send the “RST” packets. Consequently, while Zenmap conducted its scan, there were no “RST” packets sent back to open ports. The data may also be filtered to search for packets from the same source IP address with a TTL value of 128, indicating the presence of Windows.

Nmap's native Linux sends back "RST" packets with a TTL value of 64. This is a simple method to differentiate between Zenmap and Nmap. Table 2 summarizes the fingerprint characteristics of Zenmap.



Zenmap v4.90RC1 Fingerprint Summary

Internet Protocol version 4	
Packet Length	60 bytes
Identification	Randomized over 99.9% of possible values
Flags	Not Set (0x00)
TTL	Randomized between 37-59
Source Address	10.0.0.3 (attacker)
Destination Address	10.0.0.4 (target)
Transmission Control Protocol	
Source Port	63373-63374 Alternating Sequence
Destination Port	min=1 to max=65389
Sequence Number	Alternating Sequence, difference=65537 (anamoly)
Acknowledgement Number	0, correct for SYN packet
Control Flags	SYN
Window	Random selection of 4 fixed values: 1024, 2048, 3072, 4096
TCP Options	MSS=1460 bytes, 2 bytes of padding
Comments	
	#Source Port changes at same time in sequence with TCP Sequence Number shifts
	#Scans Port 80 (http) 3 times, all others only once
	#On 2nd and 3rd packet to Port 80, TCP Sequence number step jump to higher value (256x and 511.984x the base difference of 65537), then return to normal sequence
	#Window size is not a multiple of MSS
	#TCP Options only MSS, not as expected from a Windows #RST Packet sent with TTL=128

Table 2. Zenmap Fingerprint Summary

C. UNICORNSCAN

Unicornscan is a fast port scanner developed by Jack Louis [30]. We had difficulties getting the software to compile properly from source code, so the tests were run from a BackTrack 4 Pre-release version live DVD. BackTrack 4 also uses an Ubuntu Linux kernel, so there should be no behavioral differences compared to a normal Ubuntu installation [31]. The test was run as root with the command "`unicornscan 10.0.0.4`." The RUMINT illustration of the default Unicornscan is Figure 16. Note the following: multiple packet lengths, Don't Fragment Flag set, constant TTL value of 64, a large range of source ports from registered and dynamic (>1023), high concentration of well-known destination ports, with another cluster high in the dynamic range, and a broad variation in the TCP sequence numbers.

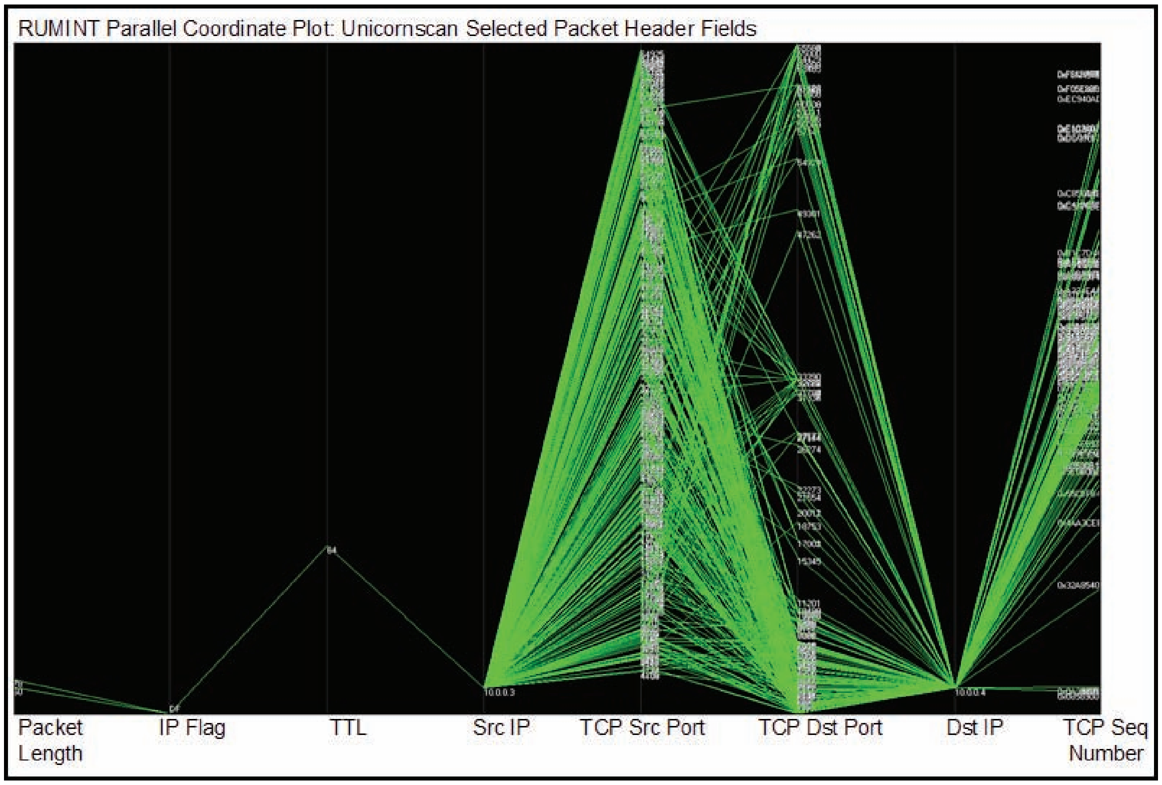


Figure 16. Unicornscan default scan in RUMINT

The two different packet lengths are caused by Unicornscan's unique large TCP Options field information, which makes each scan packet 78 bytes. By comparison, the negative response "RST" packets sent by the attacker are only 60 bytes. Unicornscan also uses a broader dynamic range of TCP source ports than the other tools with randomized TCP source ports.

Figure 17 shows Unicornscan's unique TCP Sequence Number trend. Note the random oscillation about the center range of TCP values with periodic spikes to very low values.

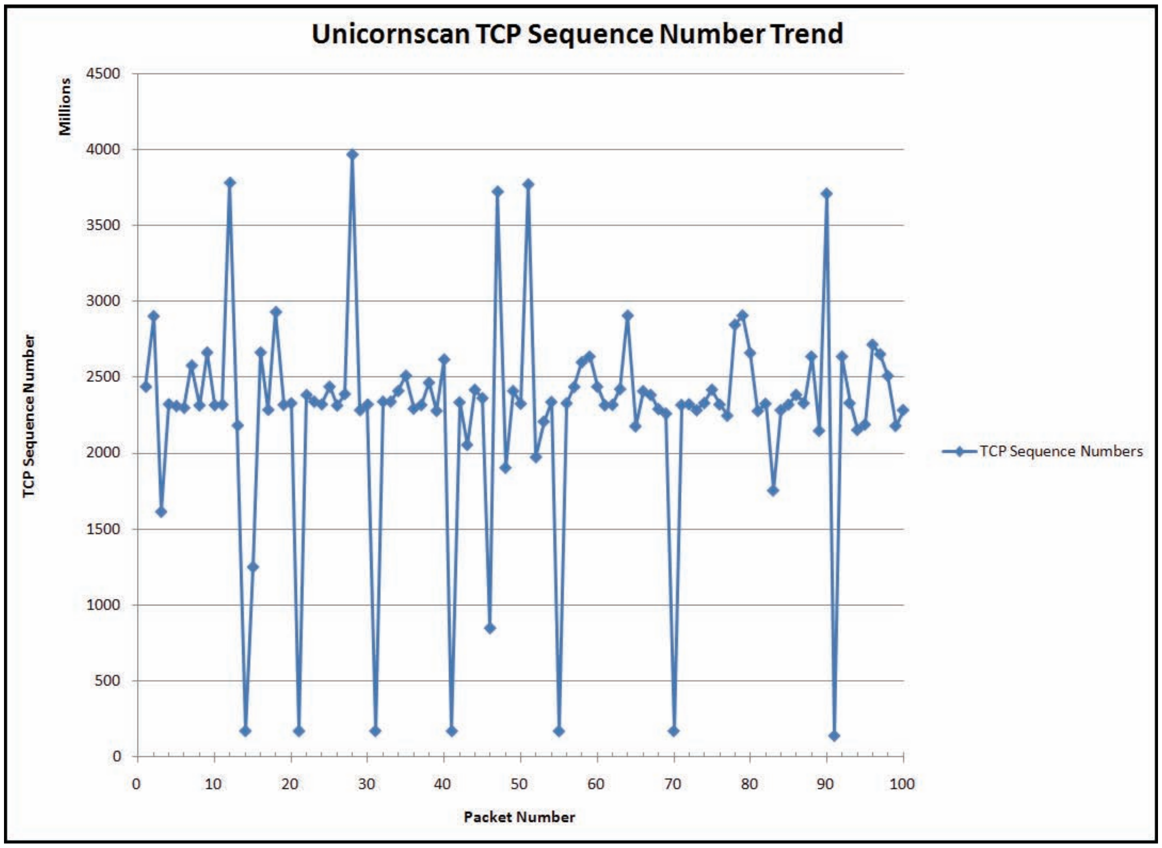
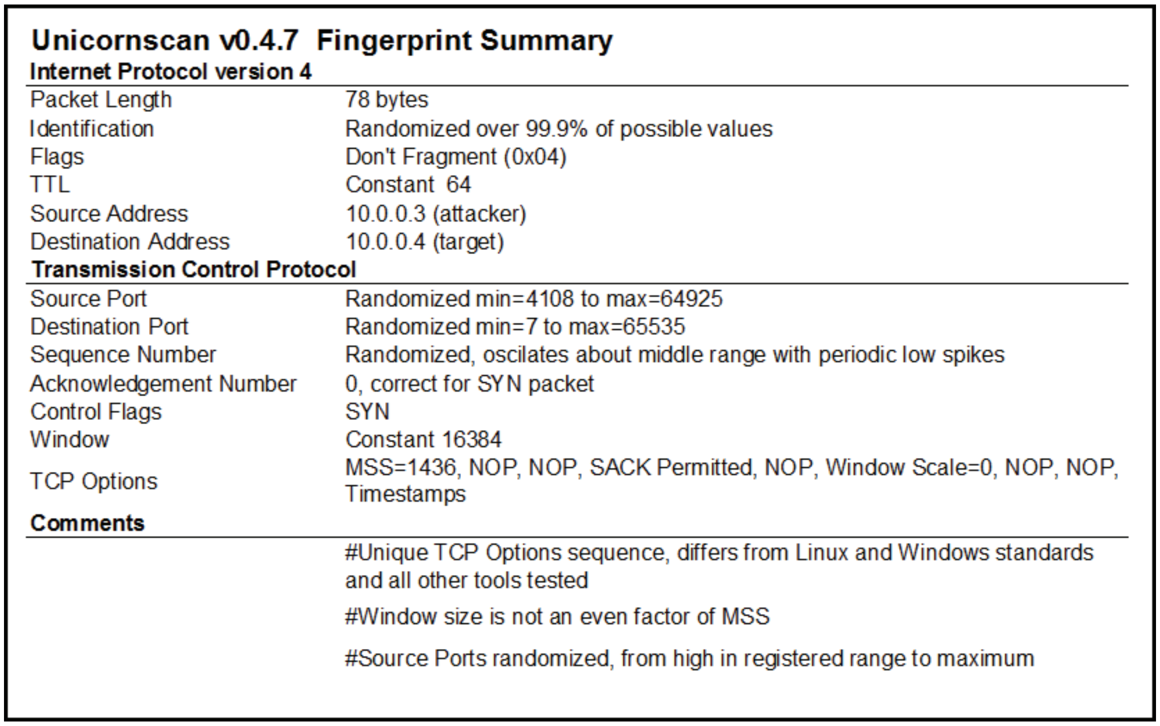


Figure 17. Unicornscan TCP Sequence Number Trend (First 100 Packets)

Unicornscan's static and dynamic fingerprint parameters are listed in Table 3.



Unicornscan v0.4.7 Fingerprint Summary

Internet Protocol version 4	
Packet Length	78 bytes
Identification	Randomized over 99.9% of possible values
Flags	Don't Fragment (0x04)
TTL	Constant 64
Source Address	10.0.0.3 (attacker)
Destination Address	10.0.0.4 (target)
Transmission Control Protocol	
Source Port	Randomized min=4108 to max=64925
Destination Port	Randomized min=7 to max=65535
Sequence Number	Randomized, oscilates about middle range with periodic low spikes
Acknowledgement Number	0, correct for SYN packet
Control Flags	SYN
Window	Constant 16384
TCP Options	MSS=1436, NOP, NOP, SACK Permitted, NOP, Window Scale=0, NOP, NOP, Timestamps
Comments	
	#Unique TCP Options sequence, differs from Linux and Windows standards and all other tools tested
	#Window size is not an even factor of MSS
	#Source Ports randomized, from high in registered range to maximum

Table 3. Unicornscan Fingerprint Summary

D. SCANRAND

Scanrand is part of the Paketto Keiretsu suite of tools developed by Dan Kaminsky [32]. By default, Scanrand requires a specified port range in its vertical scan, sequentially incrementing the destination port. Scanrand also requires a scan speed specification as a precaution to not crash the network due to the large volume of SYN scan packets. The help file example was followed as a default scan setup. The command was issued as root as follows: “`sudo scanrand –b10M 10.0.0.4:all`.” RUMINT cannot playback greater than 30,000 packets, so the scan was broken into segments for analysis. The trend is evident within the first 10,000 packets and is illustrated in RUMINT in Figure 18. Note the following: constant packet length, Don’t Fragment Flag set, TTL=255 for the scan packets, single source port, the destination port’s sequential increments are more easily observed watching the RUMINT playback progress, but the result of the first 10,000 packets can be seen, and finally a very large variation in TCP sequence numbers.

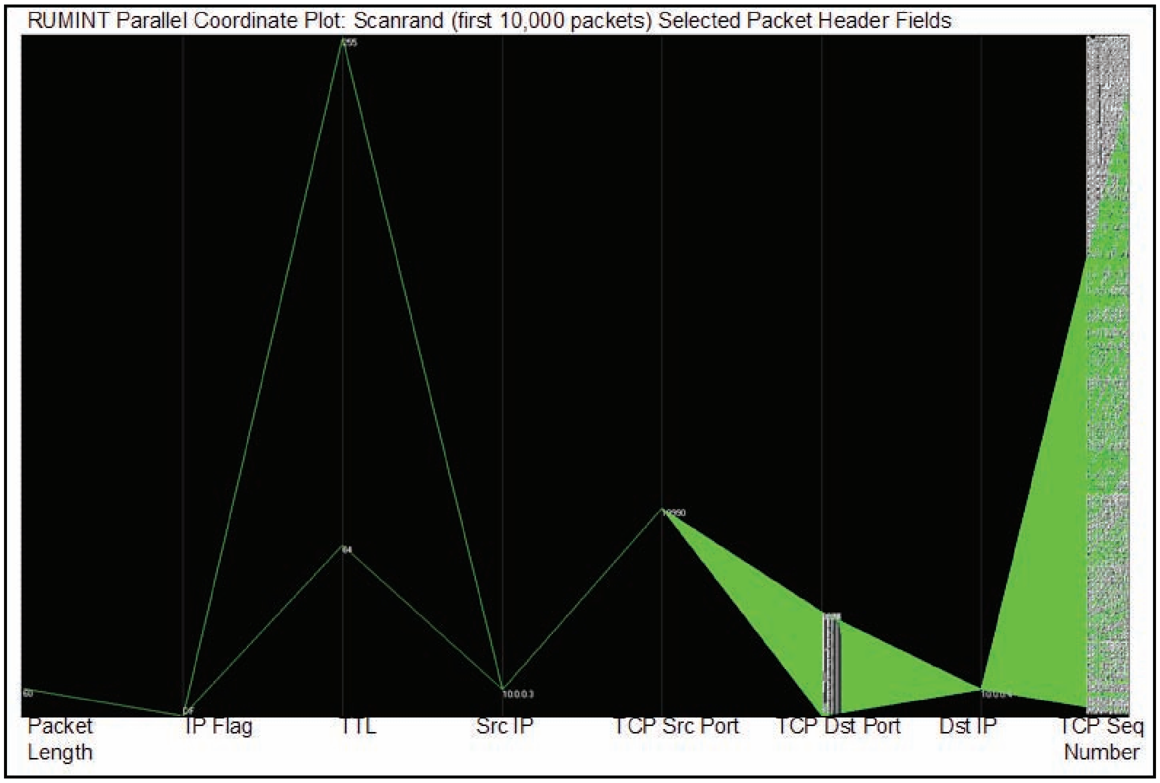


Figure 18. Scanrand first 10,000 packets in RUMINT

Similar to other tools that do not use the OS default TTL value, the RST packets conspicuously stand out from the scan packets with their TTL value of 64.

One interesting fingerprint parameter of Scanrand not shown in Figure 18 is its constant IP identification number. None of the other tools had this characteristic, so the IP identification field was considered inconsequential to the visual fingerprints. A constant or insufficiently random IP identification number would be a serious weakness in a normal OS TCP/IP stack, allowing trivial MITM attacks. Since Scanrand's goal is a reconnaissance scan and not normal TCP/IP communications, the constant IP identification field does not adversely affect the performance of the scan.

While Scanrand's IP identification number is uniquely constant, its TCP sequence number also has a unique characteristic. Scanrand's TCP sequence number is the most varied of any of the tools tested. As can be seen in Figure 19, the TCP sequence numbers span nearly the entire range of possible values (99.94% of 0 to 4,294,967,295).

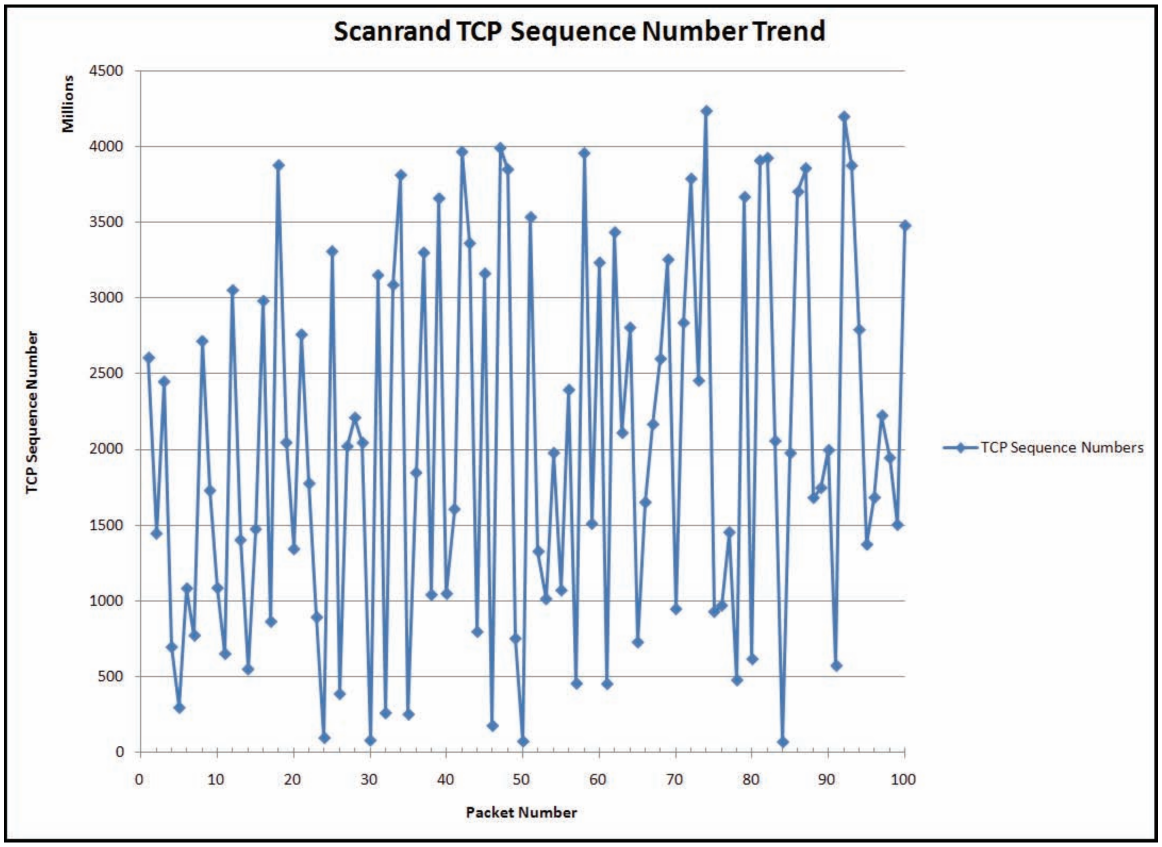
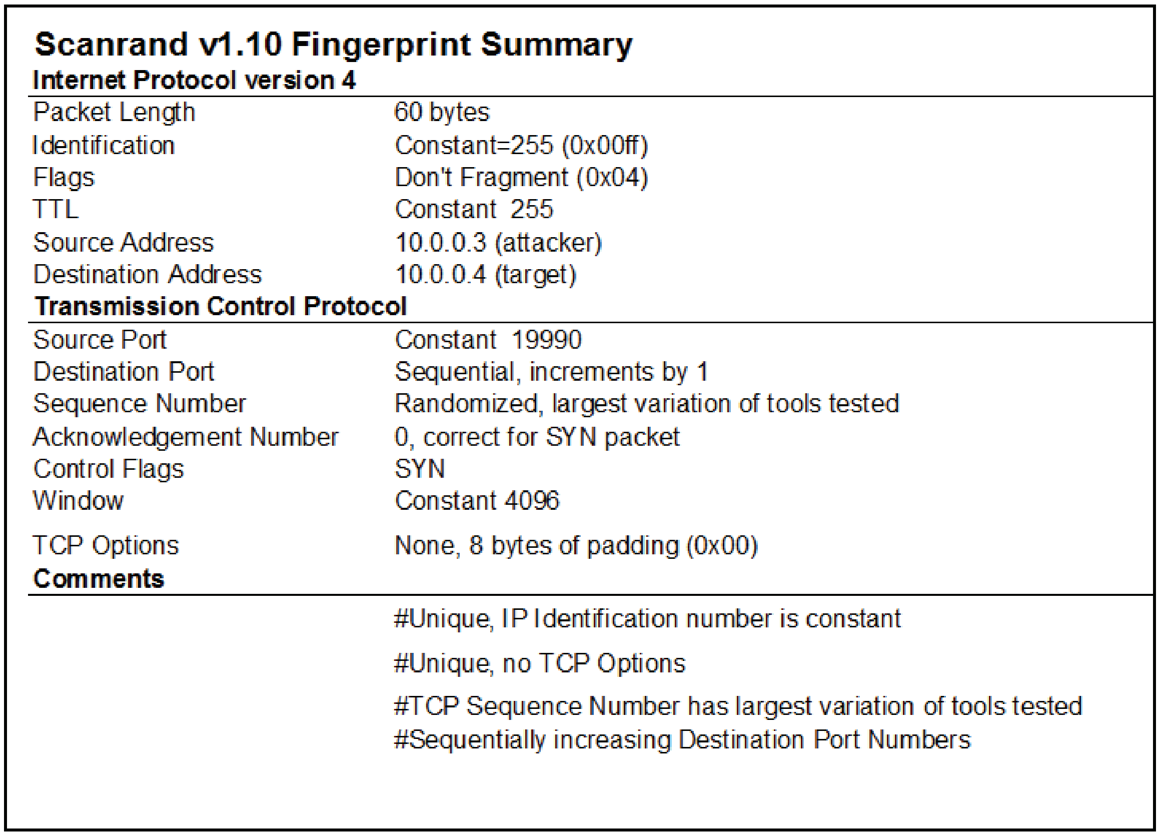


Figure 19. Scanrand TCP Sequence Number Trend (First 100 Packets)

Scanrand was also notable for its lack of TCP Options. Consequently, Scanrand's scan packets end with 8 bytes of 0x00 padding. Table 4 summarizes the fingerprint characteristics of Scanrand.



Scanrand v1.10 Fingerprint Summary

Internet Protocol version 4	
Packet Length	60 bytes
Identification	Constant=255 (0x00ff)
Flags	Don't Fragment (0x04)
TTL	Constant 255
Source Address	10.0.0.3 (attacker)
Destination Address	10.0.0.4 (target)
Transmission Control Protocol	
Source Port	Constant 19990
Destination Port	Sequential, increments by 1
Sequence Number	Randomized, largest variation of tools tested
Acknowledgement Number	0, correct for SYN packet
Control Flags	SYN
Window	Constant 4096
TCP Options	None, 8 bytes of padding (0x00)
Comments	
	#Unique, IP Identification number is constant
	#Unique, no TCP Options
	#TCP Sequence Number has largest variation of tools tested
	#Sequentially increasing Destination Port Numbers

Table 4. Scanrand Fingerprint Summary

E. STROBE

Strobe was developed by Julian Assange as a "super-optimized," fast TCP port scanner [33]. The strobe scan was initiated with the command "`sudo strobe 10.0.0.4`." The RUMINT illustration of the default Strobe scan is shown in Figure 20. Note the following: multiple packet lengths, Don't Fragment Flag set, a constant TTL value of 64, source port selected within a range, destination port sequential increments, dispersed but clustered range of TCP sequence numbers.

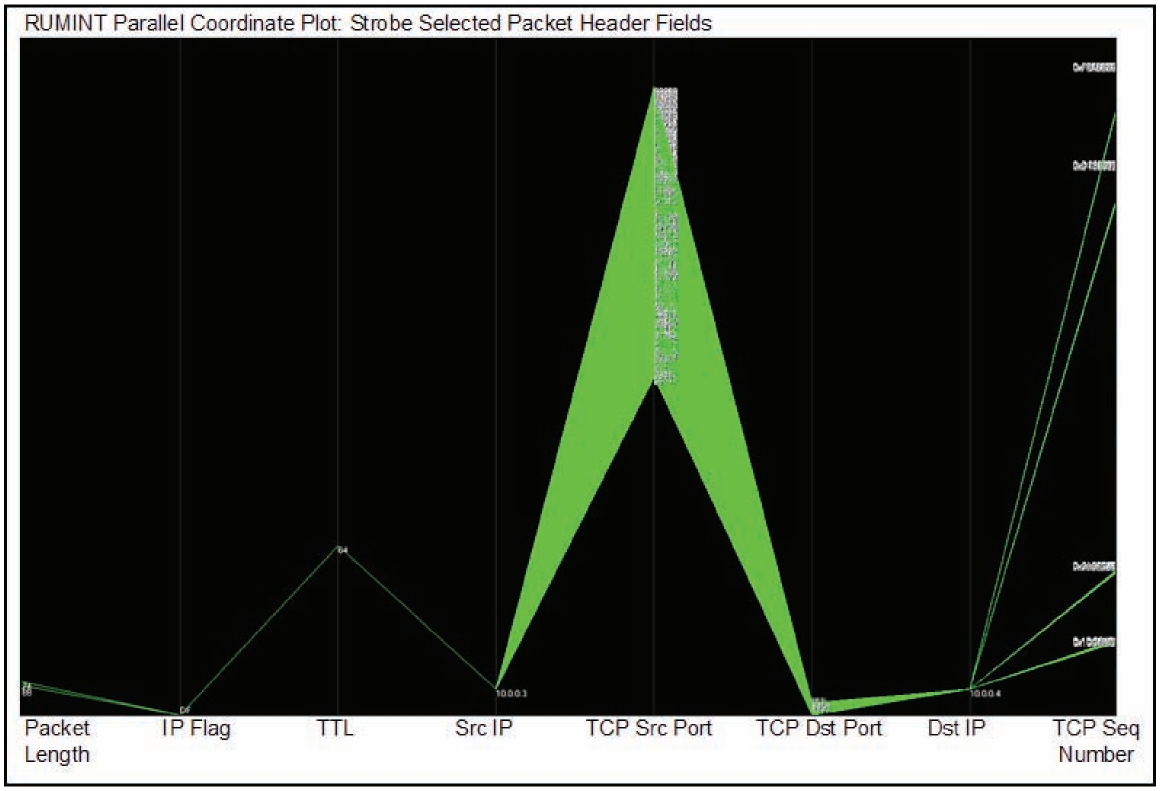


Figure 20. Strobe default scan in RUMINT

Strobe's TCP source ports were randomized in a range from a minimum of 32774 to a maximum of 60983. The multiple packet lengths highlight the difference between the SYN scan packets and the RST packets sent by the host OS. The SYN scan packets are 74 bytes long, while the RST packets are their normal 60 bytes.

Strobe has a unique TCP sequence number trend, as shown in Figure 21. It randomly oscillates in a small band but makes several large step jumps in value throughout the course of the scan.

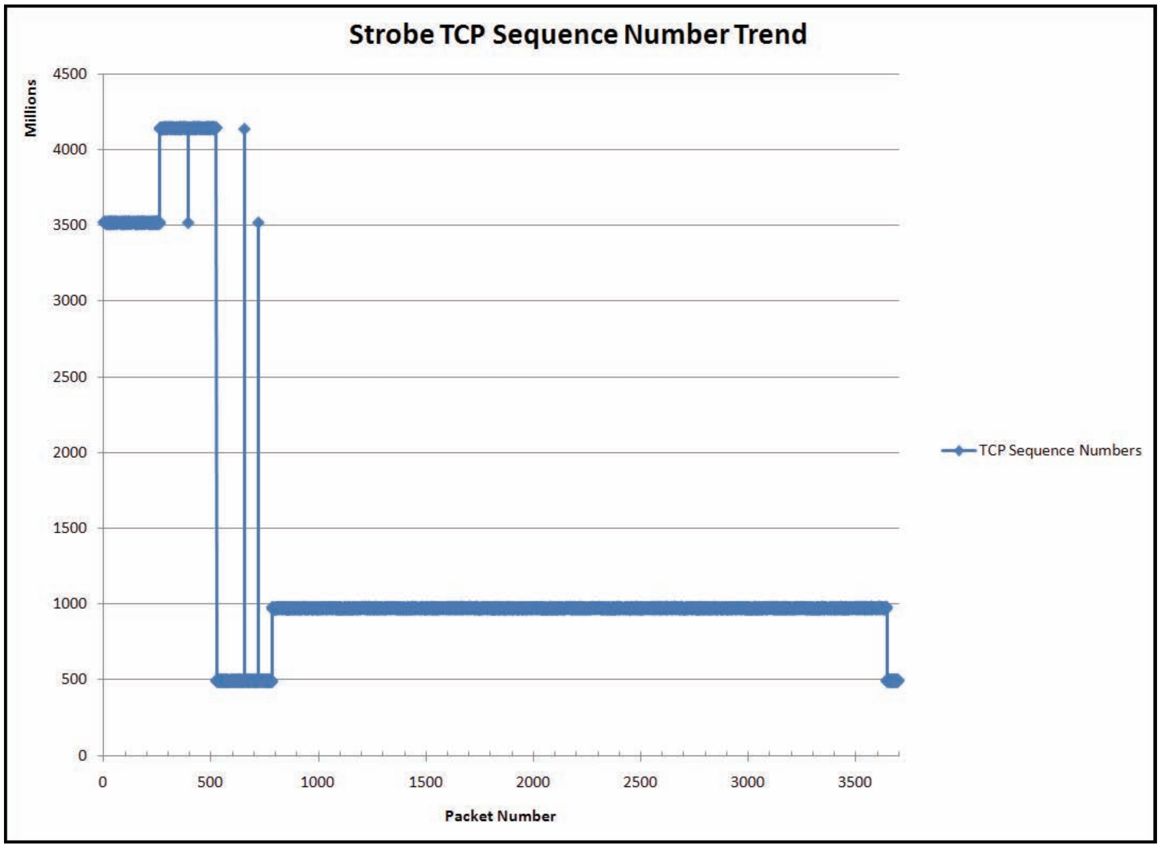


Figure 21. Strobe TCP Sequence Trend (Full Scan)

Figure 22 shows a close-up view of the random oscillations by inspecting the region that appears linear in Figure 21, between packets 1,500 and 2,000.

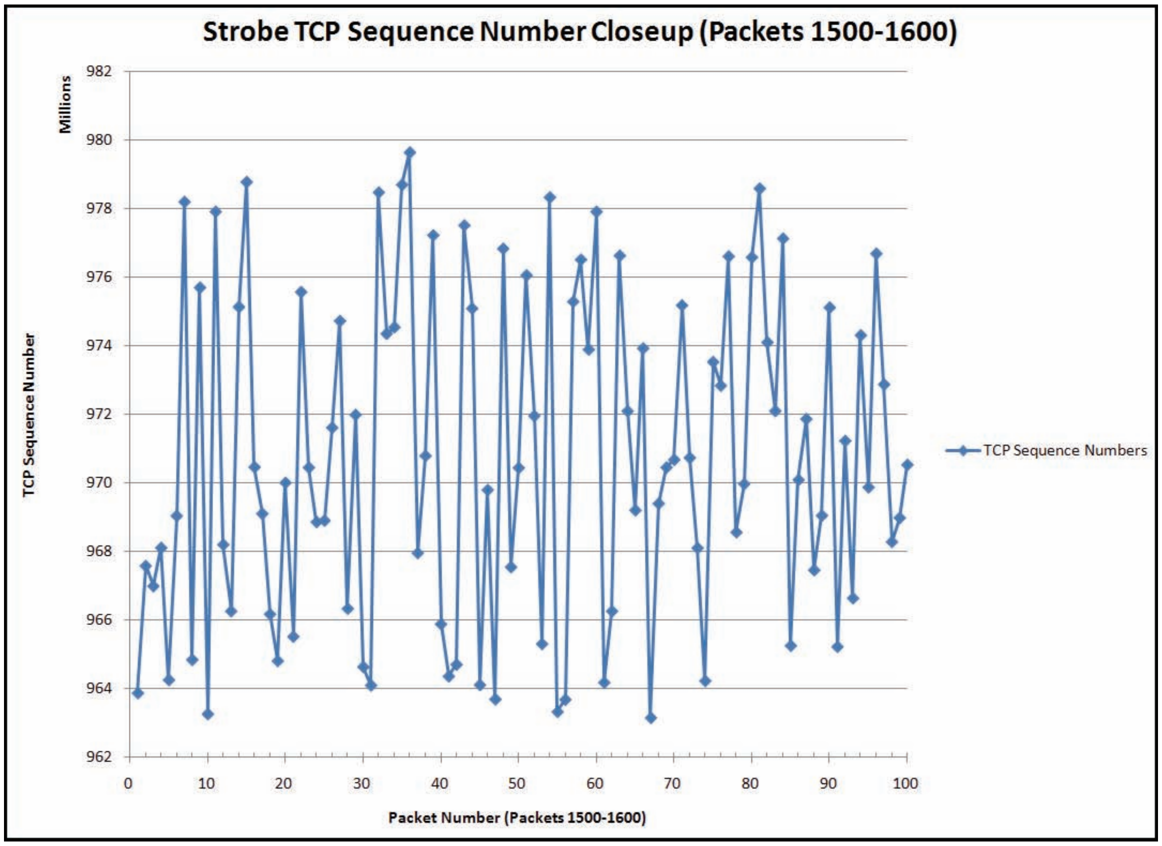


Figure 22. Strobe TCP Sequence Number Closeup

Strobe demonstrated another interesting characteristic, repeated sequential runs of destination ports. It scanned ports 1–66, then sent another ARP request for its target, then repeated the scan of those ports again. The sequence is shown in the destination port graph in Figure 23.

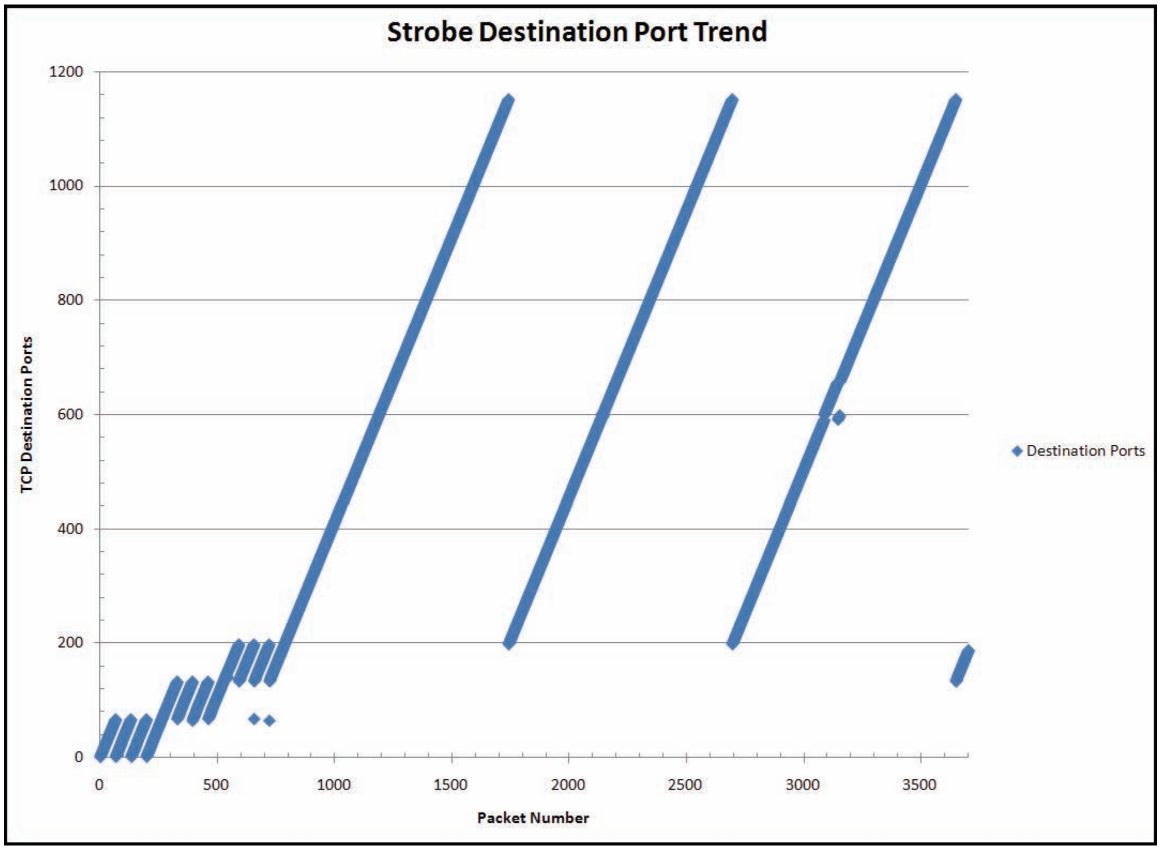


Figure 23. Strobe TCP Destination Port Trend

Table 5 summarizes the fingerprint characteristics of Strobe.

Strobe v1.06 Fingerprint Summary	
Internet Protocol version 4	
Packet Length	74 bytes
Identification	Randomized
Flags	Don't Fragment (0x04)
TTL	Constant 64
Source Address	10.0.0.3 (attacker)
Destination Address	10.0.0.4 (target)
Transmission Control Protocol	
Source Port	Randomized, min=32774 to max=60983
Destination Port	Sequential, increments by 1, several repetitive runs
Sequence Number	Randomized in small bands with large step deviations
Acknowledgement Number	0, correct for SYN packet
Control Flags	SYN
Window	Constant 5840
TCP Options	MSS=1460, SACK Permitted, Timestamps, NOP, Window scale=6
Comments	
	#Unique Destination Port Sequencing
	#Window size is an even factor of MSS, factor of 4
	#Unique TCP Sequence with small bands and large step deviations

Table 5. Strobe Fingerprint Summary

F. ANGRY IP SCANNER

Anton Keks developed the Angry IP Scanner [34]. The test scans were performed through Angry IP Scanner’s Graphical User Interface (GUI). It did not specify a default list of ports to be scanned, so a complete vertical scan from 1 to 65535 was run. Again, the capture file was parsed to 10,000 packet increments for analysis. The RUMINT visualization of the first 10,000 packets of the scan is shown in Figure 24. Note the following: the scan initiates with a UDP packet, multiple packet lengths, Don’t Fragment Flag set, constant TTL=64, randomized source port range, sequential destination port increments, and TCP sequence number increasing trend.

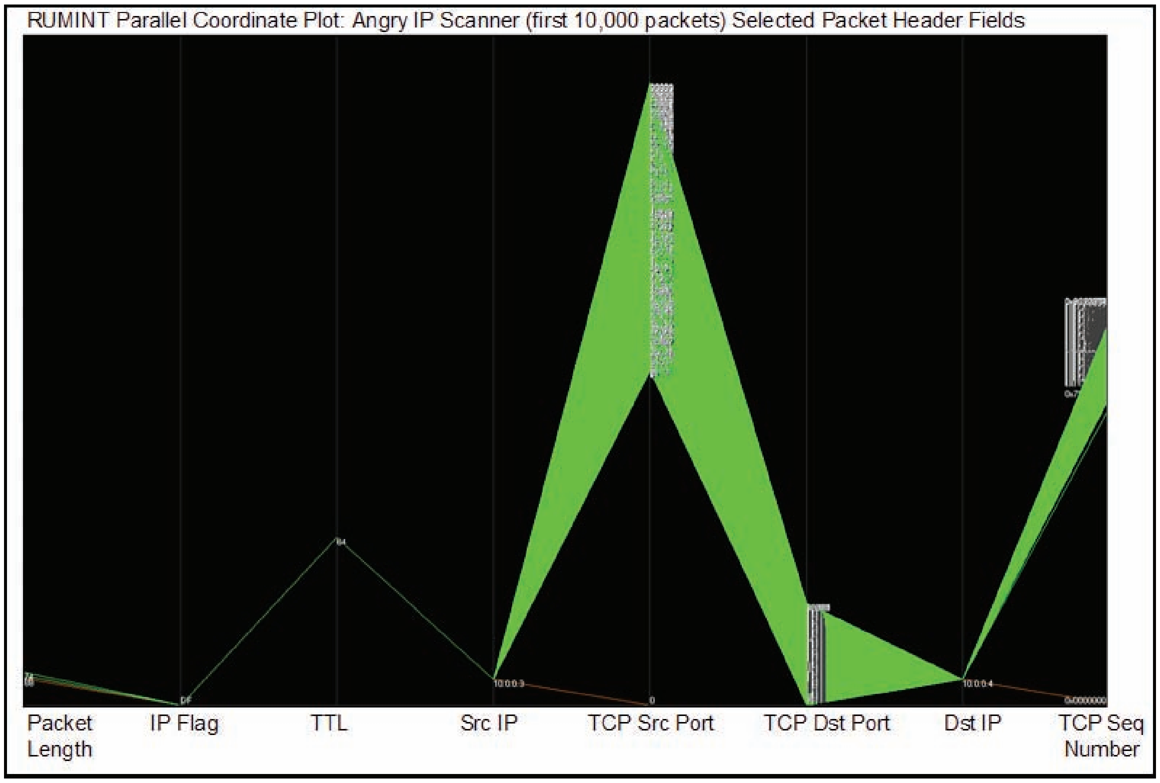


Figure 24. Angry IP Scanner first 10,000 packets in RUMINT

Similar to the previous cases, Angry IP Scanner's multiple packet lengths are caused by the SYN scan packets being 74 bytes long and the RST packets being 60 bytes long. It is unique in initiating the scan with a UDP Packet to port 33381.

Angry IP Scanner's TCP Sequence Number trend is shown in Figure 25. Note the random oscillations within a small range, but the overall trend is linear monotonically increasing.

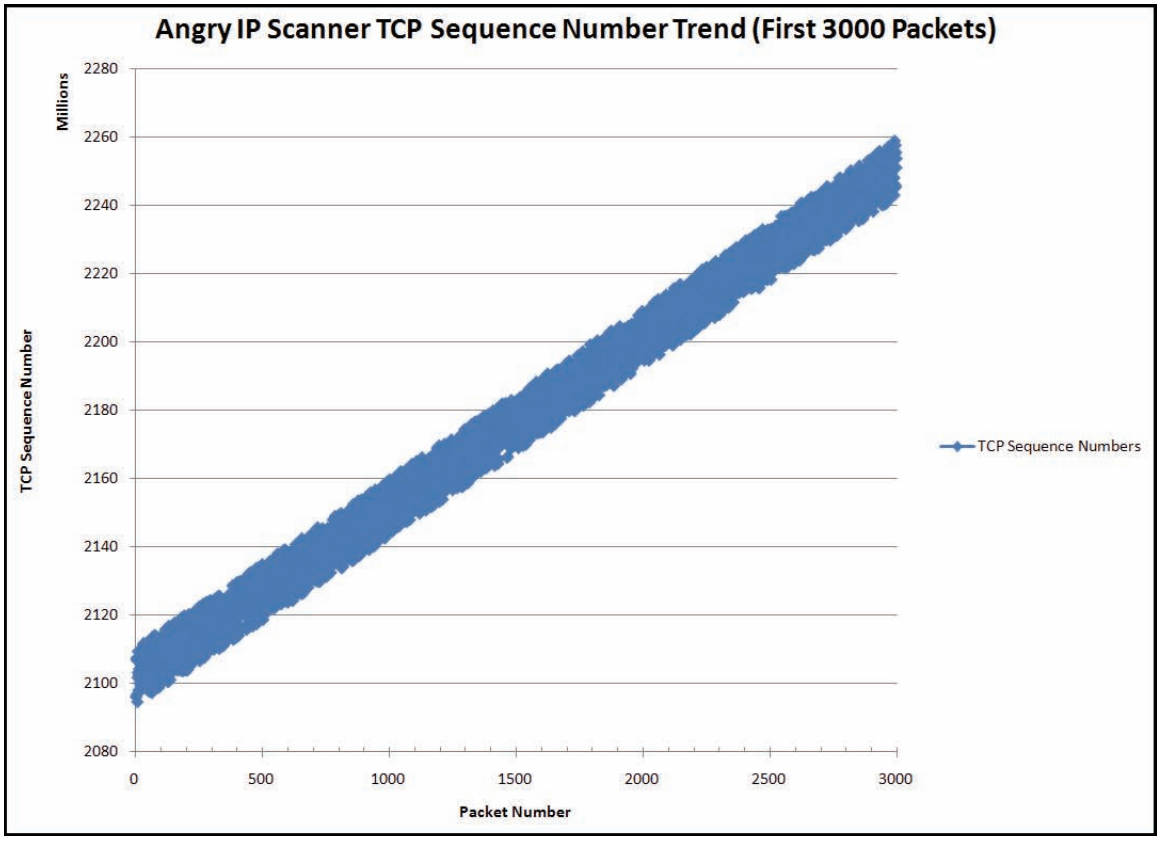
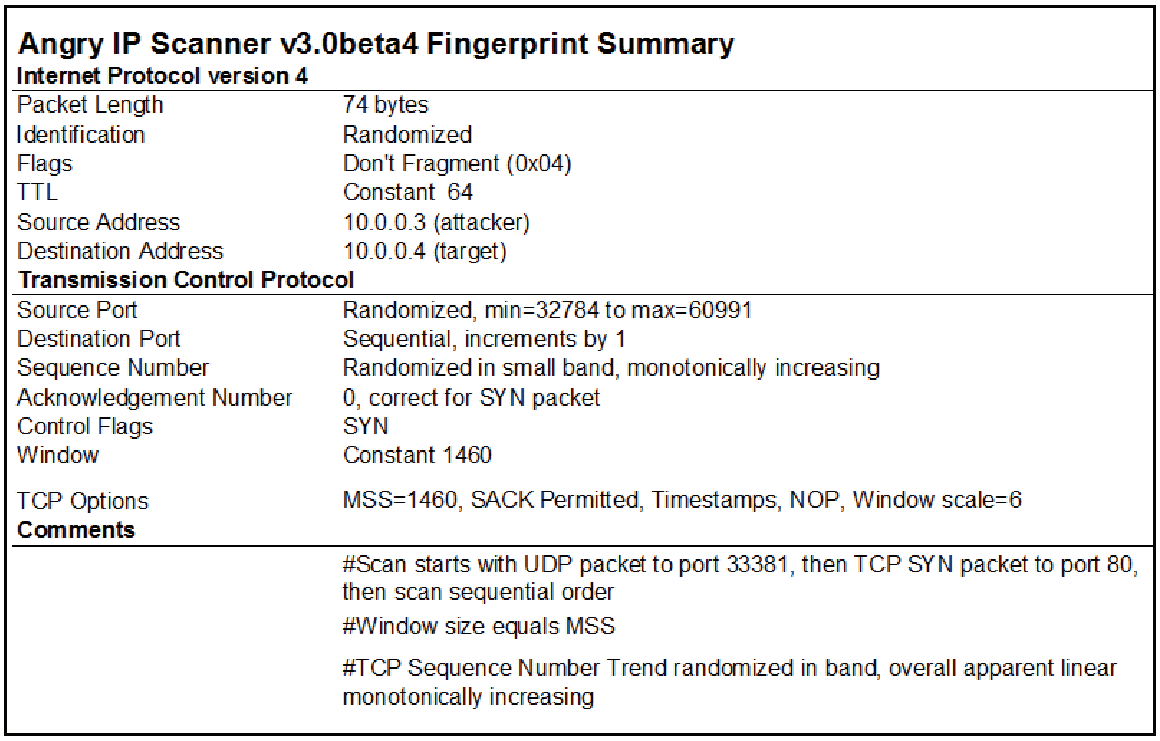


Figure 25. Angry IP Scanner TCP Sequence Number Trend (First 3000 Packets)

A summary table of the fingerprint characteristics of the Angry IP Scanner is shown in Table 6.



Angry IP Scanner v3.0beta4 Fingerprint Summary

Internet Protocol version 4	
Packet Length	74 bytes
Identification	Randomized
Flags	Don't Fragment (0x04)
TTL	Constant 64
Source Address	10.0.0.3 (attacker)
Destination Address	10.0.0.4 (target)
Transmission Control Protocol	
Source Port	Randomized, min=32784 to max=60991
Destination Port	Sequential, increments by 1
Sequence Number	Randomized in small band, monotonically increasing
Acknowledgement Number	0, correct for SYN packet
Control Flags	SYN
Window	Constant 1460
TCP Options	MSS=1460, SACK Permitted, Timestamps, NOP, Window scale=6
Comments	
	#Scan starts with UDP packet to port 33381, then TCP SYN packet to port 80, then scan sequential order
	#Window size equals MSS
	#TCP Sequence Number Trend randomized in band, overall apparent linear monotonically increasing

Table 6. Angry IP Scanner Fingerprint Summary

G. SUMMARY

The parsing and analysis process is critical to uniquely identifying the network reconnaissance tools. The methodology described above consists of three methods: visual fingerprint, TCP sequence number trend analysis, and tabulation of fingerprint characteristics.

The visualizations in RUMINT provide the fastest method to gain situational awareness regarding the reconnaissance tool's overall behavior. However, in the presence of background network traffic or multiple scans, the visualization can become occluded, necessitating further data parsing before the individual scan can be identified. The three methods developed here are complementary, allowing an analyst to make an accurate identification of the reconnaissance scan tool.

One of the most unique fingerprint characteristics of the reconnaissance tools is the TCP sequence number trend. However, it is also the most time-consuming analysis to perform. Figure 26 compares all of the TCP sequence number trends.

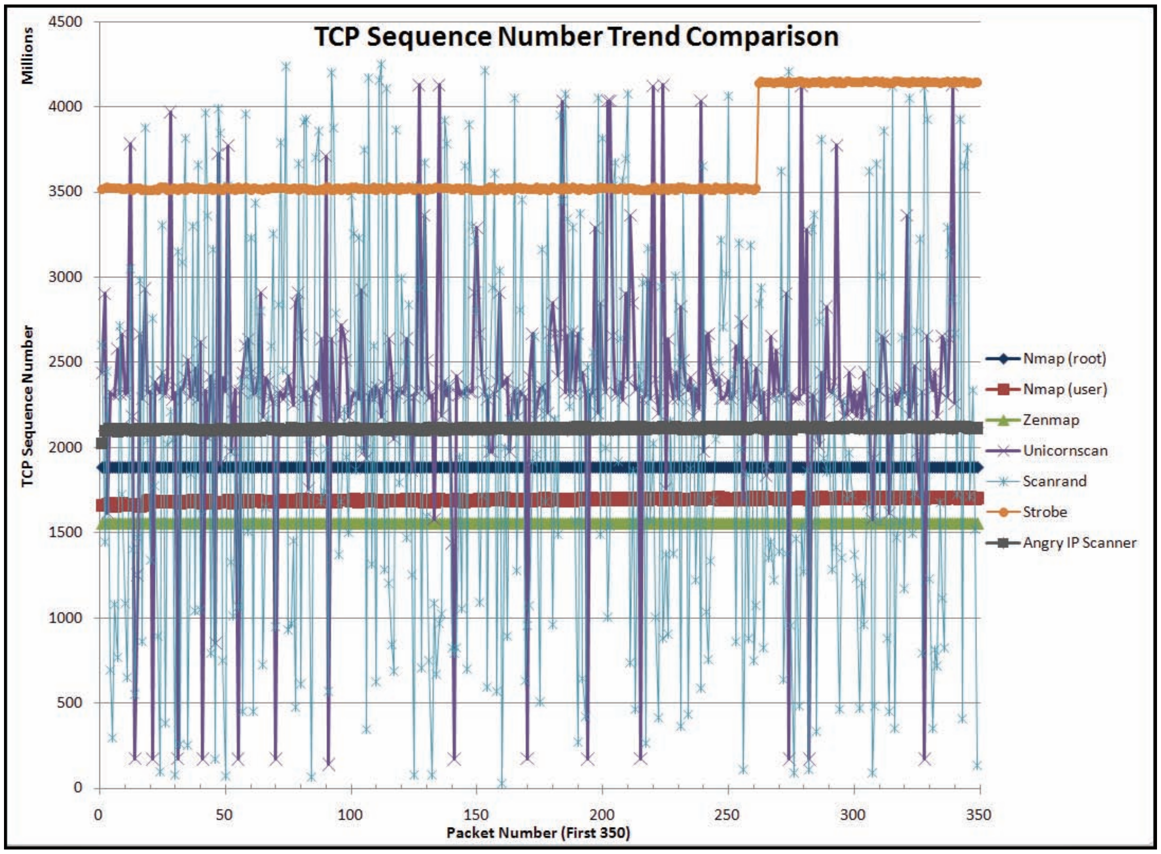
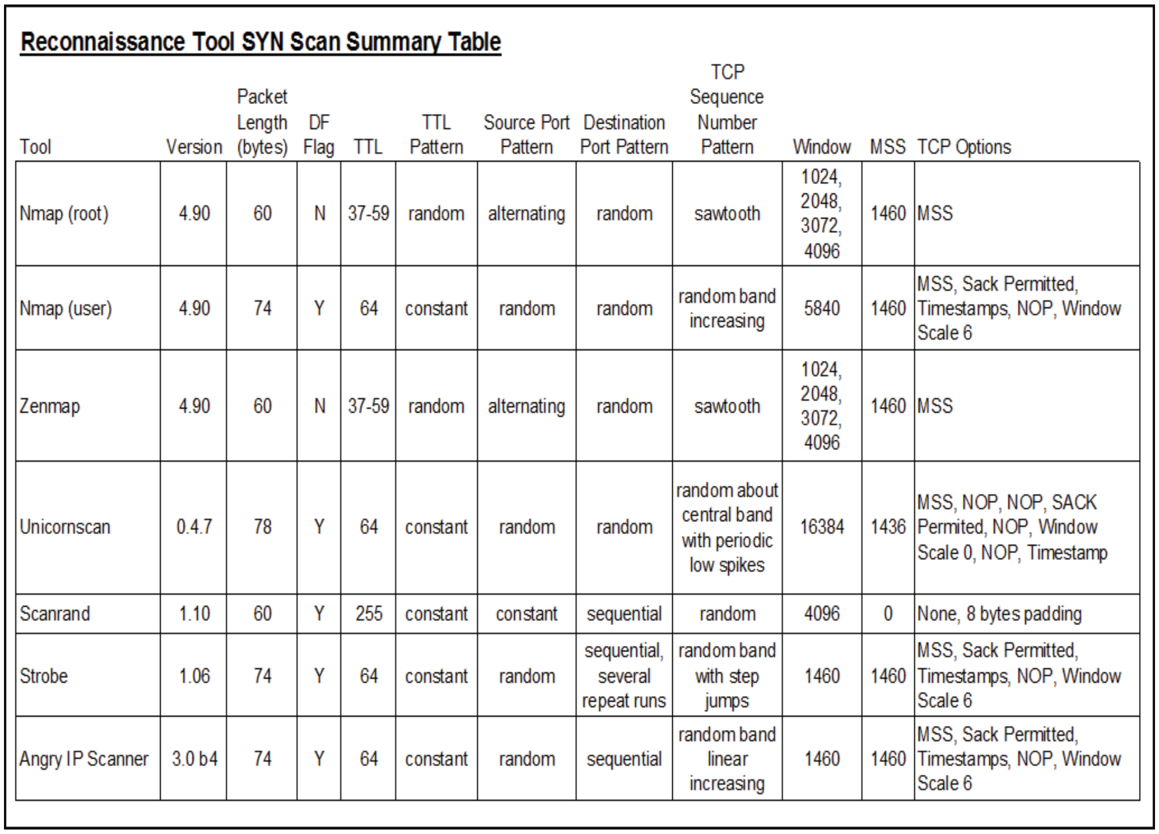


Figure 26. TCP Sequence Number Trend Comparison of All Tools Tested

Despite the large amount of data shown in Figure 26, some of the individual fingerprints can still clearly be identified. Scanrand's broad range would nearly completely occlude all other data if it were not drawn with such a small line size. Unicornscan's characteristic oscillations about the center value with periodic low-value spikes are also readily apparent. It is more difficult to discern the increasing trend of Angry IP Scanner and Nmap with user privileges. Nmap and Zenmap's distinct alternating sawtooth pattern is not visible with this scale, but it is easier to see how different that pattern is compared to any of the other tools.

Table 7 is a fingerprint characteristic summary table to compare all of the tools we tested.



Reconnaissance Tool SYN Scan Summary Table

Tool	Version	Packet Length (bytes)	DF Flag	TTL	TTL Pattern	Source Port Pattern	Destination Port Pattern	TCP Sequence Number Pattern	Window	MSS	TCP Options
Nmap (root)	4.90	60	N	37-59	random	alternating	random	sawtooth	1024, 2048, 3072, 4096	1460	MSS
Nmap (user)	4.90	74	Y	64	constant	random	random	random band increasing	5840	1460	MSS, Sack Permitted, Timestamps, NOP, Window Scale 6
Zenmap	4.90	60	N	37-59	random	alternating	random	sawtooth	1024, 2048, 3072, 4096	1460	MSS
Unicornscan	0.4.7	78	Y	64	constant	random	random	random about central band with periodic low spikes	16384	1436	MSS, NOP, NOP, SACK Permited, NOP, Window Scale 0, NOP, Timestamp
Scanrand	1.10	60	Y	255	constant	constant	sequential	random	4096	0	None, 8 bytes padding
Strobe	1.06	74	Y	64	constant	random	sequential, several repeat runs	random band with step jumps	1460	1460	MSS, Sack Permitted, Timestamps, NOP, Window Scale 6
Angry IP Scanner	3.0 b4	74	Y	64	constant	random	sequential	random band linear increasing	1460	1460	MSS, Sack Permitted, Timestamps, NOP, Window Scale 6

Table 7. Reconnaissance Tool SYN Scan Fingerprint Summary

Figure 27 illustrates a flowchart of the reconnaissance fingerprint analysis methodology.

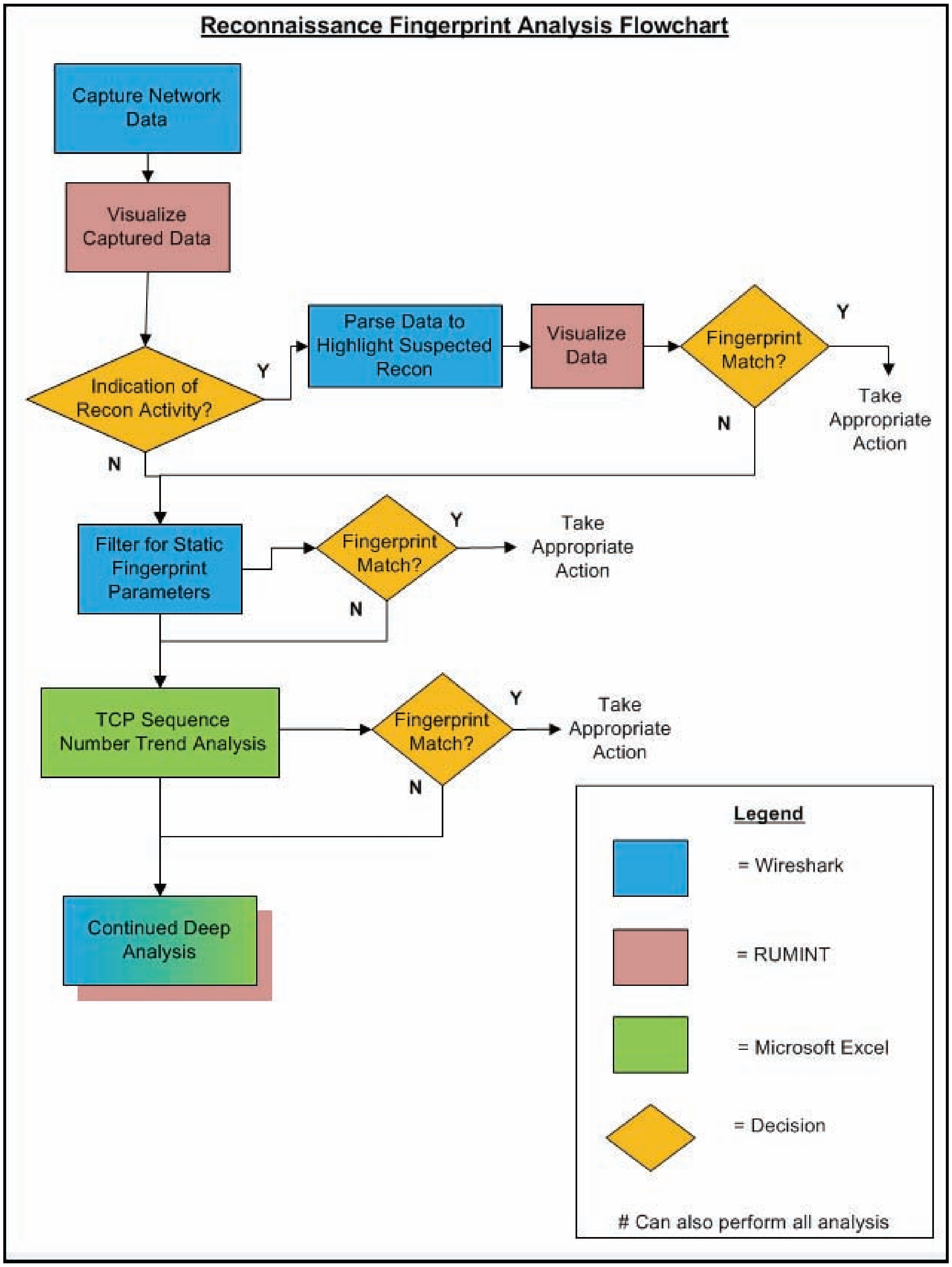


Figure 27. Reconnaissance Fingerprint Analysis Flowchart

V. CASE STUDY—NPS CYBER DEFENSE EXERCISE

A. NPS CDX'08 NETWORK SETUP

Each year since 2000, the National Security Agency's Information Assurance Directorate sponsors a Cyber Defense Exercise (CDX). The CDX is a competition between the five undergraduate service academies with the Naval Postgraduate School (NPS) and Air Force Institute of Technology (AFIT) also participating. Each school designed, built, and then attempted to defend their networks from attack by a U.S. Department of Defense (DoD) Red Team. The exercises provide hands-on experience in information assurance (IA) and real-world computer network defense (CND) for the students [35].

The NPS 2008 CDX network was setup, as shown in Figure 28.

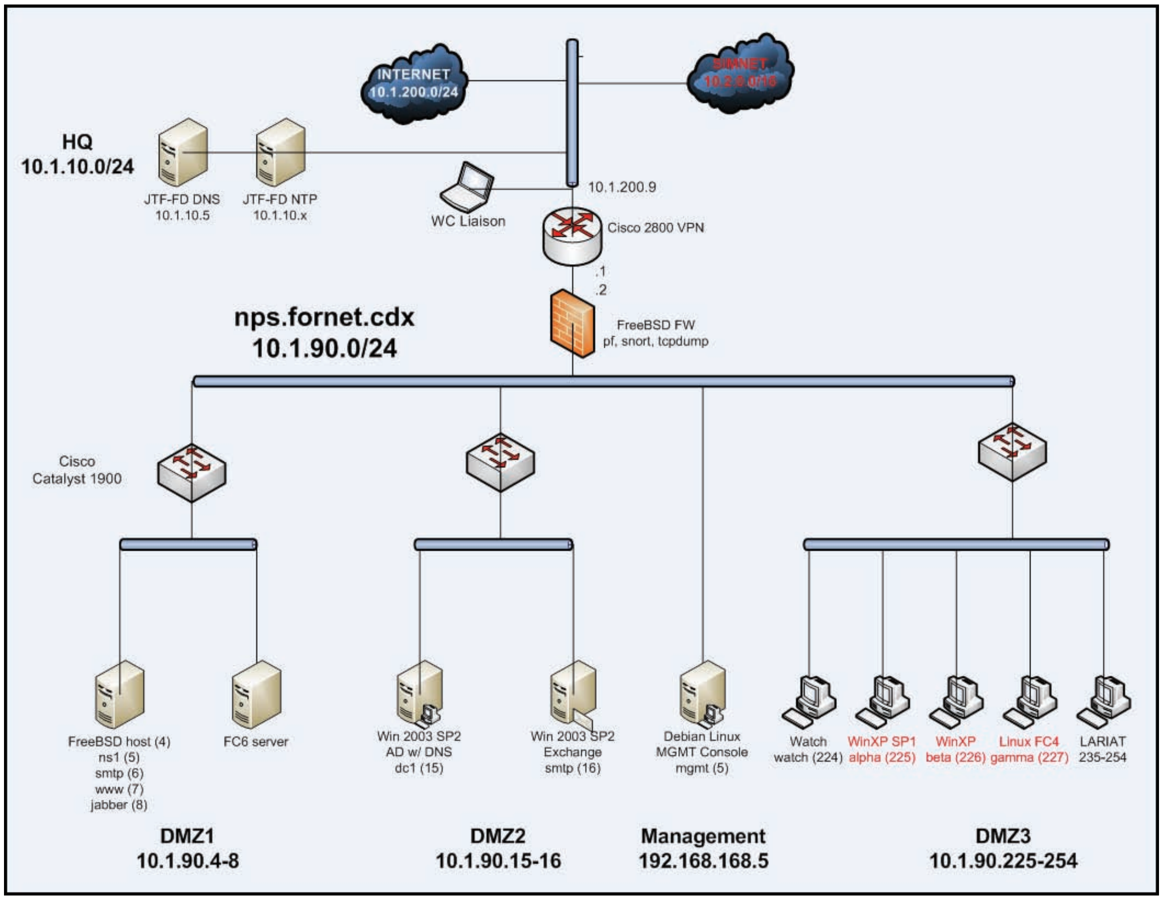


Figure 28. NPS CDX 2008 Network Diagram

B. CDX'08 DATA ANALYSIS

The data analysis began by parsing the large packet capture file to a more manageable size with Wireshark. The resulting packet capture file included the early stages of the exercise, where the adversary performed initial reconnaissance. This was determined by observing the packet capture file in RUMINT. With an indication of a reconnaissance scan in the dataset, identified by the large number of TCP Destination Ports shown in the RUMINT playback, the packet capture file was filtered to only include packets that originated outside the NPS CDX network.

From this dataset of packets originating outside the NPS CDX network, a large number of SYN packets were seen to originate from 10.2.168.245, in the adversary's dark cloud on the upper right side of the network diagram shown in Figure 28. The dataset was again parsed down in Wireshark to only include packets from that source IP address. A RUMINT visualization of the dataset at this point is shown in Figure 29.

Since most reconnaissance scans are performed using SYN packets, the dataset was filtered to only show SYN packets. There were still numerous targets, destination IP addresses, included in this dataset. It was filtered to highlight just one target that received a large number of SYN packets, 10.1.90.5. This was the IP address of the NPS CDX network's DNS server, a high value target to the adversary. The NPS DNS server is shown in the network diagram, Figure 28, in the lower left corner.

Two similar scan targets of 10.2.168.245 were identified and examined. The data and fingerprint matched that of the DNS server target data. Consequently, the rest of this example follows the analysis of the scan that targeted the NPS DNS server.

The dataset was filtered in Wireshark to display only the source IP 10.2.168.245 and destination IP 10.1.90.5 and exported to a comma-delimited value fields ".csv" file. This file was then imported into an Excel spreadsheet for further analysis. A RUMINT visualization of this dataset was also made, shown in Figure 30.

At this point, the data was analyzed to determine possible reconnaissance tool fingerprint matches.

C. IDENTIFYING LIKELY NMAP SCAN

The RUMINT illustration in Figure 29 shows the network data originating from outside the NPS CDX network, from 10.2.168.245. The large variation of destination port numbers provided a crucial initial indicator of reconnaissance scan activity. However, there is still too much data present that precludes making accurate fingerprint identification by visual analysis alone.

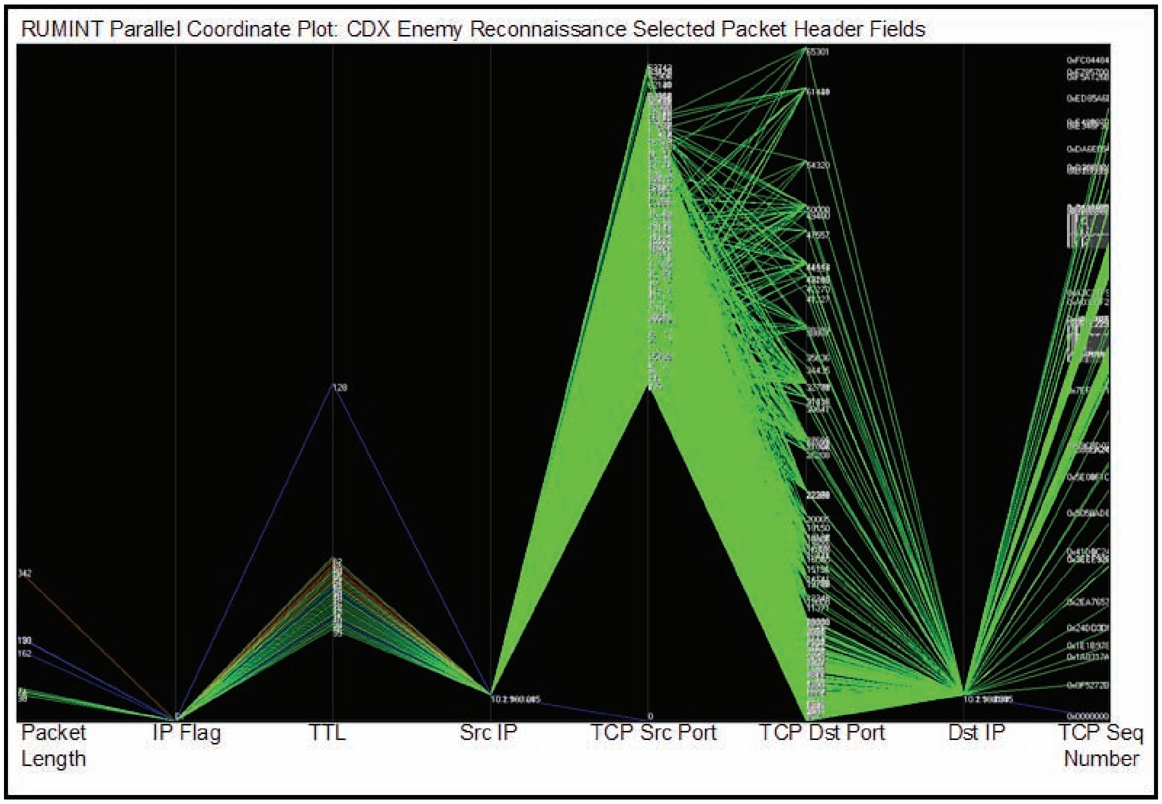


Figure 29. CDX Enemy Reconnaissance Scanning Activity in RUMINT

Next, the data was further filtered to just display the packets sent to a single target, the NPS CDX DNS server, 10.1.90.5. The RUMINT visualization is shown in Figure 30. Note the TTL value spread, small number of source ports, distribution of destination ports, and limited number of TCP sequence numbers. At this point, the analyst can begin to identify the reconnaissance tool used by visual fingerprint match.

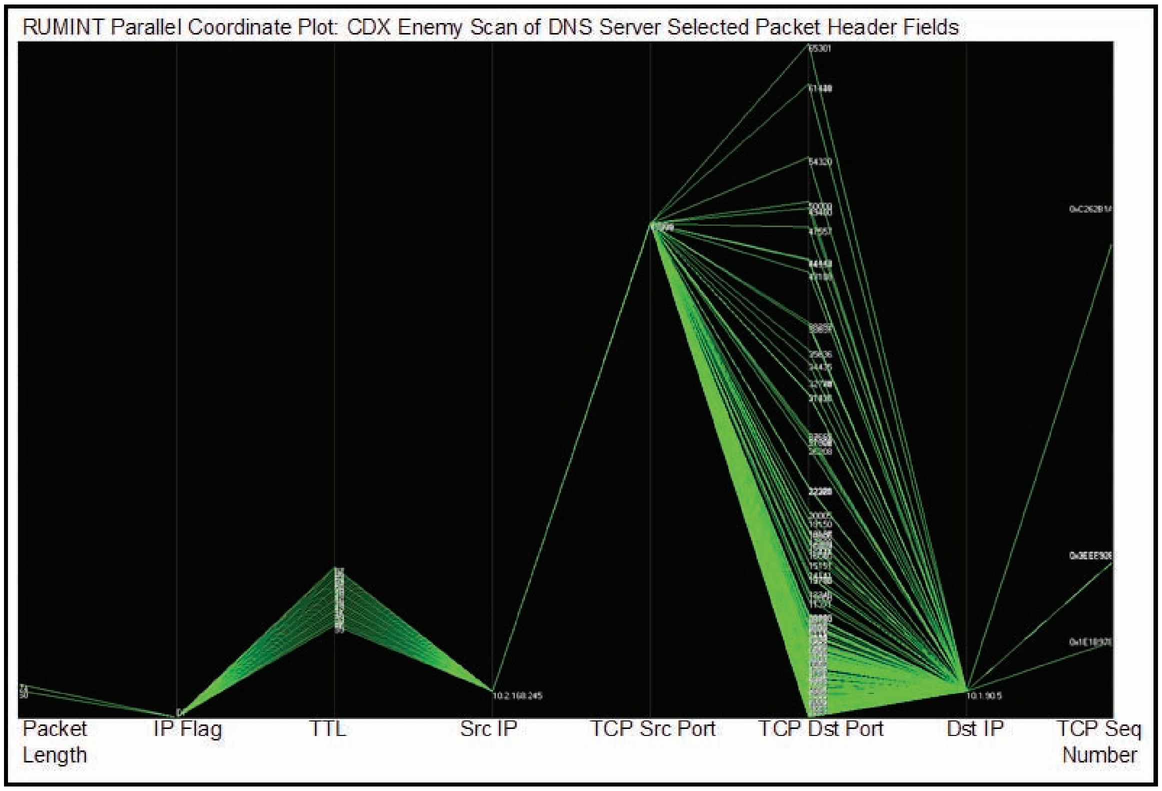


Figure 30. CDX Enemy Scan of DNS Server; Possible Nmap Scan

Based on the visual fingerprint in the previous chapter, it appears that the reconnaissance-scanning tool was Nmap. However, further specific fingerprint characteristics must be checked to validate that determination. The Nmap fingerprint summary table was used to check the CDX data, in addition to the most definitive Nmap identifier, the TCP sequence number trend. The TCP sequence number trend is shown in Figure 31.

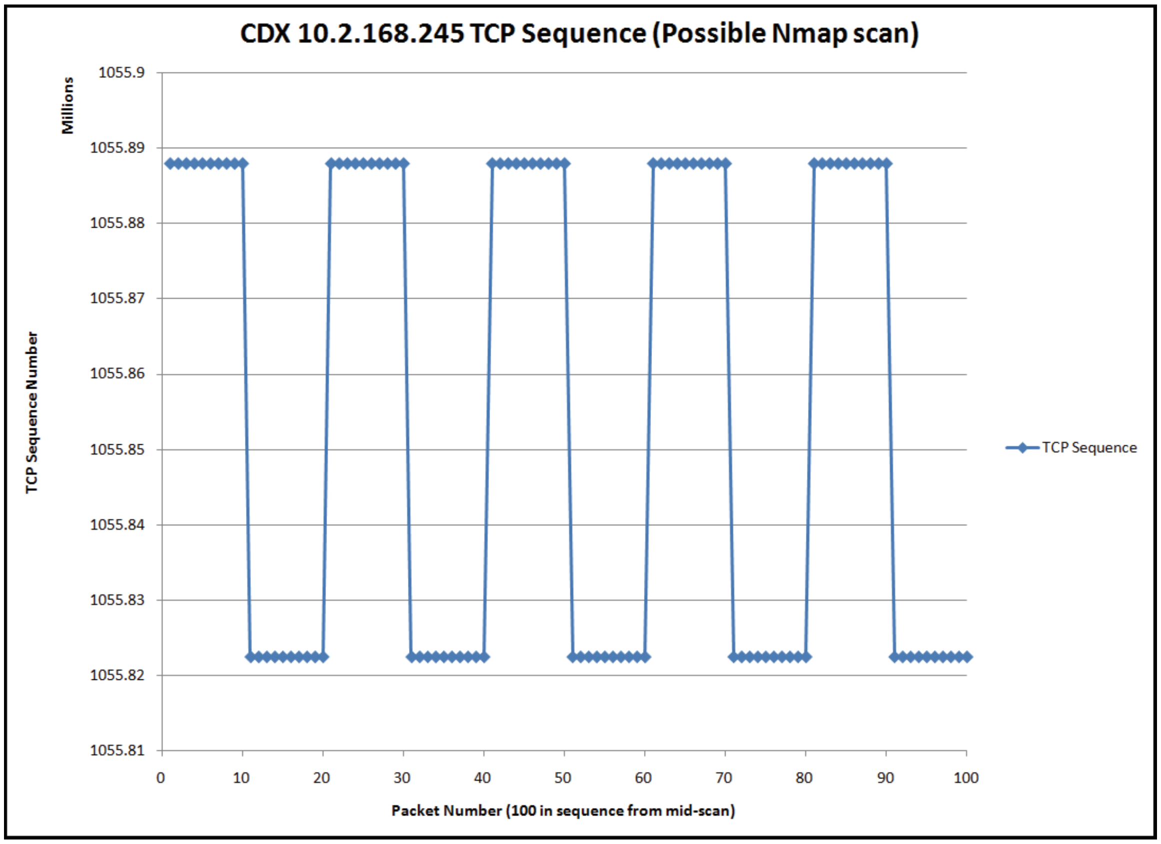
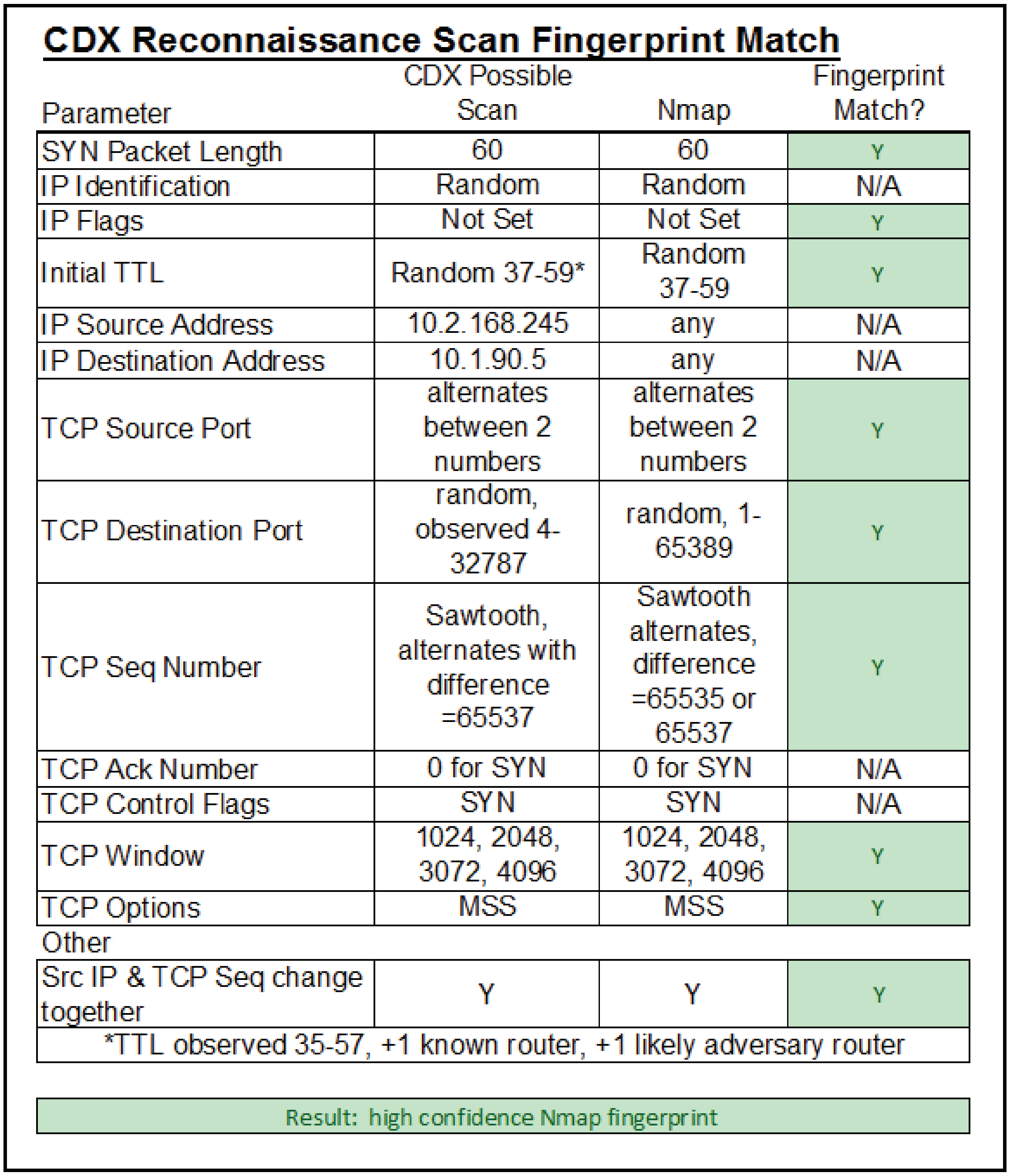


Figure 31. CDX 10.2.168.245 TCP Sequence (Possible Nmap Scan)

The distinct alternating sawtooth pattern of Nmap is readily visible. Further numerical analysis showed the difference between the alternating numbers to be 65,537. The fingerprint match of the RUMINT visualization, TCP sequence number trend analysis, and summary data table (Table 8) provides a high-confidence result that the tool the adversary used was Nmap.



CDX Reconnaissance Scan Fingerprint Match

Parameter	CDX Possible Scan	Nmap	Fingerprint Match?
SYN Packet Length	60	60	Y
IP Identification	Random	Random	N/A
IP Flags	Not Set	Not Set	Y
Initial TTL	Random 37-59*	Random 37-59	Y
IP Source Address	10.2.168.245	any	N/A
IP Destination Address	10.1.90.5	any	N/A
TCP Source Port	alternates between 2 numbers	alternates between 2 numbers	Y
TCP Destination Port	random, observed 4-32787	random, 1-65389	Y
TCP Seq Number	Sawtooth, alternates with difference =65537	Sawtooth alternates, difference =65535 or 65537	Y
TCP Ack Number	0 for SYN	0 for SYN	N/A
TCP Control Flags	SYN	SYN	N/A
TCP Window	1024, 2048, 3072, 4096	1024, 2048, 3072, 4096	Y
TCP Options	MSS	MSS	Y
Other			
Src IP & TCP Seq change together	Y	Y	Y
*TTL observed 35-57, +1 known router, +1 likely adversary router			

Result: high confidence Nmap fingerprint

Table 8. CDX Probable Nmap Scan Summary

D. CONCLUSION

This example demonstrates the analysis techniques developed in this thesis against a real-world cyber defense scenario. It is possible to uniquely identify the network reconnaissance tool used.

The analysis process also revealed more information about the adversary. The network defender knows the adversary used Nmap to scan the network. Based on the spread of TTL values, RST packets, and the Nmap scan's TTL frequency distribution, there were two routers between the adversary machine that launched the reconnaissance scans and the NPS DNS server. One router is known to be in the defender's network; therefore, the adversary is most likely also situated behind a gateway router. The RST packets' likely initial TTL of 64, Nmap's most common installation as a Linux tool, and other packets' TCP Options from 10.2.168.245, all suggest that the adversary's scanning machine is running Linux. Thus, in the course of conducting active reconnaissance, the adversary has given away information to the defender that may be useful for adjusting defenses or for military and law enforcement to take appropriate action in response to the reconnaissance activity.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

A. SUMMARY

This thesis examined the feasibility of passively fingerprinting several well-known network reconnaissance tools. The analysis methodology developed was shown to provide several different mechanisms to identify unique fingerprints. These techniques can provide valuable information to network defenders. Because the analysis is based on passive observation of network traffic, it does not alert the adversary. Information about the attacker can be collected without the attacker's knowledge. This information can be used to tailor defensive responses to the attacker's reconnaissance activities.

The results of this research can be summarized in the following main points:

- The visual fingerprints created in RUMINT confirmed the visual fingerprinting techniques first developed in [13]. RUMINT provided the most rapid indication and warning of the reconnaissance scans. This enabled the cuing of specific portions of a larger data set for further analysis.

- The analysis condensed fingerprints into static and behavioral characteristics on summary tables. This enables the network defender to make an accurate and timely identification of the reconnaissance tools being employed by the attacker.

- The analysis technique, derived from [9], introduced the application of TCP Sequence Number trend analysis to network reconnaissance tools. The TCP Sequence Number trend proved to be one of the most distinguishing aspects of each tool's fingerprint.

- Lastly, the thesis demonstrated that these techniques can be used to analyze real-world data from a Cyber Defense Exercise scenario to identify the reconnaissance tool an adversary used based on its unique fingerprint.

B. RECOMMENDATIONS

Parallel coordinate plots in visualization tools such as RUMINT are an effective method of visually displaying network data to rapidly gain situational awareness. As such, RUMINT is also a superb aid to provide the crucial early indication and warning of reconnaissance scanning activity.

Deep analysis of network traffic data is a time consuming process. Leaders must ensure plans take this time consideration into account and set realistic expectations.

If possible, record; do not throw out data on the network's external gateway. It is possible to identify what tools an adversary uses to conduct network reconnaissance. Analysis of this information can produce early indications and warning that an adversary is targeting a network or maintaining a persistent intelligence collection effort.

C. AREAS OF FUTURE RESEARCH

1. More Detailed Fingerprinting

This thesis demonstrated a number of methods to fingerprint one particular type of network reconnaissance scan; vertical TCP SYN scans. To build a more useful fingerprint database of reconnaissance tools, other tools and scan types must be examined and their fingerprints catalogued. The following areas warrant further research:

- Other TCP vertical scan types, such as ACK scans and other exotic combinations of TCP Control Flags
- ICMP, UDP, and IPv6 Scans
- Horizontal scan identification and fingerprinting
- Reconnaissance tool implementation on multiple different OSs
- Network sweep scans
- Distributed and idle scans, which attempt to hide the scan by masking the scan's source

- All of the above mentioned reconnaissance across network distance, where the attacker and target are separated by several routers
- Identifying and fingerprinting network reconnaissance in real-time
- More effective network data visualization
- Investigate pseudorandom number weaknesses, such as underlying patterns in the IP Identification field, more depth to the TCP Sequence Number trends, and specific identifiable randomization patterns in TCP Source or Destination Port selection
- Explore the feasibility of creating behavior-based attacker profiles. Analyze the tools used, sequence, and specifics of attack methodology to correlate this behavior into an attacker profile database.

2. Behavior-Based Attacker Profiles

The fingerprints developed in this thesis are only part of a broader goal: creating a database of known attacker behavioral profiles. The fingerprinting analyses demonstrated in this thesis may be expanded to cover a multitude of known reconnaissance and attack tools. Armed with this knowledge, network defenders and forensic analysts may be able to specifically identify attackers based on their reconnaissance and attack behaviors [14]. Forensic analysts have developed sophisticated capabilities to reconstruct an attacker's actions [36]. Given the degree of technical expertise required to become an expert attacker, it is likely that certain behavioral patterns emerge that could be used to identify the attacker. However, the attack specifics are often constrained by legal and other types of restrictions, impeding sharing of this type of information in the network defense community.

If Google aggregates behavioral information about individual users that may be able to be used to uniquely identify the users, similar approaches may apply to gathering information about attackers [37]. The accumulated attacker profiles would provide network defenders with key insight into the attacker's playbook. It would be apparent whether the attacker is a script-kiddie, criminal organization, terrorist group, or state-sponsored information warrior. This information could also enable law enforcement or military action, enhancing the ability of network defenders to respond appropriately.

3. Addressing False Positives

Each time network defenders create a new method or tool to better manage a network, the attackers adjust tactics to mitigate the defensive improvements. Matching network data to a fingerprint database to identify an adversary's reconnaissance tools poses the risk of creating false positives. Too many false or unsubstantiated alarms could reduce or negate the effectiveness of creating automated reconnaissance detection tools. However, the process of analyzing the network data with these improved techniques can still help the defender by reducing the unknown. The more network behavior is known in a concise manner, the better able the network defender will be to investigate and diagnose the underlying problem. False positives are a continuous issue, but can be mitigated by understanding the way the network is supposed to operate and applying some of the analysis techniques outlined in this thesis.

4. Fingerprint Aggregation and Correlation

The ease of communications on the Internet also means that attackers can strike anytime and anywhere. Both attackers and defenders can leverage the Internet's collaborative environment. Attackers can correlate information gained from attacks on a network's periphery to improve a subsequent attack on the network's core. Impediments to network defense information sharing hamper the defender's collective ability to maintain SA. With openness comes the exposure of vulnerabilities and risk, however, it is also an effective method for collectively improving defensive capabilities. This model has proven very beneficial for many of the cryptographic algorithms that provide the basis of security on the Internet [7]. The open-source collaborative model delivers the most adaptable method for aggregating and correlating network defense information like reconnaissance fingerprints. More can be done to further this collaborative effort to make network defense more time responsive, adaptable, and effective.

5. Enhancing the Spectrum of Cyberspace Operations

The attacker-defender cyberspace operations model may be conceptually limiting. Considering attacks as a penetration of a network defensive line is analogous to trench warfare. A broader spectrum of cyberspace operations is possible. Armed with improved defensive information, such as reconnaissance and attack tool fingerprints, attackers' behavioral profiles, and secure covert channel collaborative communications, a range of cyber-maneuver operations is possible. Cyberspace deception and counterintelligence operations could be employed to gain more information about adversaries and be used to potentially manipulate the adversary's decision-making process. These cyber-maneuver options enable more potential courses of action. The enhanced SA would in turn allow more appropriate and tailored responses, from law enforcement action to precision counter-strike or deception. Further detail regarding these other concepts can be found in [38]–[42].

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] W. Hewitt, *Planting the Seeds of SEAD: The Wild Weasel in Vietnam*, Thesis, School of Advanced Airpower Studies, Air University, Maxwell Air Force Base, AL, 1992.

[2] D. Schleher, *Electronic Warfare in the Information Age*. Norwood, MA: Artech House, 1999.

[3] D. Comer, *Computer Networks and Internets*. 5th ed. Upper Saddle River, NJ: Prentice Hall, 2009.

[4] E. Skoudis and T. Liston, *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses*. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2006.

[5] R. Bejtlich, *The Tao of Network Security Monitoring: Beyond Intrusion Detection*. Boston, MA: Addison-Wesley, 2005.

[6] S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed: Network Security Secrets & Solutions*. 6th ed. New York, NY: McGraw-Hill, 2009.

[7] S. Harris, *CISSP All-in-One Exam Guide*, 4th ed. New York, NY: McGraw-Hill, 2008.

[8] The Snort Project, *Snort Users Manual 2.8.4*. April, 2009. [Online]. http://www.snort.org/assets/82/snort_manual.pdf, accessed July 2009.

[9] M. Zalewski, *Silence on the Wire: A Field Guide to Passive Reconnaissance and Indirect Attacks*. San Francisco, CA: No Starch Press, 2005.

[10] M. Zalewski, "p0f.fp," Program file in *p0f version 2.0.8*, September, 2006. [Online]. http://lcamtuf.coredump.cx/p0f.shtml, accessed July 2009.

[11] A. Judd, *Improved Network Security and Disguising TCP/IP Fingerprint Through Dynamic Stack Modification*, M.S. thesis, Naval Postgraduate School, Monterey, CA, 2005.

[12] R. Marty, *Applied Security Visualization*. Boston, MA: Addison-Wesley, 2009.

[13] G. Conti and K. Abdullah, "Passive Visual Fingerprinting of Network Attack Tools," in *CCS Workshop on Visualization and Data Mining in Computer Security*, ACM 2004, pp. 45–54.

[14] G. Conti, *Security Data Visualization: Graphical Techniques for Network Analysis*. San Francisco, CA: No Starch Press, 2007.

[15] C. Burgess and R. Power, *Secrets Stolen, Fortunes Lost: Preventing Intellectual Property Theft and Economic Espionage in the 21st Century*. Rockland, MA: Syngress, 2008.

[16] R. Lee and Mandiant®, "The State of the Hack: The Advanced Persistent Threat" presented at SANS Monterey, Monterey, CA, 2008.

[17] C. Peikari and A. Chuvakin, *Security Warrior.* Sebastopol, CA: O'Reilly, 2004.

[18] J. Erickson, *Hacking: The Art of Exploitation.* 2nd ed. San Francisco, CA: No Starch Press, 2008.

[19] A. Whitaker, K. Evans, and J. Voth, *Chained Exploits: Advanced Hacking Attacks from Start to Finish.* Boston, MA: Addison-Wesley, 2009.

[20] Fyodor, "Top 100 Network Security Tools," [Online]. http://sectools.org, accessed July 2009.

[21] Foundstone, "Free Tools," [Online]. http://www.foundstone.com/us/resources-free-tools.asp, accessed July 2009.

[22] G. Lyon, *Nmap Network Scanning: Official Nmap Project Guide to Network Discovery and Security Scanning.* Sunnyvale, CA: Insecure.com, 2008.

[23] A. Orebaugh, G. Ramirez, J. Burke, G. Morris, L. Pesce, J. Wright, *Wireshark & Ethereal Network Protocol Analyzer Toolkit.* Rockland, MA: Syngress, 2007.

[24] G. Conti, "RUMINT.org," [Online]. http://www.rumint.org, accessed July 2009.

[25] "Internet Protocol," *RFC 791* [Online]. http://www.ietf.org/rfc/rfc791.txt, accessed May 2009.

[26] "Transmission Control Protocol," *RFC 793* [Online]. http://www.ietf.org/rfc/rfc793.txt, accessed May 2009.

[27] M. Zalewski, Strange Attractors and TCP/IP Sequence Number Analysis. BindView Corporation, 2001, [Online]. http://lcamtuf.coredump.cx/oldtcp/tcpseq/print.html, accessed June 2009.

[28] M. Zalewski, Strange Attractors and TCP/IP Sequence Number Analysis – One Year Later. [Online]. http://lcamtuf.coredump.cx/newtcp, accessed June 2009.

[29] Fyodor, Nmap. [Online]. http://insecure.org, accessed July 2009.

[30] J. Louis, Unicornscan. [Online]. http://www.unicornscan.org, accessed May 2009.

[31] Back Track 4 Pre-Release. [Online]. http://www.remote-exploit.org/backtrack_download.html, accessed July 2009.

[32] D. Kaminksy, Scanrand. [Online]. http://www.doxpara.com, accessed May 2009.

[33] J. Assange, Strobe: Super Optimized TCP Port Surveyor, [Online]. http://linux.maruhn.com/sec/strobe.html, accessed May 2009.

[34] A. Keks, Angry IP Scanner. [Online]. http://angryziber.com/w/Home, accessed May 2009.

[35] National Security Agency Press Release "8th Annual Cyber Defense Exercise," April 17, 2008, [Online]. http://www.nsa.gov/public_info/press_room/2008/cdx.shtml, accessed July 2009.

[36] K. Jones, R. Bejtlich, and C. Rose, *Real Digital Forensics: Computer Security and Incident Response.* Upper Saddle River, NJ: Addison-Wesley, 2006.

[37] G. Conti, *Googling Security: How Much Does Google Know About You?* Boston, MA: Addison-Wesley, 2008.

[38] S. Paxton, *Enhanced Cyberspace Defense with Real-Time Distributed Systems Using Covert Channel Publish-Subscribe Broker Pattern Communications.* M.S. Thesis, Naval Postgraduate School, Monterey, CA, 2008.

[39] P. Erdie, *Network-Centric Strategic-Level Deception.* M.S. thesis, Naval Postgraduate School, Monterey, CA, 2004.

[40] T. Wingfield, J. Michael, and D. Wijesekera, "Optimizing Lawful Responses to Cyber Intrusions," in *Proceedings of the 10th International Command and Control Research and Technology Symposium*, 2005.

[41] D. Julian, N. Rowe, and J. Michael, "Experiments with Deceptive Software Responses to Buffer-Overflow Attacks," in *Proceedings of the 2003 IEEE Workshop on Information Assurance,* 2003, pp. 43–44.

[42] J. Michael, "On the Response Policy of Software Decoys: Conducting Software-based Deception in Cyber Battlespace," in *Proceedings of the 26th Annual International Computer Software and Applications Conference,* 2002.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. NETWARCOM / N1 / N17
   Norfolk, Virginia

4. Professor Dan Boger
   Naval Postgraduate School
   Monterey, California

5. Professor James B. Michael
   Naval Postgraduate School
   Monterey, California

6. Associate Professor Raymond R. Buettner
   Naval Postgraduate School
   Monterey, California

7. Assistant Professor Greg Conti
   United States Military Academy
   West Point, New York

8. Senior Lecturer Scott Coté
   Naval Postgraduate School
   Monterey, California

9. Senior Security Engineer Dave Dittrich
   University of Washington
   Seattle, Washington

10. Mr. Andy Shaw
    Naval Postgraduate School
    Monterey, California