NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

FINGERPRINTING IPV4 AND IPV6 ROUTERS USING ICMP

by

Wesley G. Bofman and Fernando Maniego

September 2019

Thesis Advisor:	Robert Beverly
Second Reader:	Alan B. Shaffer

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE FINGERPRINTING IPV4 AND IPV6 ROUTERS USING ICMP	**5. FUNDING NUMBERS**
6. AUTHOR(S) Wesley G. Bofman and Fernando Maniego	
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 project reassesses and expands on a simple fingerprinting method for Internet Protocol version 4 (IPv4) routers, and extends that methodology to Internet Protocol version 6 (IPv6) routers. The initial methodology, developed by Vanaubel, Pansiot, Merindol, and Donnet, utilized initial time to live (iTTL) values derived from Internet Control Message Protocol (ICMP) echo-reply and TTL exceeded messages. The current project used ICMP echo-reply and destination unreachable/port unreachable, combined with a third iTTL value derived from ICMP timestamp messages, to strengthen the fingerprint. We adapted the methodology to IPv6-enabled routers using the initial hop limit (iHL) values from ICMPv6 echo-reply and destination unreachable/port unreachable messages. The main goal of this project is to develop a simple fingerprinting technique to identify IPv4 and IPv6 router platforms. We were able to successfully expand the previously developed IPv4 router fingerprint using the ICMP timestamp reply message. Using this fingerprinting methodology, Juniper routers can be identified. However, this fingerprinting technique cannot distinguish between Cisco and Huawei routers. With IPv6, it became evident that most routing devices follow the recommended iHL value of 64 (RFC 1700). Thus, our methodology cannot distinguish between IPv6 routing devices. We recommend additional analysis of Cisco and Huawei devices running IPv4 to identify differences in activity, as well as further research into IPv6 routers.

14. SUBJECT TERMS IPv4, IPv6, ICMP, ICMPv6, initial TTL, initial Hop Limit, fingerprinting, router signatures, network discovery	**15. NUMBER OF PAGES** 59
	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

THIS PAGE INTENTIONALLY LEFT BLANK

FINGERPRINTING IPV4 AND IPV6 ROUTERS USING ICMP

Wesley G. Bofman
Chief Petty Officer, United States Navy
BS, American Military University, 2016

Fernando Maniego
Chief Petty Officer, United States Navy
BS, Mapua Institute of Technology, 2000
BS, University of Maryland University College, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED CYBER OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
September 2019

Approved by: Robert Beverly
Advisor

Alan B. Shaffer
Second Reader

Dan C. Boger
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This project reassesses and expands on a simple fingerprinting method for Internet Protocol version 4 (IPv4) routers, and extends that methodology to Internet Protocol version 6 (IPv6) routers. The initial methodology, developed by Vanaubel, Pansiot, Merindol, and Donnet, utilized initial time to live (iTTL) values derived from Internet Control Message Protocol (ICMP) echo-reply and TTL exceeded messages. The current project used ICMP echo-reply and destination unreachable/port unreachable, combined with a third iTTL value derived from ICMP timestamp messages, to strengthen the fingerprint. We adapted the methodology to IPv6-enabled routers using the initial hop limit (iHL) values from ICMPv6 echo-reply and destination unreachable/port unreachable messages. The main goal of this project is to develop a simple fingerprinting technique to identify IPv4 and IPv6 router platforms. We were able to successfully expand the previously developed IPv4 router fingerprint using the ICMP timestamp reply message. Using this fingerprinting methodology, Juniper routers can be identified. However, this fingerprinting technique cannot distinguish between Cisco and Huawei routers. With IPv6, it became evident that most routing devices follow the recommended iHL value of 64 (RFC 1700). Thus, our methodology cannot distinguish between IPv6 routing devices. We recommend additional analysis of Cisco and Huawei devices running IPv4 to identify differences in activity, as well as further research into IPv6 routers.

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

ICMP	Internet Control Message Protocol
ICMPv4	Internet Control Message Protocol version 4
ICMPv6	Internet Control Message Protocol version 6
IOS	Internetwork Operating System
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
iTTL	initial time-to-live
MPLS	Multiprotocol Label Switching
OS	Operating System
OSI	Open Source Interconnect
RFC	Request for Comments
TCP	Transport Control Protocol
TTL	time-to-live
VM	virtual machine
UDP	User Datagram Protocol

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

We extend a sincere thank you to Dr. Robert Beverly. We have learned a great deal through this process, and would not have been successful without his guidance. We would also like to thank Dr. Alan Shaffer for his assistance with this project. Finally, we would like to thank our families for continuous support and patience as we worked through this program of study.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Fingerprinting is the process of identification by collecting and combining small pieces of information associated to an object to find unique properties of that object. Within the domain of computer networking, fingerprinting is often focused on identifying a host operating system (OS) for reasons that range from research to malicious intent. Operating systems of networking equipment have distinct responses to probing techniques due to differences in engineering design and implementation. For instance, recommendations from Request for Comments (RFCs) can lead to compatible implementations, but have implementation-specific unique features that can leak information. These differences can then be aggregated and correlated to establish a baseline fingerprint.

The value of fingerprinting has proven useful within network administration, reconnaissance, and analysis. For network equipment, in particular, fingerprinting has been useful in the areas of network enumeration, administration, and troubleshooting. Fingerprinting network equipment can provide an in-depth understanding of a particular internetwork, or sub-network (subnet), by providing information regarding traffic flow between nodes. Fingerprinting can also be an important part of network defense by providing the ability to determine vulnerable nodes.

The Internet Control Message Protocol (ICMP) as a mechanism for fingerprinting has been researched extensively as a means to determine, for instance, the version of Microsoft Windows running on a system. Methods of fingerprinting hosts developed by Ofir Arkin combine specific elements of the ICMP response message with the values of particular fields in the IP header [1]. His work became an authoritative source on fingerprinting using ICMP. Xprobe2 is an OS fingerprinting tool developed by Arkin, that is based on utilizing ICMP for scanning devices [2]. What makes it a valuable tool, for both offensive and defensive use, is its ability to provide host detection, service detection, network topology mapping, and OS fingerprinting [2]. Given the wide possibilities of what an ICMP probe can do, the use of ICMP as a mechanism for fingerprinting public routers across the Internet is a compelling reason to explore it further.

Popular scanning and probing tools such as Nmap [3] and Xprobe2 [2], when used in conjunction with one another, can generate a distinct fingerprint which, especially for routers, can be "signatured" for filtering. In contrast, ICMP, when used independently, is a relatively benign networking protocol. Further, ICMP is a valuable networking and troubleshooting tool, therefore security analysts and administrators may be more inclined to permit ICMP traffic. We thus hypothesize that carefully crafted ICMP probing techniques are a valuable fingerprinting tool.

A. MOTIVATION

Significant work has been done on OS fingerprinting using innocuous tools such as ICMP. For example, Arkin et. al [1] wrote a definitive book on ICMP usage in scanning that focuses primarily on OS fingerprinting. A comparative study, on which this project is based, was conducted to fingerprint routers based on the time-to-live (TTL) of routers. Although focused only on IPv4 routers, Vanaubel et al. [4] pointed out that the ability to fingerprint routers can be useful in determining if networks are heterogenous, with respect to their hardware and software, as a vector to understanding the architecture of autonomous systems. Our results indicate, as described in Chapter IV, that ICMP is permitted in many heterogenous networks for network management. As we describe in detail in Chapter II, Section B, there are features within the ICMP protocol that no longer have a legitimate use but are still configured on many routers. The ICMP timestamp message is an example of this. Thus, it makes sense to develop tools and procedures that will capitalize on this lenience. Based on the NIST Vulnerability Database, the ICMP timestamp request (CVE-1999-0524) has only been given a Low severity impact [5], while Cisco only labeled it as Informational in their Alarm Severity level, with "no known exploits" [6]. Additionally, Juniper Networks gave the timestamp request a 0.0 score in their Security Bulletin due its low probability of exploitation [7]. Given these reports, the likelihood of ICMP timestamp probing success is potentially high.

The router and switch market is dominated by Cisco, as shown in Figure 1. While this is good preliminary knowledge, as far as market share data is concerned, it is hard to quantify the number of actual deployed implementations without empirical testing, possibly through scanning and probing. This study hopes to build on related work on ICMP fingerprinting to:

1. Further the study on developing tools and methodologies that will differentiate between router manufacturers using ICMP.
2. Operate in concert with other tools and methodologies to contribute to the approximate fingerprinting of routers by manufacturer.
3. Provide a reference for future fingerprinting work on network devices that are configured for IPv6.
4. Contribute to network identity management and security.

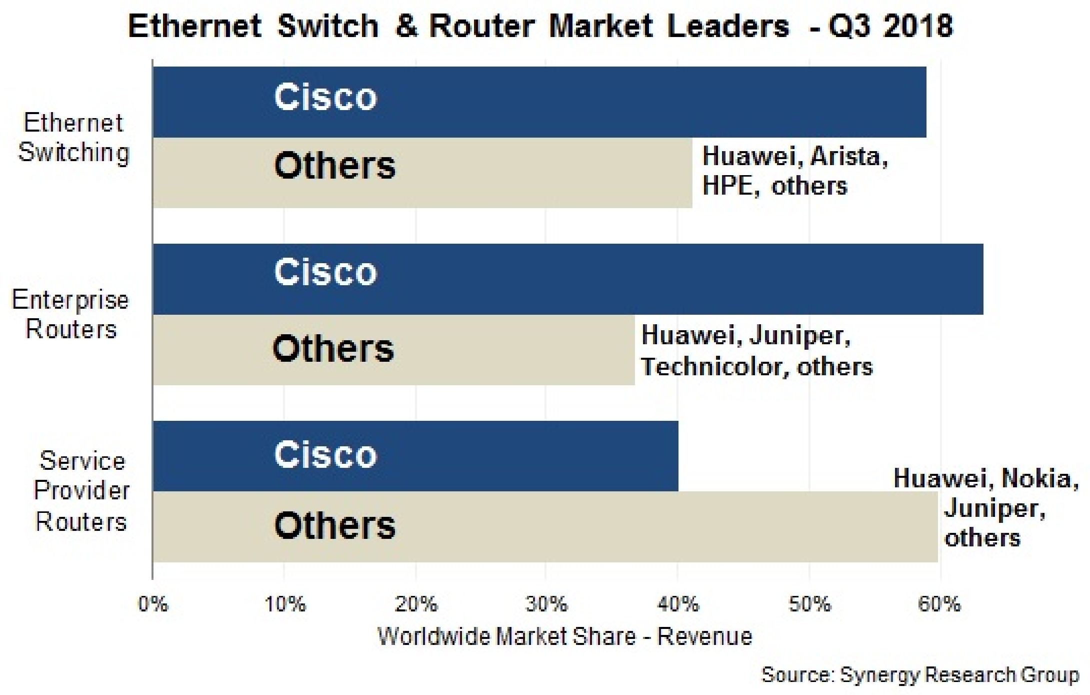


Figure 1. 2018 Networking Device Market Leaders. Source: [8].

B. RESEARCH QUESTIONS

The focus of this thesis is the use of ICMP as a mechanism to identify router platforms across public networks. We will attempt to determine which elements of the ICMP and IP protocols can be combined to produce a reliable fingerprint that most routers will respond to. In our research, we will answer these questions:

- Can we determine router platform based on an n-tuple signature?
- Can we develop a simple fingerprinting technique using ICMPv6 to identify IPv6 router platforms (e.g., Cisco, Juniper, or Huawei) based on Hop Limit values?
- Can we add additional elements to the tuple to increase fingerprint accuracy?

C. SUMMARY OF CONTRIBUTIONS

In this thesis, we develop new methods for determining router platforms by adding a new tuple to increase accuracy. This research makes the following contributions:

1. Reassessed the previous work of Vanaubel et al. [4]. Specifically, Cisco and Juniper have different TTL values when responding to ICMP type/code 0:0.
2. Added a third tuple, ICMP timestamp reply, to the IPv4 router signature.
3. Tested the ICMP router signature methodology against IPv6 routers and determined that major router platforms have aligned, using an initial hop limit value of 64 for all ICMP messages.

D. THESIS STRUCTURE

Chapter II introduces background and details of the protocols used in this study.

Chapter III discusses the methodologies used to collect data from in-house testing using point-to-point host/router configuration and live targets on the Internet.

Chapter IV provides our results based on analysis of data gathered from testing.

Chapter V details our research conclusions as well as our recommendations for future work.

II. BACKGROUND

Studies on fingerprinting network devices using ICMP, as we will discuss in detail in Section E of this chapter, are now being done to complement OS fingerprinting. Nmap [3], one of the most popular open source fingerprinting tools, is a utility for network discovery that is often used for security auditing to scan for service ports, perform OS detection and ping sweeps [9]. Xprobe2, as mentioned in Chapter I, is a tool that is designed to collect information on a remote system using ICMP in an attempt to fingerprint the OS [2]. However, there is no comparable suite of tools developed for fingerprinting networking devices.

One area of interest for fingerprinting networking devices is ICMP probing, which looks at differences in ICMP responses by type and code. RFC 791 [10] and RFC 792 [11], the internet standards for IP and ICMP, respectively, mandate compliance on certain parameters such as header length, format, fields, etc., but allow differences in how this compliance is implemented. Thus, there is a chance that differences in ICMP probing will occur between the IP header's TTL, and the ICMP header's type/code values. These differences can then be used to characterize parameters that are unique to that equipment, thereby increasing the accuracy of fingerprinting methodologies.

A. INTERNET PROTOCOL (IP)

Internet Protocol (IP), operating at layer 3 of the Open Source Interconnect (OSI) model, is the vehicle that carries datagrams, also commonly referred to as packets, across most modern networks. IP is particularly relevant to this study as it provides support for ICMP, which in turn, provides diagnostics and error reporting. It provides an addressing system that is used by packet-switched networks for proper delivery of data. Currently, there are two versions of the IP: IPv4 and IPv6. The major differences between the versions is the length of their respective address spaces, along with each having its own ICMP version.

1. IP Version 4 (IPv4)

IPv4 allows for 32 bits of address space that provides the mechanism for getting packets from the transmitter to the receiver. The IPv4 address is represented as “dotted decimal” notation, divided into four “octets” separated by periods. Each octet is 8 bits in length, as the name implies, and can hold a value from 0 to 255. Table 1 gives the summary of the contents of the IPv4 header.

Table 1. IPv4 Header. Source: [10].

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This study pays particularly close attention to the IP packet time-to-live (TTL), which is an 8-bit field in the IP header that was designed to prevent packets from endlessly bouncing around a network in the event of a routing loop. It is essentially a “self-destruct” hop limit built into the IP packet (with a hop being a forward by a network device). When a packet arrives at a router, the TTL field is checked and decremented. Should the TTL reach 0, the router discards the packet. It is important to note that only devices that perform routing functionality using the logical IP addressing can decrement the TTL; this is typically understood as “operating at layer 3 of the OSI model.”

When a packet is discarded due to the TTL reaching 0, the Internet Control Message Protocol (ICMP) takes over to notify the sender that this has occurred. Per RFC 791, the value of the TTL field "should be at least as great as the number of gateways which this datagram will traverse" [10]. This leaves router manufacturers with some leeway when setting the TTL value, which must only be a power of 2, or the maximum value of 255. Vanaubel et al. concluded that, due to nature of modern networking, most packets will reach their destination in fewer than 30 hops [4]. This would place the lowest initial value for the TTL field at 32, except when deliberately set to 1 as with the traceroute utility, described in detail in Section D of this chapter.

Consideration of initial TTL values has been leveraged by tools such as Nmap [3] and Xprobe [2] when attempting to identify the host OS (e.g., Microsoft Windows or Linux). Traditionally Microsoft used an initial TTL of 128, while Unix/Linux variants would use 64 [1]. Understanding how these initial values are set can be equally useful with routers. Cisco Systems was typically known for setting the initial TTL value to 255 (the maximum), while most Juniper Networks devices along with Linux variants used 64 [4]. This study examines packets generated by various routers to better understand their behavior, and if possible, use their behaviors to fingerprint commonly used platforms that make up the core infrastructure of the Internet.

2. IP Version 6 (IPv6)

IPv6 was developed as a replacement for IPv4 when it became apparent that the address space in IPv4 was inadequate. The address space provides for 2^n unique addresses (n being the number of bits in the address), which are necessary for distinguishing between connected nodes in a network, or internetwork. Whereas IPv4 uses a 32-bit address space, IPv6 uses 128 bits of address space, represented in hexadecimal, and divided into 8 16-bit fields separated by colons. Table 2 gives the summary of the contents of the IPv6 header.

The IPv6 header has an 8-bit Hop Limit field that is similar to the TTL field in IPv4. Like IPv4, each IPv6 packet is set with a Hop Limit by the transmitting device, which is then decremented by each device that provides routing services to the packet's destination.

If the Hop Limit reaches 0, the current router will discard the packet and generate an error message that will be sent to the transmitting device.

Table 2. IPv6 Header. Source: [12].

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

B. INTERNET CONTROL MESSAGE PROTOCOL (ICMP)

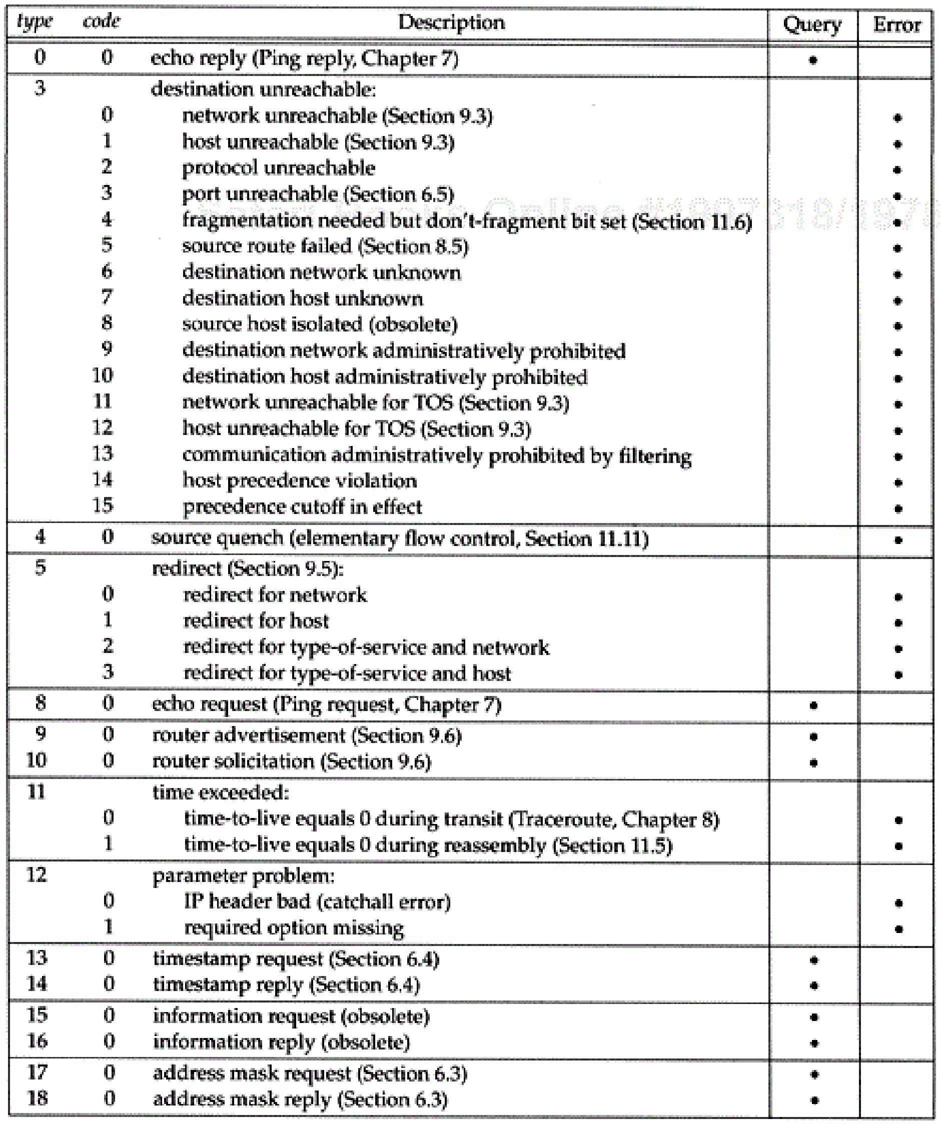
ICMP was developed in September of 1981, with the purpose of reporting errors of packets transmitted using the IP [11]. As there are two versions of the IP (IPv4 and IPv6) there are two corresponding versions of ICMP (ICMPv4 and ICMPv6). Both versions of ICMP utilize message types with associated codes to provide information about the error being reported. ICMP will also provide the IP header and a portion of the original packet that caused the error within its data field. ICMP error messages can only be generated by errors caused by an IP packet, never for errors created by other ICMP packets. Along with error reporting, ICMP has query messages which can be used to obtain information about other network hosts. There are tools that are commonly used for network administration and troubleshooting that utilize these ICMP query messages for their functionality, such as ping and traceroute. In sections C and D of this chapter, we discuss these tools and their use in this thesis.

1. ICMP Version 4 (ICMPv4)

The ICMPv4 header will always follow the IPv4 header, and consists of five fields as shown in Table 3. The fields we focus on for this study are the type and code fields, which specify the type of ICMP message, and details regarding the message type, respectively. For ICMP types that do not require more detail (e.g., type 8, echo request), the code value defaults to 0. Figure 2 shows some of the type and code values used by ICMPv4.

Table 3. ICMPv4 Header. Source: [11].

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



type	code	Description	Query	Error
0	0	echo reply (Ping reply, Chapter 7)	•	
3		destination unreachable:		
	0	network unreachable (Section 9.3)		•
	1	host unreachable (Section 9.3)		•
	2	protocol unreachable		•
	3	port unreachable (Section 6.5)		•
	4	fragmentation needed but don't-fragment bit set (Section 11.6)		•
	5	source route failed (Section 8.5)		•
	6	destination network unknown		•
	7	destination host unknown		•
	8	source host isolated (obsolete)		•
	9	destination network administratively prohibited		•
	10	destination host administratively prohibited		•
	11	network unreachable for TOS (Section 9.3)		•
	12	host unreachable for TOS (Section 9.3)		•
	13	communication administratively prohibited by filtering		•
	14	host precedence violation		•
	15	precedence cutoff in effect		•
4	0	source quench (elementary flow control, Section 11.11)		•
5		redirect (Section 9.5):		
	0	redirect for network		•
	1	redirect for host		•
	2	redirect for type-of-service and network		•
	3	redirect for type-of-service and host		•
8	0	echo request (Ping request, Chapter 7)	•	
9	0	router advertisement (Section 9.6)	•	
10	0	router solicitation (Section 9.6)	•	
11		time exceeded:		
	0	time-to-live equals 0 during transit (Traceroute, Chapter 8)		•
	1	time-to-live equals 0 during reassembly (Section 11.5)		•
12		parameter problem:		
	0	IP header bad (catchall error)		•
	1	required option missing		•
13	0	timestamp request (Section 6.4)	•	
14	0	timestamp reply (Section 6.4)	•	
15	0	information request (obsolete)	•	
16	0	information reply (obsolete)	•	
17	0	address mask request (Section 6.3)	•	
18	0	address mask reply (Section 6.3)	•	

Section references in this figure refer to sections in the original source, not this report.

Figure 2. ICMPv4 Types and Codes. Source: [13].

One point of interest within the ICMPv4 is the timestamp request and reply. A recent study showed that, while being superseded by NTP, ICMP is still not deprecated and there are a significant number of hosts on the Internet that are still responding to it. The study also classified 13 distinct behaviors that leak target host information that can be used for “fine-grained operating system fingerprinting and coarse geolocation” [14]. This summary of behaviors is listed in Table 4.

Table 4. ICMP Timestamp Fingerprints. Source: [14].

Num	Class	Checksum
1	Normal	Valid
2	Lazy	Valid
3	Checksum-Lazy	Bad
4	Stuck	Valid
5	Constant 0	Valid
6	Constant 1	Valid
7	Constant LE 1	Valid
8	Reflection	Valid
9	Non-UTC	Valid
10	Timezone	Valid
11	Little Endian	Valid
12	Linux hton () Bug	Valid
13	Unknown	Valid

The ICMP timestamp message creates a unique packet, which provides a 96-bit payload field. This payload is separated into three 32-bit timestamp fields as shown in Table 5. The Originate Timestamp field is populated by the sender with the time the message was last touched prior to being sent. The Receive Timestamp field is populated by the receiver upon first touch of the message, and the Transmit Timestamp field is also populated by the receiver upon last touch prior to the reply being sent [13]. As originally designed, the ICMP timestamp message can provide useful information, such as one direction transit time between two devices. However, as previously discussed, a recent study showed that this message leaks information (e.g., OS, kernel version, and local time zone), which has been used for fingerprinting and geolocation determination [14].

Table 5. Timestamp Message. Source: [11].

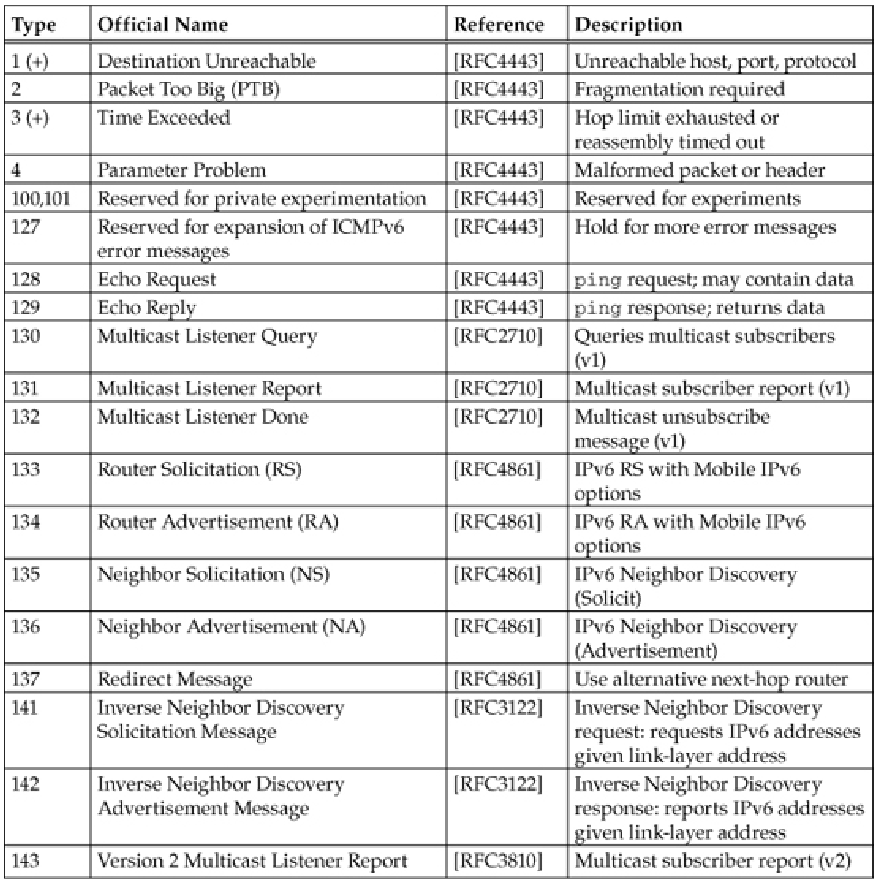
```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |      Code     |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Originate Timestamp                                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Receive Timestamp                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Transmit Timestamp                                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2. ICMP Version 6 (ICMPv6)

Like ICMPv4, an IPv6 header will always precede the ICMPv6 header. The structure of the ICMPv6 header, shown in Table 6, is nearly identical to that of ICMPv4, though it uses different values for its types and codes. Some of the functionality of ICMPv4 is not supported in version 6, such as the ICMP timestamp message. This study will focus on ICMPv6 types 1, 3, and 129 (destination unreachable, time exceeded, and echo reply, respectively). Figures 3 and 4 show some of the types and codes used by ICMPv6.

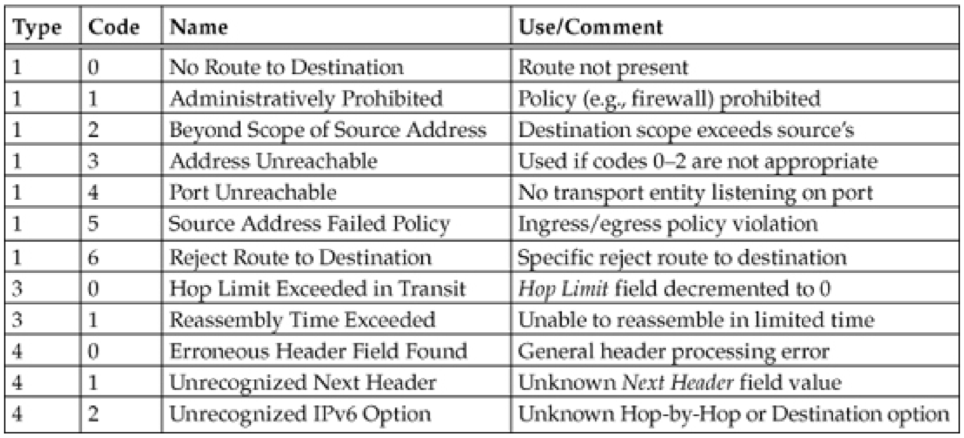
Table 6. ICMPv6 Header. Source: [15].

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|     Type      |     Code      |          Checksum             |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|                             Unused                            |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|                    As much of invoking packet                 |

+                as possible without the ICMPv6 packet          +

|               exceeding the minimum IPv6 MTU [IPv6]           |
```



Type	Official Name	Reference	Description
1 (+)	Destination Unreachable	[RFC4443]	Unreachable host, port, protocol
2	Packet Too Big (PTB)	[RFC4443]	Fragmentation required
3 (+)	Time Exceeded	[RFC4443]	Hop limit exhausted or reassembly timed out
4	Parameter Problem	[RFC4443]	Malformed packet or header
100,101	Reserved for private experimentation	[RFC4443]	Reserved for experiments
127	Reserved for expansion of ICMPv6 error messages	[RFC4443]	Hold for more error messages
128	Echo Request	[RFC4443]	`ping` request; may contain data
129	Echo Reply	[RFC4443]	`ping` response; returns data
130	Multicast Listener Query	[RFC2710]	Queries multicast subscribers (v1)
131	Multicast Listener Report	[RFC2710]	Multicast subscriber report (v1)
132	Multicast Listener Done	[RFC2710]	Multicast unsubscribe message (v1)
133	Router Solicitation (RS)	[RFC4861]	IPv6 RS with Mobile IPv6 options
134	Router Advertisement (RA)	[RFC4861]	IPv6 RA with Mobile IPv6 options
135	Neighbor Solicitation (NS)	[RFC4861]	IPv6 Neighbor Discovery (Solicit)
136	Neighbor Advertisement (NA)	[RFC4861]	IPv6 Neighbor Discovery (Advertisement)
137	Redirect Message	[RFC4861]	Use alternative next-hop router
141	Inverse Neighbor Discovery Solicitation Message	[RFC3122]	Inverse Neighbor Discovery request: requests IPv6 addresses given link-layer address
142	Inverse Neighbor Discovery Advertisement Message	[RFC3122]	Inverse Neighbor Discovery response: reports IPv6 addresses given link-layer address
143	Version 2 Multicast Listener Report	[RFC3810]	Multicast subscriber report (v2)

Figure 3. ICMPv6 Message Types. Source: [16].



Type	Code	Name	Use/Comment
1	0	No Route to Destination	Route not present
1	1	Administratively Prohibited	Policy (e.g., firewall) prohibited
1	2	Beyond Scope of Source Address	Destination scope exceeds source's
1	3	Address Unreachable	Used if codes 0–2 are not appropriate
1	4	Port Unreachable	No transport entity listening on port
1	5	Source Address Failed Policy	Ingress/egress policy violation
1	6	Reject Route to Destination	Specific reject route to destination
3	0	Hop Limit Exceeded in Transit	*Hop Limit* field decremented to 0
3	1	Reassembly Time Exceeded	Unable to reassemble in limited time
4	0	Erroneous Header Field Found	General header processing error
4	1	Unrecognized Next Header	Unknown *Next Header* field value
4	2	Unrecognized IPv6 Option	Unknown Hop-by-Hop or Destination option

Figure 4. ICMPv6 Codes. Source: [16].

C. PING

Ping, and its IPv6 counterpart Ping6, is a command line troubleshooting tool used to determine if a host can be reached, and is responsive. Both Ping and Ping6 utilize ICMP by sending an echo request message to a specified IP address. When the destined device receives the echo request packet, it will respond with an echo reply. If the destined device is unreachable, the last router in the path that processes the echo request packet will send an ICMP error message back to the sender. This message will be a destination unreachable message with the corresponding code that provides more detailed information about the error. Any networked device will respond to Ping, unless the ICMP protocol is being blocked within the network. Blocking ICMP, however, is not a common practice in most production networks due to its value for network administration and troubleshooting.

D. TRACEROUTE

traceroute is another common command line troubleshooting tool that utilizes ICMP. Its purpose is to identify the IP addresses of all layer 3 devices that a packet traverses en route to the destined IP address. To do this, the traceroute utility will create a series of packets, the first of which will have its TTL/Hop Limit field set to 1. When the packets arrive at the first layer 3 device, the TTL/Hop Limit is decremented to zero, and if that device does not own the destination IP address, an ICMP time exceeded message is returned. The source address of the ICMP TTL exceeded message provides the IP address of the router along the path reachable at the probed TTL. The traceroute utility then sends another series of packets, incrementing the TTL/Hop Limit by 1. This process of incrementing the TTL/Hop Limit by 1 will continue until the destination IP is reached, or the default timeout occurs (typically 30 hops).

There are three methods of conducting a traceroute query: ICMP, User Datagram Protocol (UDP), and Transmission Control Protocol (TCP). When ICMP is used as the query within the traceroute utility an ICMP echo request (type 8, code 0) message is sent. Each intermediate device will respond with an ICMP error message Time Exceeded, TTL/Hop Limit exceeded in transit. The target device however, will respond with echo reply. UDP is a simple protocol, at the transport layer of the OSI stack, that offers no reliability. UDP produces one datagram for every "output operation by a process," resulting in a single IP datagram being sent [13]. When the traceroute utility uses UDP probes, the destination IP address combined with a random ephemeral port is placed in the data field. When the correctly identified device receives the packet, it will not respond with Time Exceeded, TTL/Hop Limit exceeded in transit, but instead will respond with ICMP error message destination unreachable, port unreachable (type 3, code 3). This happens because the randomly selected port in the data field is not open and listening on the target device. Traceroute can also be configured to use TCP probes, but was not used for the purposes of this study.

E. IPV4 ROUTER FINGERPRINTING USING TTL-BASED SIGNATURES

The methodology developed by Vanaubel et al. [4]. relies on the differences in initial time-to-live (iTTL) values that are set for response packets, as network devices respond to various ICMP messages. iTTL values are typically set to 32, 64, 128, and 255. The researchers noted that there is no requirement for setting a specific iTTL value, only a suggestion of 64 as described in RFC 1700 [17]. This gives router manufacturers the freedom to configure these response packets with an iTTL of their choosing.

The way Vanaubel et al. [4]. were able to determine the iTTL value was by assuming the lowest TTL value greater than the value of the received response. This is based on the statistic that most IP packets will either reach their destination or be dropped in fewer than 30 hops. For example, if Vanaubel et al. [4] captured a response with a TTL of 47, it would be safe to assume the iTTL was set to 64. Vanaubel et al. [4]. analyzed initially two types of ICMP messages: time exceeded (type 11, code 0), and echo reply (type 0, code 0) [4].

By using the traceroute utility, Vanaubel et al. [4]. captured the responses from all intermediate routers in order to analyze the ICMP time exceeded message. Similarly, using the ping utility, they captured the ICMP echo reply messages from routers probed with a corresponding echo request. These differences became the fingerprint for the various router manufacturers, and in some cases, the router OSs. A two-value (two tuple) signature was then established as follows: the first value was the inferred iTTL from the time exceeded message, and the second was the iTTL from the echo reply message. They also recognized the value of incorporating additional iTTLs from different messages, such as from a destination unreachable message generated by a UDP probe, or by incorporating additional field values from the IP header. They determined that, within their data set, only about 40% of routers responded to the UDP probes. For this reason, they did not add this value to their signature.

Vanaubel et al. [4] described probing techniques that include a basic pair-signature: time-exceeded elicited by traceroute probe, and an echo-reply from echo-request probe. Using a Paris traceroute, a traceroute that addresses the load balancing deficiencies of the older traceroute version to ensure each packet in a trace follows the same path [18], they probed 1 million randomly selected destinations using 200 Planet Lab vantage points (121 in the United States, 10 in Europe, and 69 in other countries) [4]. Their data, collected to classify signature distributions, showed that Cisco platforms created the signature <255,255>, Juniper routers running Junos created <255,64>, Juniper routers running JunosE created <128,128>, and a fourth category of router that includes multiple platforms (Brocade, Alcatel, Linux) of <64,64>. Based on the Vanaubel et al. [4] study, the Cisco signature made up 50% of their results. Vanaubel et al. [4] concluded, that Cisco has the majority of networking infrastructure, is sound and that it is aligned with the current market share.

Vanaubel et al. [4] go on to test their methodology against routers configured with Multiprotocol Label Switching (MPLS), in an effort to better understand the visibility of MPLS traffic. Since there is no significant, if any, deployment of pure IPv6 MPLS networks [19], this paper will not adapt this part of the Vanaubel et al. methodology [4]. In situations where IPv6 packets come in contact with MPLS-configured routers, the traffic is routed over an IPv4 core network; this configuration is referred to as 6PE [20]. It might also be possible to extend the methodology of Vanaubel et al. [4]. by adding a third iTTL value derived from the ICMP timestamp message. While validating their methodology, we plan to see if this additional value can strengthen this fingerprinting methodology. ICMP timestamp types and codes were not carried over to ICMPv6 so the timestamp element will apply only to the IPv4 fingerprinting strategy.

III. METHODOLOGY

In Chapter II, we discussed the background information about the layer 3 protocols that are pertinent to this study (i.e., IP and ICMP). Specifically, we discussed the significant use of ICMP as a mechanism for fingerprinting hosts, and in recent studies, routers. In this chapter, we discuss methods of developing signatures for three major router platforms in a controlled laboratory environment, and describe how we conducted probing of nearly one million IP addresses associated with production routers on the internet.

A. SIGNATURE DEVELOPMENT

In Chapter II, we discussed the background information about layer 3 protocols that are pertinent to this study (i.e., IP and ICMP). Specifically, we discussed the significant use of ICMP as a mechanism to fingerprint routers. In this chapter, we will discuss the methods of developing signatures initially from a controlled laboratory environment then finally from testing in the public internet of almost a million IP addresses.

For our initial two-tuple signature, we chose the ping and traceroute utilities for creating a router fingerprinting technique, as they are universal and innocuous networking tools, as described in Chapter II, sections C and D. We chose UDP for the traceroute probe, as an ICMP traceroute would provide the same end result as a ping (echo reply) from the targeted device, in addition to the TTL exceeded messages from intermediate routers along the path to the probed destination. However, Vanaubel et al. concluded that when restricted to two-tuple signatures, “several platforms may have the same signature,” and that “an n-tuple with n > 2 would be more helpful” [4]. Therefore, we added a third tuple using the ICMP timestamp message. To create a timestamp request message, we utilized a C-language program called icmptime.c [21], developed by Stevens [13]. This program crafts an ICMP timestamp message (type 13, code 0) that, when sent to a router, results in an ICMP timestamp reply as described in Chapter II, Section B.

All of our in-lab probing is performed from an Ubuntu 18.04 VM running on a VMware Fusion hypervisor. Using Wireshark, an open source traffic analyzer software tool [22], we captured the traffic between the Ubuntu workstation and each device being probed. The first part of our testing was conducted on selected test routers configured for IPv4, namely Cisco, Juniper, and Huawei. We based our decision to test these routers mainly on the extent of their market share, as discussed in Chapter I, Figure 1. The testing parameters for IPv6-enabled routers differed in that we used the Hop Limit field (as described in Chapter II, Section A) from the IPv6 header, and ICMP timestamps were not included in the ICMPv6 protocol, thus limiting our IPv6 signature to a two-tuple. The IPv6 versions of ping and traceroute were utilized to establish the two-tuple IPv6 signature for each platform tested.

1. Cisco 7200 Series

To test the Cisco Internetwork Operating System (IOS), we used GNS3 [23], open-source networking software that allowed us to virtualize real hardware devices, to create a simple virtual network using a Cisco ISO file. By using GNS3 [23], vice a bare metal Cisco router, we could test multiple versions of Cisco IOS as required. We created a simple point-to-point configuration of a Cisco 7200 router, running IOS 15.2(4)S5, connected to a virtual Ubuntu 18.04 workstation running on VMware Fusion within the GNS3 [23] environment, as shown in Figure 5.

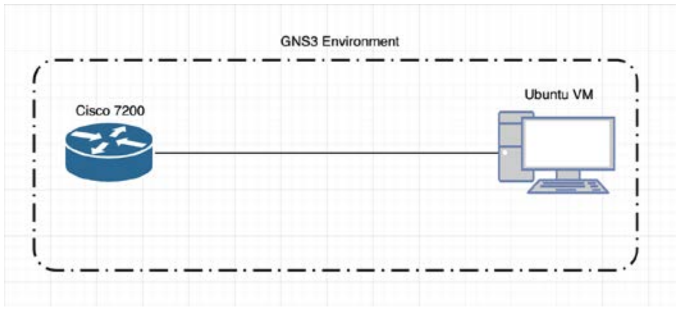


Figure 5. GNS3 Virtual Cisco Network

We configured the Cisco 7200 router interface Eth0 with IPv4 address 172.16.2.1/24, and connected it to the Ubuntu workstation network interface card (NIC) configured with IPv4 address 172.16.2.4/24. From the Ubuntu workstation, we probed the router with a traceroute query using UDP probes (the default behavior of the traceroute utility packaged with the Ubuntu 18.04 OS), and four ping (echo request) packets. The

Cisco 7200 device, when probed with a traceroute, responded with ICMP error message destination unreachable, port unreachable (type 3, code 3) with an associated iTTL of 255 as discussed in Chapter II, Section E. Following the ping probes to the Cisco 7200 device, the result was an echo reply (type 0, code 0). This was combined with the iTTL value of 255 from the IP header.

At this point we have established a 2-tuple signature for Cisco 7200 series routers of <255, 255> using the iTTLs from the response to a UDP traceroute query and ping. In contrast, the signature developed by Vanaubel et al. was also <255, 255>, but combined the iTTLs from an ICMP error message time exceeded, TTL exceed in transit (type 11, code 0) and the response from ping. Vanaubel et al. selected the iTTL from the ICMP error message time exceeded, TTL exceed in transit (type 11, code 0) for their signature due to the results of their probing of production routers. Their results showed that more than 40% of probes using UDP traceroute did not respond [4]. We take a closer look at time exceeded messages in Chapter IV, but use the iTTL from ICMP error message destination unreachable, port unreachable (type 3, code3) as our initial signature. We chose to use the destination unreachable, port unreachable (type 3, code 3) due to our proposed application of creating a fingerprinting technique. When given just an IP address, not knowing how many hops away it is, or not having the IP address of the device that lies behind it in the path can make it incredibly difficult to probe for a time exceeded message. By using the type 3, code 3 error message, specific devices can be probed and potentially fingerprinted. Based on our in-lab testing, the iTTL appears to be the same for both (type 3, code 3) and (type 11, code 0) error messages for all routers tested.

We add the third tuple to the signature by way of the ICMP timestamp message. As previously mentioned, the icmptime.c [21] program was used for this probe. The timestamp reply (type 14, code 0) from the Cisco 7200 was accompanied by an iTTL of 255. Thus, our 3-tuple signature for Cisco 7200 series routers, derived from the device responding to a UDP traceroute, ping, and a timestamp query is <255, 255, 255>.

Finally, we configured the Ubuntu workstation and the Cisco 7200 router with an IPv6 address with a /64 prefix. When probed with trace6 using UDP, the Cisco device responded with ICMPv6 error message destination unreachable, port unreachable (type 1, code 4). These test probes resulted in an IPv6 two-tuple signature of <64, 64>.When probed with an echo request (type 128, code 0), the Cisco device responded with an echo reply (type 129, code 0) accompanied by an initial Hop Limit (iHL) of 64.

2. Juniper M7i

To test a Juniper router, we chose not to use a Junos VM, which was configured as an Ubuntu OS simulating a Juniper OS, to eliminate the possibility of getting a ping or traceroute response from the Ubuntu OS vice the Junos VM guest OS. Instead we used a bare-metal Juniper M7i router running Junos Software Suite 8.2R1.7. The probing was done via a point-to-point ethernet connection to an Ubuntu 18.04 VM. From the Ubuntu workstation, we probed the M7i router with the same series used in the Cisco probing (i.e., a UDP traceroute query, four ping [echo request] packets, and an ICMP timestamp query using icmptime.c) [21].

The Juniper M7i device, when probed with a UDP traceroute query, responded with ICMP error message destination unreachable, port unreachable (type 3, code 3) with an associated iTTL value of 255. When probed with an echo request, the Juniper M7i responded with an echo reply (type 0, code 0) with an associated iTTL of 64. Thus, the initial 2-tuple signature, derived from the Juniper device responding to a UDP traceroute, and a ping (echo request) was <255, 64>. The timestamp query against the Juniper M7i router elicited a timestamp response (type 14, code 0) with an associated iTTL of 64, resulting in a 3-tuple signature of <255, 64, 64>.

Lastly, we configured the Juniper M7i router with an IPv6 address with a /64 prefix. When probed with trace6 using UDP the Juniper M7i responded with ICMPv6 error message destination unreachable, port unreachable (type 1, code 4), with an iHL of 64. When probed with an echo request (type 128, code 0), the Juniper M7i responded with an echo reply (type 129, code 0) accompanied by an iHL of 64. This resulted in a two-tuple IPv6 signature for Junos of <64, 64>. Based on this result alone, we cannot differentiate between routers running Cisco IOS and Junos in IPv6 networks.

3. Huawei

For our Huawei router testing we used eNSP, an open source Huawei network emulator test engine similar to the Cisco Packet Tracer [24]. We configured ethernet interface Eth0 on a Huawei V100R001C00 router, running software version 5.110 within eNSP [24], with IPv4 address 192.168.2.1 and the Ubuntu VM network interface with 192.168.2.2. Both devices were connected to a cloud within their respective environments, with the eNSP cloud configured to loopback address 127.0.0.1:3002, and the GNS3 cloud configured to loopback address 127.0.01:3000. We created a logical bridge on the host that connected 127.0.01:3002 to 127.0.0.1:3000. The traffic between the two systems was piped through the logical bridge using the loopback addresses, as depicted in Figure 6.

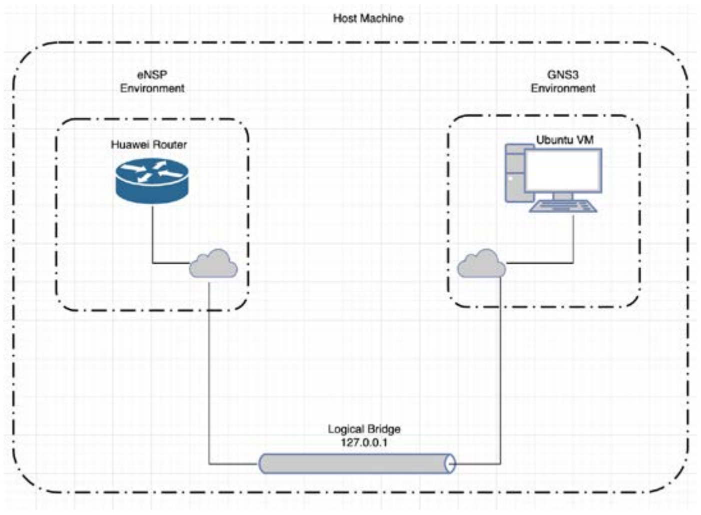


Figure 6. Huawei Virtual Network

As with the Cisco and Juniper tests, we probed the Huawei device with a UDP traceroute query, a series of four ping (echo request) packets, followed by an ICMP timestamp query. The Huawei device responded to the UDP traceroute with an ICMP error message destination unreachable, port unreachable (type 3, code 3) with an associated iTTL of 255. It responded to the echo request with an echo reply (type 0, code 0) with an associated iTTL of 255.These probes resulted in an initial two-tuple signature of <255, 255> matching that of the Cisco 7200 series router. When probed with the ICMP timestamp query, the Huawei device responded with an ICMP timestamp reply (type 14, code 0) and an associated iTTL of 255. The resulting 3-tuple signature for Huawei is <255, 255, 255>, which is identical to that of the Cisco 7200 series router.

As in the case of the other two routers, we configured the Huawei device with an IPv6 address with a /64 prefix. When probed with a trace6 using UDP, the Huawei device responded with ICMPv6 error message destination unreachable, port unreachable (type 1, code 4). When probed with an echo request, the Huawei device responded with an echo reply (type 129, code 0) with an iHL of 64. This resulted in an IPv6 two tuple signature of <64, 64>. The IPv6 in-lab testing resulted in all tested router platforms (Cisco, Juniper, and Huawei) having the same two-tuple signature, as shown in Table 1. This test result is undesirable for fingerprinting because there is no differentiating factor between router platforms. Based on this result, we recommend that researchers use an n-tuple signature, where n > 2, for better results. We discuss additional ICMP message types that can be explored in Chapter V.

Table 7. iTTL/iHL Values

	Traceroute	**Ping**	**Timestamp**	**Ping6**	**Trace6**
Cisco IOS iTTL	255	255	255	64	64
JunOS iTTL	255	64	64	64	64
Huawei iTTL	255	255	255	64	64

The recommended default TTL for IP, as prescribed in RFC 1700, is 64 [17]. It would appear that when implementing IPv6 the major router platform manufacturers chose to align with this recommendation. Thus far, we have determined that the probing types tested are not adequate to differentiate the platforms, with the exception being the ability to identify Junos configured with IPv4.

B. LIVE TARGETS IN PUBLIC INTERNET

The server used to conduct our live router probing was an Ubuntu 16.04 VM, hosted on a well-connected university server in Massachusetts, USA, and accessed using a remote secure shell (SSH) connection. We obtained a list of 973,000 IPv4 addresses and a list of 187,000 IPv6 addresses from CAIDA's Archipelago [25] active topology measurements, specifically the IPv4 routed /24 topology dataset [26,27] and the IPv6 topology dataset [28,29], both from August 18, 2019. To obtain router interface addresses, we discovered the set of unique addresses responding with ICMP time exceeded messages. We ignored any other responses, and responses from the traceroute target destination. We did not perform any alias resolution, so there may be multiple interfaces in our dataset that belong to the same physical router.

Prior to initiating the probing of the dataset, we launched tcpdump on interface Eth0 to save the activity to a packet capture file for later analysis. Then we configured tcpdump to write to a separate packet capture file for each probe conducted. For the ping and traceroute probing, we utilized Scamper, a tool developed to actively probe the Internet in order to analyze topology and performance [30]. This tool supports standard ping and traceroute techniques, as well as Paris traceroute, described in Chapter II, Section D.

We started the probe by sending a four-packet series of pings to our IPv4 dataset at a rate of 500 packets per second. Then we ran two instances of Scamper [30], the first to initiate the ping utility and the second to initiate the traceroute utility. Finally, we ran the traceroute probe with the default configuration, UDP probes and Paris traceroute as described in Chapter II, Section E.

To accomplish the timestamp probing, we utilized zmap [31] with the incorporated sundial module [32]. The sundial module has the ability to send four different types of timestamp query packets:

1. Standard, which has the origin time set correctly.

2. Bad Clock, which has the origin time set incorrectly.

3. Bad Checksum, a timestamp request with a bad checksum.

4. Duplicate Timestamp, a timestamp request with all three time fields populated with the origin time.

For this study, we probed our IPv4 dataset with each of the four types of timestamp request packets. The responses from each type of timestamp query were captured, by tcpdump, into a single packet capture file.

Using |Scamper, we ran ping and traceroute probing against the IPv6 dataset. As stated in Chapter II, timestamps are not supported by IPv6, so zmap [13] and sundial [14] were not used against this dataset. Similarly, results of these probes are contained in separate packet capture files, and their analysis is covered in detail in Chapter IV.

IV. RESULTS

We discussed in Chapter III the methods used to probe three different router manufacturers using ICMP in our lab-based testing, followed by the probing of nearly one million production IP addresses across the Internet. We developed a 3-tuple signature consisting of iTTL values associated with ICMP message type/code 3:3 (destination unreachable, port unreachable), type/code 0:0 (echo reply), and type/code 14:0 (timestamp reply) for fingerprinting three major router platforms. The goal was to determine if the signatures would help to interpret the iTTL distribution that resulted from the probing of production routers. We added the third probe, the ICMP timestamp reply, to the tuple in the hope of increasing the accuracy of fingerprinting. This chapter will focus on the analysis of the results collected from both series of tests. We have devoted a significant part of this study to reassessing previous work on fingerprinting TTL-based routers, which is discussed in this chapter. We also expand the methodology to routers configured with IPv6 addressing, and analyze the IPv6 iHL distribution resulting from the production router probing. To the best of our knowledge, our work is the first to analyze IPv6 iHL distribution.

A. ANALYSIS OF IPV4

To analyze the data within each packet capture file, we used a custom python script which imported dpkt [33], a packet capture analysis tool that can parse a large variety of protocols and extract the data from each protocol of a capture file. For this study, the information we are interested in is the TTL value and source address from the IPv4 header, and the ICMP type/code values from the ICMP header. We configured our custom python script to pull just the TTL associated to packets containing each of the ICMP type/code values described in Chapter III, as well as type/code 11:0 (time exceeded, TTL exceeded in transit) from the traceroute capture file. We also configured the python script to deduplicate response packets based on the source IP address. The script incremented elements in an array corresponding to possible TTL values 0 to 255. The script was then configured to group the elements by range to produce a count of assumed iTTLs. We followed the same methodology that Vanaubel et al. [4] used to determine the initial TTL value: elements 0-32 were assigned iTTL of 32, 33-64 were assigned iTTL of 64, 65-128 were assigned iTTL of 128, and elements >128 were assigned iTTL of 255. The output of the script was written to a .csv file used to produce the figures in this chapter.

B. COMPARISON TO PREVIOUS WORK

For our initial goal of validating the work of Vanaubel et al. [4], we pulled the same data from our production router probing as that used by Vanaubel et al. [4] to create the cumulative distribution frequency (CDF) displayed in their paper. This data included type/code 0:0 (echo reply), type/code 3:3 (destination unreachable, port unreachable), and type/code 11:0 (time exceeded, TTL exceeded in transit) that resulted from probing approximately one million routers as shown in Figure 7.

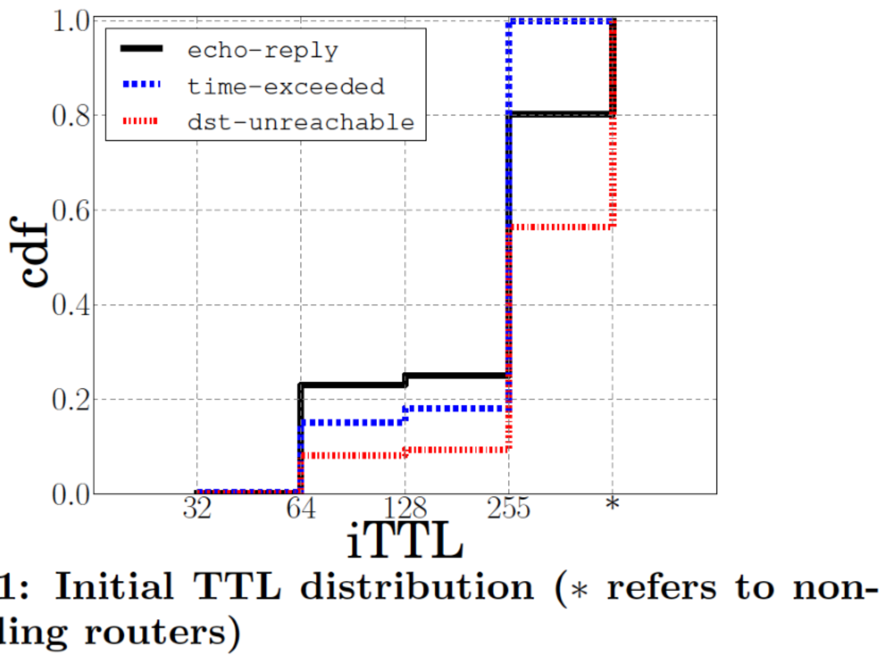


Figure 7. Vanaubel et al. CDF. Source [4]

The CDF that Vanaubel et al. [4] generated from their results showed that with echo reply messages, nearly 60% had an associated iTTL of 255, just over 20% had an associated iTTL of 64, and the remaining 20% received no response. The Vanaubel et al. results showed that with destination unreachable messages, the expected response from a UDP traceroute, just over 40% had an associated iTTL of 255, just over 40% received no response, and less than 10% had an associated iTTL of 64. The Vanaubel et al. results then showed that for the time exceeded messages, the expected response from intermediate hops during a traceroute query, just over 80% had an associated iTTL of 255, and just under 20% had an associated iTTL of 64. Responses with iTTLs of 32 and 128 were negligible with all response types.

The results of our production router probing are shown in Figure 8. We are using a probability distribution function (PDF) because each column represents a range of values that have an assumed starting point. The X axis displays each assumed starting point (iTTL values), and the Y axis displays the percentage of occurrence (where 1 represents 100%). As with the Vanaubel et al. [4] CDF, the asterisk symbolizes no response received. Regarding the echo reply messages, 40% had an associated iTTL of 255, 30% had an associated iTTL of 64, and 30% received no response. For the destination unreachable messages, about 35% had an associated iTTL of 255, 10% had an associated iTTL of 64, and more than 50% received no response. Finally, the time exceeded messages resulted in 80% with an associated iTTL of 255, and 20% with an associated iTTL of 64.

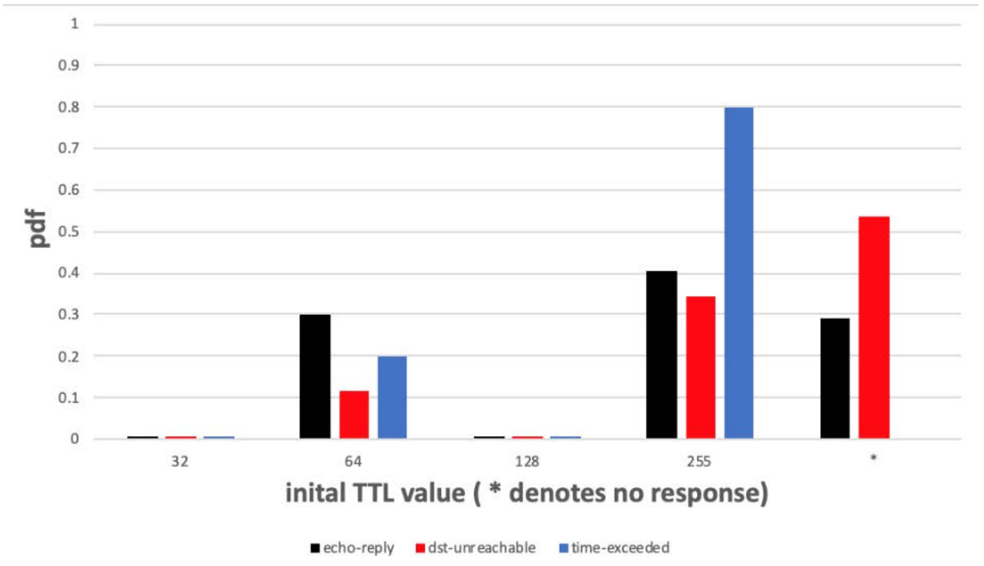


Figure 8. iTTL Distribution from Ping and Traceroute Probes

Based on this data, over the past six years we see two significant changes in echo reply messages, the first being a significant decrease in iTTLs of 255, and the second being an increase in those receiving no response. We see a similar trend with destination unreachable messages with an iTTL of 255. Interestingly, echo reply messages with an iTTL of 255 resulted in a lower percentage in our study than that of Vanaubel et al., where both the iTTL of 64 and no response increased. With destination unreachable messages we saw iTTL of 255 drop slightly, and the no response increase slightly, but the distribution stayed more consistent, as shown in Table 8.

Table 8. Comparison Chart of iTTL Values

Vanaubel et al.	Echo-reply	Dst-unreachable	Time-exceeded
iTTL 255	~53%	~46%	80%
iTTL 128	~2%	~1%	~2%
iTTL 64	~24%	~8%	~16%
iTTL 32	~1%	~1%	~1%
No response(*)	~20%	~44%	~1%
Our results	**Echo-reply**	**Dst-unreachable**	**Time-exceeded**
iTTL 255	~41%	~34%	~79
iTTL 128	~1%	~1%	~1%
iTTL 64	~30%	~11%	~19%
iTTL 32	~1%	~1%	~1%
No response(*)	~27%	~53%	-

C. ADDITION OF THE TIMESTAMP REPLY

As discussed in Chapter III, to generate the timestamp tuple for the signature we used icmptime.c [21] during the lab testing. For the production router probe we used zmap [31] with the sundial module [32]. The lab testing results, displayed in Table 7, show that the iTTL values for the timestamp reply messages are identical to those of the iTTL values of the echo request packets for the three platforms tested. As we described in Chapter III, the timestamp probe included all four types of timestamp message (standard, bad clock, bad checksum and duplicate timestamp). When the timestamp probe packet capture data was added to the distribution, roughly 50% of the routers did not respond to any of the timestamp probes, 30% had an associated iTTL of 255, and 20% had an associated iTTL of 64. It is interesting to note that our Internet probing also shows that the timestamp responses follow the same trend as echo reply and destination unreachable, with an iTTL of 255 being at least 10% higher than that of 64, as shown in Figure 9. Also, after adding the timestamp reply as the third tuple, the distribution is still roughly the same.

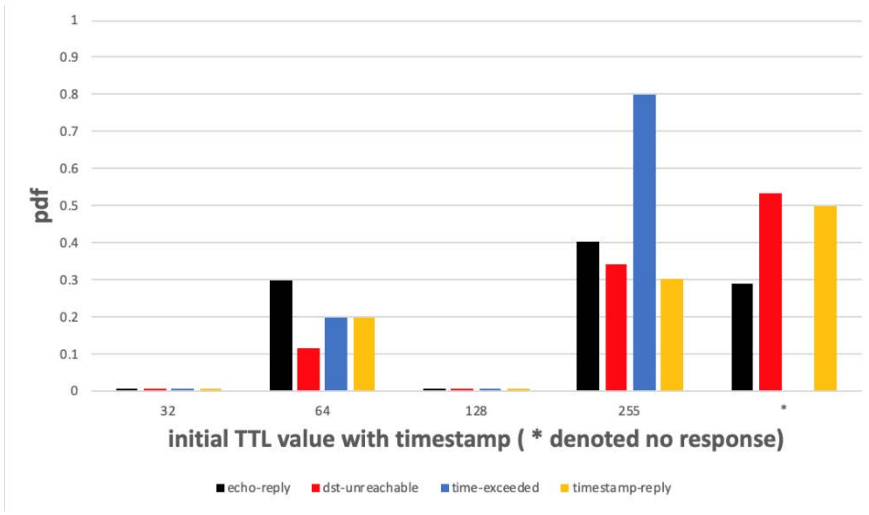


Figure 9. IPv4 iTTL Distribution for ICMP Timestamp Reply

We discussed in Chapter II that we created our initial 2-tuple signatures using ICMP error message destination unreachable, port unreachable (type 3, code 3), and echo reply (type 0, code 0) instead of using the time exceeded, TTL exceeded in transit (type 11, code 0) as was done by Vanaubel et al [4]. Our production router probing of nearly one million IPv4 addresses resulted in 452,218 destination unreachable, port unreachable responses, in contrast with 219,588 time exceeded, TTL exceeded in transit responses from unique source IP addresses. Of the 452,218 destination unreachable packets, only 5,323 did not have an associated iTTL of 64 or 255. Because our three platforms all align with the iTTLs of 64 and 255, based on our in-lab testing, we analyze the main signature distribution for possible combinations of these two values shown in Figure 10.

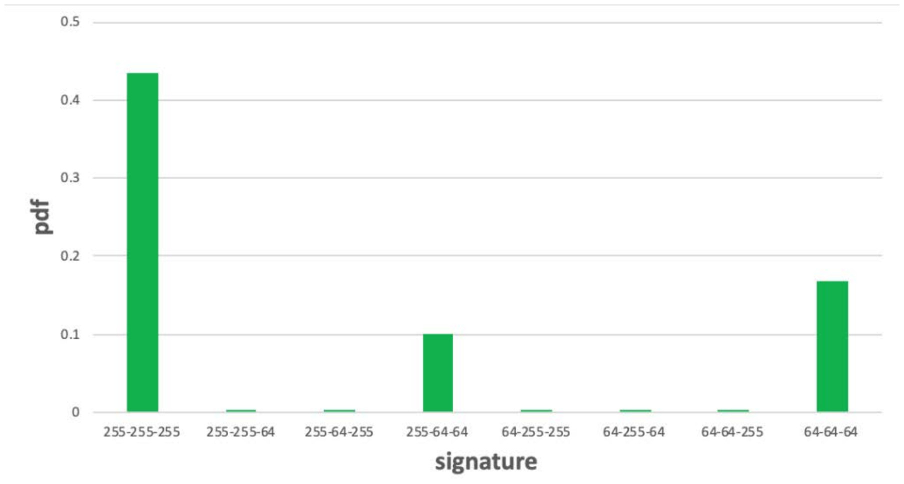


Figure 10. 3-Tuple Signature Distribution

The 3-tuple signature <255,255,255> makes up nearly 45% of IPv4 addresses that responded to all three probes. This signature also matches that of the Cisco and Huawei signatures we developed in Chapter II, Section A (refer to Table 7). The 3-tuple signature <255,64,64> makes up 10% of IPv4 addresses that responded to all three probes. This signature matches that of the Junos signature we developed in Chapter II, Section A. The signature <64,64,64> makes up nearly 20% of the IPv4 addresses that responded to all three probes. This signature did not correlate to any of the three platforms tested in this study, however Vanaubel et al. determined that the 2-tuple signature <64,64> comprised of Brocade, Alcatel, and other Linux variants [4]. We can speculate that our 3-tuple signature <64,64,64> represents this group as well. Further study will be necessary to determine this.

D. IPV6 IHL DISTRIBUTION

Parsing the capture file for the IPv6 testing was done in the same manner as with the IPv4 analysis. We altered the python script to pull the HL field from the IPv6 header, and the type/code fields from the ICMPv6 header. For IPv6 we are interested in the type/code values of 129:0 (echo reply), 1:4 (destination unreachable, port unreachable), and 3:0 (time exceeded, HL exceeded in transit). As was shown in Table 7, there was no difference in HL distribution for the platforms tested in the lab. Analysis of the IPv6 probing, shown in Figure 11, shows that the iHL value of 64 dominates as that of iHL of 255, which is consistent with what was observed in lab testing.

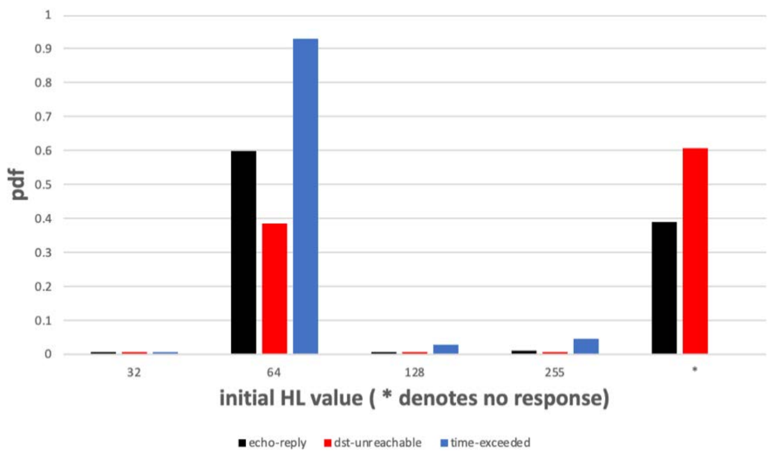


Figure 11. IPv6 iHL Distribution for Ping and Traceroute

This result is a stark contrast from the IPv4 probing results, which had a significant split between iTTLs 64 and 255. Also, the time exceeded responses having an iHL of 64 increased to more than 90% of the total, while all other iHL value counts are marginal or non-existent. Future study could determine the kind of devices are producing these values. Additionally, the non-responses increased approximately 10% for each message type from the results obtained for IPv4. This contrast may be attributed to the fact that router manufacturers are now more in-line with the implementation of RFC 1700 [17] in IPv6 than they were with IPv4. This may be due to router manufacturers, like Cisco Systems, already having their default TTL already set at the time the recommendation (RFC 1700) was released. Our hypothesis is that non-iHL 64 responses may be coming from devices that are not routers but servers with Windows OS configured to perform routing functions for small or remote autonomous systems. This could explain the iHL of 128 as Microsoft has traditionally used 128 as the default TTL [1].

E. SUMMARY

In this chapter, we discovered that we were able to successfully expand the previously developed IPv4 router fingerprint, developed by Vanaubel et al. [4], using the ICMP timestamp reply message. Using this fingerprinting methodology, Juniper routers can be identified. However, this fingerprinting technique cannot distinguish between Cisco and Huawei routers. With IPv6, it became evident that most routing devices follow the recommend iHL value of 64, as prescribed in RFC 1700 [17]. Thus, our methodology cannot distinguish between IPv6 routing devices. Our recommendation to conduct additional analysis of Cisco and Huawei devices running IPv4 to identify differences in activity, as well as further research into IPv6 routers is detailed in Chapter V.

V. CONCLUSIONS AND FUTURE WORK

In this study, we reassessed the previous work of Vanaubel et al. [4], and expanded the simple fingerprinting method for IPv4 routers by adding a third tuple. Specifically, we extended the fingerprinting technique by combining the iTTL values associated with ICMP messages of types: destination unreachable, port unreachable (type 3, code 3), echo reply (type 0, code 0), and timestamp reply (type 14, code 0). We then adapted the methodology to test IPv6-enabled routers, instead using the initial Hop Limit value associated with the following ICMPv6 message types: destination unreachable, port unreachable (type 1, code 4), and echo reply (type 129, code 0). Our adaptation for IPv6 is limited to a 2-tuple signature because the ICMP timestamp message is not included in ICMPv6, per RFC 4443 [15].

Within the scope of our in-house lab and Internet testing, we were able to distinguish a probing response difference between Cisco/Huawei and Juniper using our third tuple, timestamp. Comparing our results with those of Vanaubel et al. showed that Juniper routers have a distinct signature of <255,64,64>, which sets them apart from the rest of router manufacturers. However, we were unable to determine a difference between Cisco and Huawei, as their responses are the same for all values of the tuple (i.e., <255,255,255>). We also noticed a third category of routers, with <64,64,64> signatures. They were not within the scope of our study, but we suspect belong to the group of routers including Brocade, Alcatel, and other Linux based routers as determined in the Vanaubel et al. study. The details of this group can be explored in future work. From our IPv6 probing, we were unable to distinguish our target router platforms from one another due to identical results of their responses to the ICMPv6 messages used. It may be necessary to find an $n > 2$ signature, or a different probing technique, to fingerprint IPv6 configured routers.

It is interesting to point out that, since the publication of the study by Vanaubel et al. on fingerprinting routers using TTL-based signatures, our testing results suggest that, for leading router manufacturers using the IPv4 protocol, no major configuration changes have occurred, as far as the RFC 1700 recommended initial TTL value of 64 is concerned

[17]. This observation leads us to believe that future work is still needed on router fingerprinting using information elicited through leakages in ICMP reply messages. Additionally, based on our testing results, it appears that router manufacturers are now more aligned with their implementations of RFC 1700 [17] in IPv6. Thus, further exploration of fingerprinting techniques for these devices is necessary.

For routers configured with IPv4, our probing of production routers resulted in some responses with iTTL values of 32 and 128. Further exploration can determine what these devices are, which may answer why their responses differ from the majority of routers. Further exploration will also be necessary to identify a method of distinguishing Cisco devices from Huawei. Based on recent market trends [8], we speculate that the large majority of routers that match the <255,255,255> signature are likely Cisco devices. However, as Huawei continues to grow it may become necessary to develop a way to identify and distinguish them in the wild.

Considering our inability to define an initial fingerprinting methodology for routers configured with the IPv6 protocol, it will be necessary to explore additional methods to make this possible. Within the ICMPv6 protocol there are other error messages that can be explored:

- Packet Too Big—Since fragmentation is only performed at the source of the packet and not by routers along the path of the packet, as detailed in RFC 2460 [12], it would be interesting to find out if router platforms respond identically or differently when an oversized packet is received.

- Parameter Problem—RFC 2460 states that, “If an IPv6 node processing a packet finds a problem with a field in the IPv6 header or extension headers such that it cannot complete processing the packet, it MUST discard the packet and SHOULD originate an ICMPv6 Parameter Problem message to the packet's source, indicating the type and location of the problem” [12]. If router manufacturers adhere to this RFC’s recommendation on Parameter Problem responses, the ‘type’ and ‘location’ of the problem could be a valuable source of information.

Our testing of IPv6-configured routers shows that there are some routers producing iHL values of 128 and 255 when sending an ICMPv6 message. We recommend that future work be performed to identify the type of these devices.

Further study can also be conducted to integrate this methodology with other active fingerprinting techniques, such as Nmap [3]. Regarding signature distribution, additional analyses may be able to determine how groups of particular signatures are dispersed throughout the network. For instance, future work could determine whether the signature distribution within China will look the same as the signature distribution within the United States, or whether network routers in certain geographic locations are configured to not respond to ICMP. It could also investigate whether there are applications that can benefit from adding a router platform as an input.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] O. Arkin, “ICMP usage in scanning—the complete know how version 3.0,” unpublished.

[2] O. Arkin and F. Yarochkin, “A ‘fuzzy’ approach to remote active operating system fingerprinting,” unpublished.

[3] G. Lyon, Palo Alto, CA. 1997. Nmap, ver. 7.80. [ONLINE]. Available: https://nmap.org

[4] Y. Vanaubel, J. Pansiot, P. Merindol, and B. Donnet, “Network fingerprinting: TTL-based router signatures,” presented at ACM SIGCOMM, Hong Kong, China, August 12, 2013.

[5] National Institute of Standards and Technology, “CVE-1999-0524,” July 10, 2017. [ONLINE]. Available: https://nvd.nist.gov/vuln/detail/CVE-1999-0524#vulnCurrentDescriptionTitle

[6] Cisco, “ICMP timestamp request,” August 31, 2011. [ONLINE]. Available: https://tools.cisco.com/security/center/viewIpsSignature.x?signatureId=2007&signatureSubId=0

[7] Juniper Networks, “2015-10 Security bulletin: CTPView: Multiple vulnerabilities in CTPView,” October 14, 2015. [ONLINE]. Available: https://kb.juniper.net/InfoCenter/index?page=content&id=JSA10705

[8] Synergy Research Group, “Ethernet switch & router market at an all-time high; Cisco increases its market share,” December 5, 2018. [ONLINE]. Available: https://www.srgresearch.com/articles/ethernet-switch-router-market-nudges-all-time-high-cisco-increases-its-market-share

[9] C. Jackson, *Network Security Auditing*. Indianapolis, IN: CISCO Press, 2010.

[10] *Internet Protocol*, RFC 791, 1981. [ONLINE]. Available: http://www.ietf.org/rfc/rfc791.txt

[11] *Internet Control Message Protocol*, RFC 792, 1981. [ONLINE]. Available: http://www.ietf.org/rfc/rfc792.txt

[12] *Internet Protocol version 6 (IPv6)*, RFC 2460, 1998. [ONLINE]. Available: https://www.ietf.org/rfc/rfc2460.txt

[13] W. R. Stevens, *TCP/IP Illustrated: The Protocols*. Reading, MA: Addison-Wesley Publishing, 1994.

[14] E. Rye and R. Beverly, “Sundials in the shade, An internet-wide perspective on ICMP timestamps,” presented at PAM, Puerto Varas, Chile, March 28, 2019.

[15] *Internet Control Message Protocol (ICMPv6)*, RFC 4443, 2006. [ONLINE]. Available: https://tools.ietf.org/html/rfc4443

[16] K.R. Fall and W. R. Stevens, *TCP/IP Illustrated.* -2nd ed. Ann Arbor, MI: Edwards Brothers, 2012.

[17] *Assigned Numbers*, RFC 1700, 1994. [ONLINE]. Available: https://tools.ietf.org/html/rfc1700

[18] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, “Avoiding traceroute anomalies with Paris traceroute,” presented at ACM SIGCOMM, Pisa, Italy, September 11, 2006.

[19] *Gap Analysis for Operating IPv6 Only MPLS Networks*, RFC 7439, 2015. [ONLINE]. Available: https://tools.ietf.org/html/rfc7439

[20] *Connecting IPv6 Islands over IPv4 MPLS using IPv6 Provider Edge Routers (6PE)*, RFC 4798, 2007. [ONLINE]. Available: https://tools.ietf.org/html/rfc4798

[21] W. R. Stevens, Reading, MA, USA. 1994. Icmptime.c. [ONLINE]. Available: https://www.unf.edu/public/cda6506/rbutler/stevens.software/author.written/tcpipi/icmptime/

[22] G. Combs.1998. Wireshark. [ONLINE]. Available: https://www.wireshark.org/about.html#authors

[23] D. Bombal, D. Ziajka, J. Grossmann, and J. Duponchelle. 2019. GNS3. [Online]. Available: https://gns3.com/software

[24] Huawei Technologies. 2019. eNSP. [ONLINE]. Available: https://support.huawei.com/enterprise/en/network-management/ensp-pid-9017384/software

[25] Center for Applied Internet Data Analysis, “Archipelago (ARK) measurement infrastructure,” June 7, 2019. [ONLINE]. Available: http://www.caida.org/projects/ark/

[26] Center for Applied Internet Data Analysis, “The IPv4 routed /24 topology dataset,” March 27, 2019. [ONLINE]. Available: http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml

[27] The CAIDA UCSD IPv4 Routed /24 Topology Dataset - August 19-30, 2019, www.impactcybertrust.org, doi: 10.23721/107/1354084

[28] Center for Applied Internet Data Analysis, “The IPv6 topology dataset,” March 27, 2019. [ONLINE]. Available: http://www.caida.org/data/active/ipv6_allpref_topology_dataset.xml

[29] The CAIDA UCSD IPv6 Topology Dataset - August 19-30, 2019, http://www.caida.org/data/active/ipv6_allpref_topology_dataset.xml.

[30] M. Luckie, “Scamper: a scalable and extensible packet prober for active measurement of the Internet,” presented at ACM SIGCOMM, New Delhi, India, August 30, 2010.

[31] Z. Durumeric, E. Wustrow, J.A. Halderman, *Zmap: Fast internet-wide scanning and its security applications*. In: USENIX Security. pp. 605–620 (2013)

[32] E. C. Rye, *Sundial ICMP Timestamp Inference Tool* (2019), https://www.cmand.org/sundial

[33] The Python Package Index. 2019. Dpkt, ver. 1.9.2. [ONLINE]. Available: https://pypi.org/project/dpkt/

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