Covert Channel Using ICMPv6 and IPv6 Addressing

Geoffrey Ackerman
Department of Computing Security
Rochester Institute of Technology
gma8175@rit.edu

Daryl Johnson
Department of Computing Security
Rochester Institute of Technology
daryl.johnson@rit.edu

Bill Stackpole
Department of Computing Security
Rochester Institute of Technology
Bill.Stackpole@rit.edu

Abstract— *Internet Protocol version 6, the latest revision of the Internet Protocol (IP), is rising in popularity. Along with it has come ample opportunity for the discovery and utilization of fresh, new covert channels. This paper proposes a covert channel using this "IP Next Generation Protocol", widely referred to as IPv6, as well as its associated protocol ICMPv6. As a proof-of-concept, two hosts running respective sender and receiver python scripts will take advantage of ICMPv6 Echo messages and the IPv6 addressing scheme to send and receive data unbeknownst to any host, person, or other entity that may be monitoring or watching over the network.*

Keywords: IP, IPv6, ICMPv6, Covert Channel, Sniffing, Spoofing

1. Introduction

A covert channel is a communication channel that transfers information in ways prohibited by computer security policy and unspecified by the respective protocol. This can be accomplished through the use of the covert channel's structure to transfer small amounts of data at a time. While encryption denies a traffic observer knowledge of the contents of a conversation, the goal of covert channels is to deny knowledge of the conversation itself. Without knowledge of the operation of a covert channel, observation of the traffic would not reveal the channels existence or contents. Even if the message is seen, there is no way of knowing that the datagram is something out of the ordinary. It is an example of security through obscurity. This paper will discuss some IP-based covert channels that have been discovered and will propose, demonstrate, and evaluate a new channel using the IPv6 and ICMPv6 protocols.

1.1 Covert Channel Types

Covert channels are generally categorized as storage or timing channels. A storage channel "involves the direct or indirect writing of a storage location by one process and the direct or indirect reading of the storage location by another process" [1]. A timing channel involves transfers of information based on a chosen timing interval whose messages must be synchronized with a similar clock on each end. Additionally, some advocate for a third category called behavioral channels [2]. These channels use an alteration of internal states or behavior of an application to leak information.

1.2 Related Covert Channels

Joe Klein, a network expert with the North American IPv6 Task Force, said,

> "We are expecting a lot here to be discovered and disclosed. But just like the early implementation of any technology, we expect to find defects and covert channels." [3]

Many IPv6-based and IPv4-based covert channels have already been found. One such example of an IP-based covert channel is known as the Loki Project [4]. This channel uses the ICMPv4 protocol. Covert data is sent and received through Echo-Request and Echo-Reply packets. More specifically, the tool uses the data field of these echo messages, a field of both arbitrary length and content normally used for timing information, to hide the data. Since this field is both optional and not useful to most devices, it is not normally checked.

Another example is a tool called VoodooNet, or v00d00n3t [5], which is one of the most commonly referenced covert channels using IPv6. Created by R.P. Murphy and presented at Defcon 14 in 2006, this tool uses a technique known as 6to4 tunnelling by encapsulating IPv6 network traffic within today's standard, IPv4. Through optional extension headers in IPv6 packets, messages can be transferred from one host to another. This methodology of 6to4 tunneling was not revolutionary, as it had been demonstrated and used for years, but the use of it as a way to covertly send information was.

IPv6 implements "extension headers", which are used to carry optional Internet Layer information. No intermediary nodes read these headers as they are only meant for the destination node. These headers can be abused to create covert channels, as described in [6]. One example uses unusual options in the extension header so that when the destination node reads this option, it will skip the extension header and move on to the next one, thus allowing covert data to be stored in the data section of the skipped header. Another example uses the PadN option, which is used to align packet boundaries by inserting two or more octets of padding into a header's Options. Per the IPv6 RFC, this padding should consist of 0's. However, certain operating systems will accept non-zero padding, thereby allowing

arbitrary covert data.

An extensive study of IPv6-based covert channels can be found in the dissertation "Network-Aware Active Wardens in IPv6" by Grzegorz Lewandowski of Syracuse University [7]. In this paper, Lewandowski theorizes many fields in many different protocols that use IPv6 that may be used as a covert channel. He suggests the idea of setting a false source IPv6 address multiple times but does not explore the idea any further. To understand how this can be used as a covert channel, one must have an understanding of the protocols.

2. Protocol Overviews

2.1 History of IPv6

Due to the unprecedented expansion of Internet usage and the ever-increasing number of new devices being connected to the Internet, the impending shortage of IPv4 address space available for use was recognized. In response, the Internet Engineering Task Force (IETF) initiated the design and development of new protocols and standards to eventually supplant Internet Protocol version 4. The result, the "IP Next Generation Protocol", was developed with larger 128-bit addresses compared to the 32-bit addresses used by IPv4. This allows for an astoundingly larger address space, approximately $3.4x10^{38}$ addresses, as compared to the approximately $4.3x10^{9}$ addresses available in IPv4. The basic protocol was published in 1998 [8] and as of September 2013 the percentage of users using IPv6 reaching Google services surpassed two percent [9]. In 2006 the associated Internet Control Message Protocol (ICMPv6) specification [10] was published to serve as a critical component of IPv6.

2.2 ICMPv6

ICMPv6 uses Echo-Request and Echo-Reply messages in the same way as ICMPv4. Within these message packets there are six distinctive fields (*Table 1*). This covert channel will use the Identifier field, which is 16-bits in length, and the optional Data field of arbitrary length. The Identifier's purpose is to match corresponding Echo-Request and Echo-Reply messages with each other. This gives two hosts exchanging echo messages confirmation that they are talking with the intended target. The actual number used does not have an effect on the outcome of the communication. The data field consists of zero or more octets of arbitrary ASCII data, generally used for timing information (i.e. computing round trip time).

Table 1: ICMPv6 Packet

Type	Code = 0	Checksum
Identifier	Sequence Number	Optional Data

2.2.1 Neighbor Discovery Protocol

In version 6 of IP and ICMP, ARP no longer exists. As a replacement, there is the Neighbor Discovery Protocol [11] operating in the Link Layer of the Internet model [12]. Hosts and routers use this new protocol for address autoconfiguration, determining the link-layer addresses of neighboring hosts, to keep track of which neighbors can be reached, and to find routers that are available to forward their packets. The protocol uses Router and Neighbor Solicitations and Advertisements. Any host can issue a Router Solicitation to find any routers attached on a link. These packets are sent out periodically. A router can also advertise its presence on a link using Router Advertisements. Routers will advertise themselves periodically or in response to a specific solicitation. Similarly, any node can determine link-layer addresses of any neighbors using a Neighbor Solicitation. These are also used to determine if a known host is still reachable using a cached link-layer address. Hosts use Neighbor Advertisements to respond to Neighbor Solicitations. All of these packets play an integral role in the functioning of the ICMPv6 protocol, and therefore the IPv6 protocol.

2.3 IPv6

The IPv6 addressing scheme (*Table 2*) commonly consists of a 48-bit ISP-assigned "Site Prefix" field, a 16-bit "Subnet" field, and a 64-bit "Interface Identifier". An example address is: 21DA:D3:1:2F3B:2AA:FF:FE28:9C5A/64 (leading zeroes and groups of one or more consecutive zeroes can be omitted):

Each address consists of eight groups of four hexadecimal numbers. Each of these groups is 16 bits, or two octets. An address can be assigned through various means, including from the network interface's MAC address, from a DHCPv6 server, or through manual configuration. Every IPv6-enabled interface must have a link-local address, which has the prefix fe80::/64. The site prefix can be either link-local, unique local (equivalent to IPv4 private addresses), or global (equivalent to IPv4 Internet addresses). Global IPv6 addresses are globally routable and can be used to connect to addresses with a global scope anywhere, or addresses with link-local scope on the directly attached network. Global Unicast addresses have an IPv6 prefix of 2000::/3 [13].

Table 2: IPv6 Address

Site Prefix	Subnet	Interface ID
21DA:00D3:0001:	2F3B:	02AA:00FF:FE28:9C5A

3. New Covert Channel Process

The proposed covert channel is demonstrated as a proof-of-concept two-sided python script, which utilizes Scapy [14]. Scapy is a packet manipulation tool that can create and send custom packets containing arbitrary data. The python

script consists of a sender and a receiver on two different hosts, both run with root privileges.

The Identifier field of an ICMPv6 Echo-Request packet, which can be manually configured by the sender, and the Data field, which consists of optional arbitrary data of arbitrary length, will be modified to initiate a connection and to establish a dynamic alphabet by which messages can be encoded and decoded. The use of a dynamic alphabet is purely for the sake of complexity. If this channel were to be discovered, the randomization and changing of alphabets would help to mitigate any frequency analysis done on the source IPv6 addresses. Once the encoding step is complete, the sender will use a statically configured IPv6 source address to represent, transmit, and deliver a desired message via raw UDP packets. A few assumptions must be made for the purpose of this proof-of-concept: 1) both the sender and receiver have access to root privileges, 2) IPv6 is enabled on all devices that the covert channel traverses, 3) sender and receiver real IPv6 addresses must be a shared secret of both parties, and 4) UDP is not blocked.

The proposed covert channel uses the ICMPv6 protocol much in the same way as the Loki Project. However, it is unique in that the Echo-Request messages are solely used to establish the alphabet(s) that will be used to encode and decode the subsequent message. The message is created and transferred when this process is taken one step further using an alternative method. UDP datagrams with spoofed, unique source IPv6 addresses are used to transfer the message with the source addresses representing the actual message fragments. Every two octets of the interface identifier represents one character of a message that has been created using the alphabet defined in each Echo-Request packet. An example using three unique source IPv6 addresses sending "Hello World" is illustrated in *Fig. 1*.

IPv6 Address: 2000::4ca2:8df7:1589:1589

Translation:	4ca2	8df7	1589	1589
	H	e	l	l

IPv6 Address: 2000::1617:9742:1617:a49e

Translation:	1617	9742	1617	a49e
	o	W	o	r

IPv6 Address: 2000::7950:b701:0000:0000

Translation:	7950	b701	0000	0000
	l	d		

Fig. 1: Hello World Example

Note: the use of simple hex-based encoding/decoding could potentially be replaced with a more complex encryption mechanism. The use of encryption with this channel would help to normalize character frequencies and eliminate the effectiveness of frequency analyses. That being said, this paper's focus is on making use of the spoofed IPv6 source addresses of UDP packets, not the variety of techniques that could be utilized to establish a way to obscure or obfuscate the intended message.

3.1 The Process

3.1.1 Initial Alphabet Definition

Since the encoded message will be an IPv6 address, each plaintext alphanumeric character must have an associated string of four hexadecimal values (the alphabet), as illustrated in *Fig. 2*.

A	5120	G	907d	M	640f	S	5d74	Y	9129
a	32bf	g	7e30	m	62da	s	4b6c	y	860e
B	bf05	H	4ca2	N	3f45	T	f43e	Z	1a89
b	36b7	h	ae28	n	e4a3	t	3f52	z	c0df
C	5c3e	I	1589	O	dd17	U	67dc	1	e285
c	51cf	i	3afd	o	1617	u	dd29	2	625d
D	429d	J	5653	P	bfec	V	ef2a	3	5519
d	b701	j	ffc9	p	6a70	v	88b0	4	abc5
E	e2f7	K	3096	Q	4b07	W	9742	5	5a67
e	8df7	k	56f9	q	9df4	w	d27a	6	1ed4
F	f68a	L	612b	R	c56c	X	7378	7	a841
f	ea53	l	7950	r	a49e	x	482d	8	9edb
								9	a075

Fig. 2: Example Alphabet

The receiving host, known as Bob, starts his receiving script, which begins by passively sniffing the network for ICMPv6 messages with his IPv6 address as the destination address. The sending host, known as Alice, reads in the message to send and chooses the number of UDP packets to send per alphabet. Alice then calculates the number of unique alphabets that are needed and creates them using a random 4-digit hex-string generator.

In *Fig. 3* Alice is seen inputting her message and deciding on the number of packets to encode and send per each unique alphabet. Then three variables, two of which are sent with the initial alphabet, are displayed. Once the appropriate number of alphabets are created and stored, the message needs to be encoded. The encoding occurs before anything is sent to Bob. Based on the chosen number of packets per alphabet, each alphabet is used to encode a certain number of characters in the original message. The encoding is accomplished character by character - each plaintext character is compared with each letter in the alphabet array and if they match, the associated hex string is used as that letter's encoding and as one octet of the IPv6 address to be spoofed.

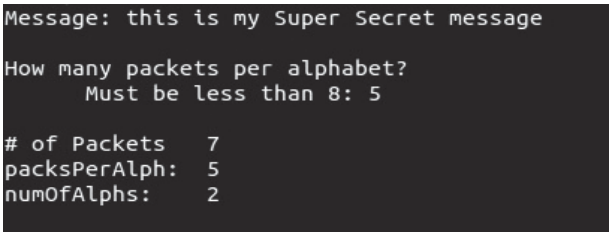

Fig. 3: Alphabet Creation

Now Alice is ready to initiate the message sending process. To start, an ICMPv6 Echo-Request is built with an arbitrary ID value (e.g. 0x62, or 98 in decimal) and the first alphabet is inserted into the Data field starting at the 98th position (Note: any follow-up alphabet Echo-Requests will have an ID value of 0x63. These values are arbitrary but they must be an agreed upon, non-random value). All data in positions 0-97 are randomly generated values. The Echo-Request message is then sent to the recipient's IPv6 address.

When Bob reads through all of the packets that have been captured by the filter he set, he looks for an Echo-Request destined for his address, with an ID Value of 0x62, and data in an alphabet format at position 98. The following is the alphabet as it is stored in the Data field of the Echo-Request, shown in *Fig. 4*.

A2e8c**a**927f**B**7c6d**b**6874**C**db94**c**8a2b**D**dfcd**d**d0b6**E**c42c
e9029**F**29a**ff**6d2**G**1958**g**a7d8**H**fbf9**h**e10a**I**c418**i**3890
J93ef**j**f048**K**598d**k**3095**L**af13**l**2ba9**M**cb13**m**d50c**N**9d03
nf0be**O**995c**o**1eb1**P**ff74**p**205e**Q**a9e6**q**1782**R**6b73**r**f4c2
Sc9d4**s**9cd4**T**b50b**t**24d8**U**7bc9**u**3a54**V**421c**v**fdfb**W**72c9
w10a1**X**e2eb**x**59d0**Y**f236**y**4ba0**Z**5206**z**67921fda424d10
31bd8**4**395a**5**9ac0**6**4f207f193**8**41799f842**1715**

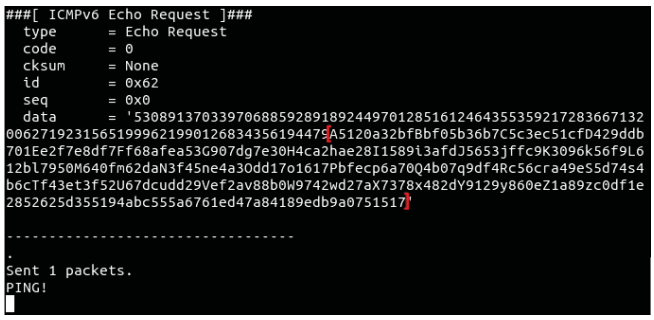

Fig. 4: Alphabet w/in ICMPv6 Echo-Request (Alice)

When this is found, the alphabet is stored and an Echo-Reply is crafted as an ACK to be sent back to Alice. Immediately after, Bob begins sniffing for UDP packets destined for him *(Fig. 5)*.

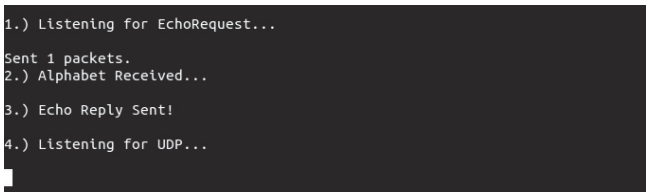

Fig. 5: Listening for UDP Message Fragments (Bob)

3.1.2 Message Sending

After sending the initial alphabet Echo-Request, Alice begins sniffing the network, awaiting the Echo-Reply acknowledging retrieval of the alphabet. Once the ACK is received, five UDP packets with Global prefixes are created from the first alphabet and sent to Bob. Four characters from the message are used per packet (i.e. the four groups of two octets used as the interface identifier in the IPv6 source address). Each address is created from the original message, spoofed as the source address, and sent to Bob via UDP *(Fig. 6)*.

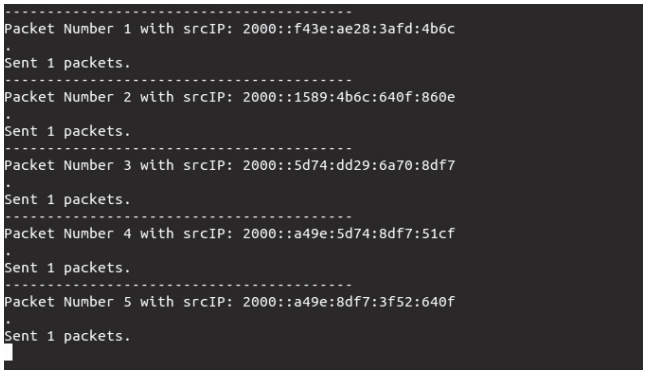

Fig. 6: Message Fragments Sent! (Alice)

3.1.3 Message Retrieval

Bob reads each source IPv6 address in each UDP packet addressed to his IPv6 address and parses out the interface identifier. These encoded message fragments are stored until the entire message is received. After all UDP packets have been inspected, Bob sends an Echo-Request with a message in the Data field signaling that the message fragments have been received and he is ready for the next alphabet to be defined *(Fig. 7)*.

```
1.) Listening for EchoRequest...

Sent 1 packets.
2.) Alphabet Received...

3.) Echo Reply Sent!

4.) Listening for UDP...


Sent 1 packets.

5.) Echo Request Sent!

    - Ready for next Alphabet
```

Fig. 7: Messages Received - Ready for next Alphabet (Bob)

3.1.4 Repeat Until Complete

From this point onward, assuming there is more message data to send, the cycle of (Alice) defining an alphabet, (Bob) ACK, (Alice) sending message fragments, (Bob) reading and storing IPv6 interface identifiers, and (Bob) requesting a new alphabet is repeated. Once the end of the message is sent and Bob requests a new alphabet, Alice sends an Echo-Request with an ID Value of 0x64, signaling the end of the message. Once the end of the message is signalled, Bob can decode the message using each alphabet he has stored for the chosen number of message fragments, as specified in the initial alphabet definition *(Fig. 8)*.

```
************************************************
********************DECODING********************
************************************************

Encoded Message:
f43eae283afd4b6c15894b6c640f860e5d74dd296a708df7a49e5d748df751cfa49e8df
73f52640f8df74b6c4b6c32bf7e308df7

Decoded Message:
This Is My Super Secret Message

************************************************
************************************************
************************************************
```

Fig. 8: Received Encoded and Decoded Messages (Bob)

4. Covert Channel Characteristics

To evaluate this covert channel, Eric Brown's criterion as described in [15] will be used. Using the following characteristics and metrics this channel's implementation is evaluated.

4.1 Type

This covert channel is considered a storage channel. While the described proof-of-concept's implementation depends on the sender and receiver agreeing on a set period of time to sniff the network while waiting for the necessary packets to come through, the actual data being sent is stored within a field of an IPv6-based UDP packet (i.e. the Source Address field). The timing itself does not contribute to the message being sent.

4.2 Throughput

The throughput of this covert channel was tested using a 2,000 character message, which required 500 UDP packets. The average time it took for these 500 packets to be sent using variable amounts of virtual RAM is shown in *Fig. 9*. The results show that the amount of vRAM allocated per VM is generally proportional to the channel's throughput. The tests conducted were performed on two Linux virtual machines running Ubuntu 14.04 in VMware Workstation. The host machine was a Lenovo U430p laptop with the following specifications:

Windows 8.1 (64-bit)
Intel Core i3-4010u @ 1.70 GHz
4.00 GB RAM w/ 99.73 MHz bus speed

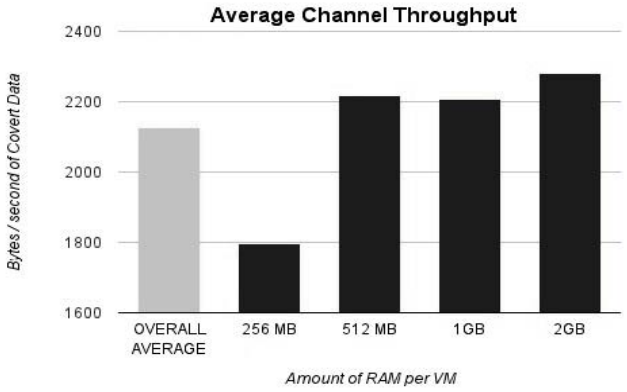

Fig. 9: Channel Throughput

The overall average time it took for each full message transmission was 3.799 seconds. Dividing the total number of characters in the message by the time it took to send equates to 531.2 characters per second. Since one character is represented by four bytes of data, the overall average bandwidth of this covert channel is 2124.938 bytes/second.

The calculated throughput of this channel may vary by the chosen time interval used by each party and the number of alphabets used. There is an inverse relationship between this time interval and the overall bandwidth of the channel. Additionally, an attacker would likely limit the successive transmission of packets in order to avoid detection due to floods of packets. Spreading out the transmissions over longer intervals of time and limiting the amount of UDP messages sent at once reduces the throughput while decreasing probability of detection. Finally, the overall bandwidth of the channel will be limited by 1) The third party application (Scapy) that is used to send and receive the packets as it will have a limit to how quickly it can do so, and 2) The natural latency caused by the infrastructure that the packets must traverse to and from the source and destination.

4.3 Robustness

The authors consider the survivability of this channel to be high. An extremely important aspect of this covert channel is its routability. Because the IPv6 address varies only in the interface identifier, the prefix can be link-local, unique local, or global, whichever the user decides would work best in the given situation. Encountering firewalls or proxy servers may cause problems, but due to the need for both IPv6 and ICMPv6 protocols in modern networks, these perimeter devices will likely allow all of this traffic in and out. With the increased number of Internet connected devices using IPv6, network managers will not want to implement a block of this protocol for fear of restricting their clients and other IPv6 users who are legitimately on the network. Additionally, if the user is aware of the unique local prefix of the site he/she is a part of, they would be able to circumvent a firewall restricting non-site specific addresses simply by using that unique local prefix instead of a global prefix for each spoofed source address. Moreover, sending the data over standard ports, such as port 80, 443, or 8080, would only decrease the likelihood of the data being blocked.

4.4 Detection

Little traffic generated by this channel looks out of place or malicious. The one packet-field that appears different from normal is the data field of the Echo-Requests that are sent to and from each host. However, this field is rarely inspected by any devices since its intended purpose is timing information. Using the standard Ping networking utility, this data field is padded with arbitrary data (e.g. on Windows, in both ICMPv4 and ICMPv6, the alphabet is used). However, there is no standard for this payload data and it can be 0 or more bytes in length. It would be possible for an intrusion detection system, such as Snort, an open source network IDS/IPS, to use manual rules that will check for anomalous payloads in particular packets. If the payload of the Echo-Request packets were to be checked for abnormal length or content, an alarm may be triggered. The default payloads for common operating systems could additionally be learned and used to detect any abnormal ICMPv6 packets. That being said, using standard default rule sets, Snort failed to detect anomalous payloads in [16] in multiple tests. On top of that, there is nothing abnormal about the IP addresses that are spoofed.

If this covert channel were implemented using a site-specific prefix, a site with a large amount of IPv6 traffic would need some sort of constantly-updating white list to keep track of every legitimate IPv6 address on the network. If properly implemented, anti-spoofing and anti-alien firewall rules could be used to detect site-specific addresses coming in from the Internet. If implemented using the global prefix, as demonstrated in the proof-of-concept, it may be possible for network border watchers to recognize non-local source addresses. However, being the equivalent of public IPv4 addresses, these global addresses are not likely to be seen as abnormal. Furthermore, the likelihood of IPv6 source addresses to be analyzed statistically and for abnormalities is low. Overall, the authors consider the probability of detection to be low-to-medium.

4.5 Prevention

One quick and easy way to prevent this channel's implementation would be disabling ICMPv6, eliminating the receiver's ability to decode the message that he receives. However, due to the importance and vitality of the ICMPv6 protocol, it cannot be blocked. If IPv6 is enabled, ICMPv6 must also be enabled (and vice-versa). Therefore, a user can enable IPv6 on his or her host machine (assuming it is not already enabled by default), establishing an open channel to anyone on the outside. Another way to prevent this attack is to totally block all IPv6 traffic from coming in or going out. While this may have been an applicable solution five years ago, prior to the depletion of IPv4 address space, it is no longer viable. Businesses must be aware of and ready and able to manage the use of IPv6 on their network. They must have the protocol enabled on all devices and network managers must monitor, analyze, and inspect this traffic thoroughly. It is common best practice to block ICMP that is initiated from an external host. This defense could prevent a dynamic alphabet from being possible. Completely disabling UDP would prevent the main covert channel from existing entirely.

Intrusion detection/prevention systems pose a threat to this covert channel. Snort can trigger alerts based on signature, protocol, and anomaly-based inspection. ICMPv6 payloads can be inspected for length and if they are longer than a specified value, an alert is triggered. With the alphabet in the Echo-Request payloads, this type of rule could possibly detect this aspect of the covert channel. However, the aforementioned tests show that Snort IDS fails to detect abnormal ICMP traffic, such as large packet sizes, using the standard rule sets that come with the software. So unless specific rules are applied to detect proper versus improper length of a payload or variations in how certain operating systems build these packets, even the most widely deployed IDS/IPS solution worldwide will not detect or prevent the dynamic alphabet aspect of this channel.

Due to the covert channel's mechanism, little will trigger an alarm at the network level. Therefore, on Brown's rating scale of hard, moderate, and easy this channel is considered hard to prevent.

5. Future Work

Further investigation into the IPv6 protocol should be performed to find additional storage fields that are variable and not normally considered worthy of analysis. These fields may become the basis for even more IPv6 covert channels. Automated timing of the channel has not been tested with

success; the users of the proof-of-concept must interact with the scripts by issuing a KeyboardInterrupt when needed (i.e. at each timing interval's end). Testing of Snort rules, other IDS/IPS rules, and firewall rules against the channel's mechanism need to be performed as well for more precise data on robustness, detectability, and preventability.

6. Conclusion

IPv6-based attacks are by no means uncommon and they will only become more and more prevalent as time goes on and IPv6 becomes more widely accepted and implemented. If network managers do not take IPv6 into account when they are building and monitoring their network infrastructure, they are likely to be a target of an IPv6-based attack. Whether a corporation sees IPv6 as a business driver or not, the protocol is probably running on their network. A network with IPv6-enabled routers, firewalls, and IDS/IPS can have rogue IPv6 traffic coming in and out of the network without being scrutinized. In this case, the new covert channel has free reign to run between any number of devices on the network. As demonstrated by this covert channel proof-of-concept, an open IPv6 channel presents a serious vulnerability to a company's network and can result in a dangerous exploitation. The covertness of this channel is such that with an IPv6 implementation, or lack thereof, there is a very serious risk of successful attack using these standard and essential "Next Generation" protocols.

References

[1] "Covert Storage Channel." ATIS Telecom Glossary. N.p., n.d. Web. 1 Dec. 2013.

[2] Johnson, Daryl, Peter Lutz, and Bo Yuan. "Behavior-Based Covert Channel in Cyberspace." Proc. of The 4th International Conference on Intelligent Systems & Knowledge Engineering, Belgium, Hasselt. N.p.: n.p., 2009. Print.

[3] Lemos, Robert. "Covert Channel Tool Hides Data in IPv6." www.securityfocus.com. N.p., 11 Aug. 2006. Web. 09 Oct. 2013.

[4] Loki Project Daemon9 AKA Route. "Project Loki." www.phrack.com. Phrack Magazine, Aug. 1996. Web. 12 Nov. 2013.

[5] Murphy, R. P. "IPv6 / ICMPv6 Covert Channels." Las Vegas: Defcon, Aug. 2006. PDF.

[6] Mavani, Monali, and Leena Ragha. "Covert Channel in IPv6 Destination Option Extension Header." Circuits, Systems, Communication and Information Technology Applications. International Conference. 2014. (CSCITA 2014). Proc. of 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), India, Mumbai. Vol. 1. N.p.: Institute of Electrical and Electronics Engineers (IEEE), 2014. 219-24. Print.

[7] Lewandowski, Grzegorz. "Network-Aware Active Wardens in IPv6." Diss. Syracuse U, 2011. Print.

[8] Deering, S., and R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification."Http://www.ietf.org/. N.p., Mar. 2006. Web. 30 Sept. 2013.

[9] Roberts, Phil. "Internet Society." http://www.internetsociety.org/. N.p., 24 Sept. 2013. Web. 02 Nov. 2013.

[10] Conta, Et Al. "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification." sww.ietf.org. N.p., Mar. 2006. Web. 30 Sept. 2013.

[11] Narten, T., E. Nordmark, W. Simpson, and H. Soliman. "Neighbor Discovery for IP Version 6 (IPv6)." IETF Tools. Internet Engineering Task Force, Sept. 2007. Web. 12 Nov. 2013.

[12] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[13] Hinden, R., OâĂŹDell, M., and S. Deering, "An IPv6 Aggregatable Global Unicast Address Format", RFC 2374, July 1998.

[14] "Scapy." Secdev.org, n.d. Web. 30 Sept. 2013. <http://www.secdev.org/projects/scapy/>.

[15] Brown, Eric, Bo Yuan, Daryl Johnson, and Peter Lutz. "Covert Channels in the HTTP Network Protocol: Channel Characterization and Detecting Man-in-the-Middle Attacks." N.p., n.d. Web. 22 Sept. 2013.

[16] Stokes, Kristian, Bo Yuan, Daryl Johnson, and Peter Lutz. "ICMP Covert Channel Resiliency." N.p.: n.p., 2009. PDF.