May 24th, 3:00 PM

Covert6: A Tool to Corroborate the Existence of IPv6 Covert Channels

Raymond A. Hansen
Department of Computer and Information Technology, Purdue University, hansenr@purdue.edu

Lourdes Gino
Department of Computer and Information Technology, Purdue University, lginod@purdue.edu

Dominic Savio
Department of Computer and Information Technology, Purdue University

COVERT6: A TOOL TO CORROBORATE THE EXISTENCE OF IPV6 COVERT CHANNELS

Raymond A. Hansen, Lourdes Gino, Dominic Savio
Department of Computer and Information Technology
Purdue University West Lafayette, IN 47907
[hansenr, lginod]@purdue.edu

ABSTRACT

Covert channels are any communication channel that can be exploited to transfer information in a manner that violates the system's security policy. Research in the field has shown that, like many communication channels, IPv4 and the TCP/IP protocol suite have been susceptible to covert channels, which could be exploited to leak data or be used for anonymous communications. With the introduction of IPv6, researchers are acutely aware that many vulnerabilities of IPv4 have been remediated in IPv6. However, a proof of concept covert channel system was demonstrated in 2006. A decade later, IPv6 and its related protocols have undergone major changes, which has introduced a need to reevaluate the current state of covert channels within IPv6. The current research demonstrates the corroboration of covert channels in IPv6 by building a tool that establishes a covert channel against a simulated enterprise network. This is further validated against multiple channel criteria.

1. INTRODUCTION

Covert channels and various other research on the subject have been in existence for more than four decades. As early as 1973, [22] explained a method of exploiting the communication channels that are "not intended for information transfer at all, such as the service program's effect on the system load" and termed it "Covert Channels" as opposed to the "legitimate channels" of communication. In 1986, the Department of Defense not only gave a formal definition for covert channels, but also classified them into "storage channels" and "timing channels" based on their methods of operation [23]. While a covert storage channel involves "the direct or indirect writing of a storage location by one process and the direct or indirect reading of it by another," a covert timing channel involves signaling mechanisms that modifies the response time, which conveys the data [25]. Many research studies showed that the weaknesses in the TCP/IP design and implementation could be exploited, which paved a way for new covert channels, mainly in IPv4. Although, the beginning of IPv4 address exhaustion in February of 2011 [27], is demanding the organizations to migrate to IPv6. This has led to various research studies to proactively find covert channels in IPv6 so as to provide better network security. The current research attempts to create a tool to prove the capabilities of establishing such covert channels in IPv6 so that it could be studied in detail.

2. REVIEW OF RELEVANT LITERATURE

In 1997, [30] showed that IPv4 (mainly TCP encapsulated in IPv4) has vulnerabilities, which were exploited in order to create a covert channel, which was called "covert TCP program," and allowed for the leaking vital data through a network. The IP identification field, TCP initial sequence number field, and TCP acknowledgment number field were used to accomplish the exploit. It was explained how these posed a major security threat and how these could be exploited "in the areas of data smuggling and anonymous communication." Also noted in this work was that the detection of such channels could be difficult, mainly if they were encrypted or if "the packets were bounced off an external server" making them seem legitimate.

[2] furthered the research by [30] and presented his thesis on covert channel analysis and data hiding in IPv4. He not only provided a compendium of prior research in the field, but also explained various exploits that allowed the creation of covert channels. He explained that covert channel based on packet header manipulation were not restricted to only a TCP header, but also IGMP and ICMP headers by the use of various encoding mechanisms. He also discussed another method of creation of covert channels, "data hiding through packet sorting." This method worked based on various algorithms that could be used to sort and resort packets. These packets could then be encrypted using the IPSec architecture, thereby providing confidentiality through the network and avoiding detection of the packet header modifications by powerful Firewalls or IDS. Interestingly, he did not discuss the negative implications of covert channels such as data leakage, rather presented it as a means of improving the network security. Finally, he noted covert channels on IPv6 as future work, as early as 2002. Unfortunately, there is no published work on this topic.

[24] reviewed various research studies on covert channels at the time and showed the existence of numerous other exploits than the usual packet header manipulations discussed by the previous researchers, that allowed the creation of covert channels. They included, "Covert Messaging through TCP Timestamps," "IP Checksum Covert Channels and Selected Hash Collision," "Malicious ICMP Tunneling," and "Exploitation of data streams authorized by a network access control system for arbitrary data transfers: tunneling and covert channels over the HTTP protocol" They also described various implementations of covert channels such as, "Data Hiding in TCP/IP with HTTP Reverse Proxy Servers," and "IP Covert Timing Channels." Like, [2], [24] provided a glimpse of IPv6 covert channel research at that time. They also showed how [15] exploited the IPv6 Destination option to create a covert messaging tool. Even though IPv6 deployments were uncommon even as late as 2008 [10], the research on covert channels in IPv6 existed as early as 2003.

One of the two important research studies that are the basis for the current research is [25], while the other one being a proof of concept by [26]. To be precise, [25] did a "specification-based analysis, to identify redundancies and ambiguities in the protocol semantics that could potentially be used to carry covert data," and provided a comprehensive list of 22 potential covert channels in IPv6. Although, none are known to have been exploited as of this writing, some of their findings included:

1. The number of research studies on network storage covert channels was more than the number of research studies on network timing covert

channels owing to the synchronization issues and lower band- width of the latter. In their opinion, this potentially eliminates the need for further study on network timing covert channels

2. The most effective defense mechanisms of the time against IPv4 covert channels were protocol scrubbers, traffic normalizers and active wardens. They did not explain if these defense mechanisms hold good against IPv6 covert channels, which paves a way for a study on how these mechanism fare against IPv6 covert channels
3. There were at least six IPv6 covert channels, created by exploiting six fields in the IPv6 header such as, Traffic Class, Flow Label, Pay- load Length, Next Header (adding various extension headers to it), Hop Limit and Source Address
4. Other potential IPv6 covert channels could be created by exploiting the extension headers such as the Hop-by-Hop Options Header, Routing Header, Fragment Header, Destination Options Header, Authentication Header, Encapsulation Security Payload Header
5. More covert channels could be created by tunneling traffic such as IPv6 in IPv4, IPv6 in IPv6 and IPv4 in IPv6.

They finally noted covert channels in ICMPv6 as their future work, but in the same year, [26] showcased this to be possible.

In 2006, [26] demonstrated a proof of concept tool called "V00d00n3t," which exploited the ICMPv6 echo-reply payload to create a covert data channel. He showed that he was able to transfer data over the Internet in a network infrastructure which was designed using a tunnel broker and a set of routers and servers, as explained in the topology in [26]. Although the design validated that the packets would survive in a 'slick' 6 network and wild uncontrolled environments, he noted that the limitation was the survival of the covert data in a production environment with firewalls, IPS/IDS etc. This motivated the introduction of Firewalls and IDS in the current study.

3. NEED FOR REEVALUATION

Although [25] discussed the various covert channels in IPv6 and [26] showed a proof of concept of the IPv6 covert channels, major changes have been introduced in IPv6 since 2006, making the research studies potentially obsolete and introducing the need to understand the current scenario. Both [25] and [26] worked on the basis of the three IPv6-related RFCs that existed in 2006, RFC 2460 [16], RFC 3697 [28], and RFC 4443 [11].

Since 2006, there have been no fewer than 12 RFC drafted to update IPv6 and ICMPv6 operations. Some of these RFC were drafted in an attempt to close potential security issues, while others were drafted to improve operations of those protocols. While some of these RFCs successfully closed the issues specified within, others were not so successful. In some cases, when one issue was resolved, another was inadvertently created with respect to the specifics of establishing and maintaining covert channels. Some of those relevant RFCs (along with additional readings on the impacts and potentials for exploitation) are:

1. RFC 4294 "IPv6 Node Requirements", April 2006 [17]
2. RFC 4727 "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", November 2006 [13]
3. RFC 4884 "Extended ICMP to Support Multi-Part Messages", April 2007 [7]

4. RFC 5095 "Deprecation of Type 0 Routing Headers in IPv6", December 2007 [1]. [19]
5. RFC 5722 "Handling of Overlapping IPv6 Fragments", December 2009 [20]
6. RFC5871"IANAAllocationGuidelinesfor theIPv6RoutingHeader", May 2010 [3]
7. RFC 6437 "IPv6 Flow Label Specification", November 2011: This RFC obsoleted RFC 3697 used by [26] [29]
8. RFC 6564 "A Uniform Format for IPv6 Extension Headers", April 2012 [21]
9. RFC 6935 "IPv6 and UDP Checksums for Tunneled Packets", April 2013 [9]
10. RFC 6946 "Processing of IPv6 Atomic Fragments", May 2013 [14]
11. RFC 7045 "Transmission and Processing of IPv6 Extension Headers", December 2013 [18]
12. RFC 7112 "Implications of Over-sized IPv6 Header Chains", January 2014 [8]

Taking into consideration the above changes, the current research aims at understanding and validating the establishment of covert channels in a representative enterprise IPv6 implementations. In addition, most of the previous research studies were theoretical, without a proper proof of concept to validate the theory. To remediate this deficiency, the goal of this research required building a tool that could exploit the flaws in design and implementation of IPv6 and related protocols to create a covert channel, thereby providing proof of concept.

4. NETWORK DESIGN

The main part of the proof of concept was building a software tool that could covertly transfer data through an IPv6 network akin to the real-world networks. In order to simulate this, the network topology described in the figure 1 was used. This topology was designed similar to the one tested by [26] in his POC, as an uncontrolled network environment and proved to be working. The local site was designed with an Ubuntu PC (Linux kernel 3.19.0-26) which acted as a client that leaked information, a Cisco router (IOS 15.5(1)T) and a PFSense firewall (v2.2.4) with Snort (2.9.7.5 pkg v3.2.7) and Suricata (2.0.8 RELEASE pkg v2.1.6) packages installed, simulating a small business enterprise network. The PFSense firewall was connected on its WAN side to the IPv4 Internet. The PFSense firewall created a IPv6 over IPv4 tunnel to a tunnel broker maintained by Hurricane Electric [12]. The remote site consisted of a Ubuntu server (Linux kernel 3.19.0-26) that received the leaked data, and was behind a Cisco router (IOS 15.5(1)T). The Cisco router was connected on its WAN side to the IPv4 Internet and created an IPv6 over IPv4 tunnel to Hurricane Electric.

The PFSense firewall was configured as a potential small-to-medium enterprise network, with the some basic features such as NAT, access control and deep packet inspection of known protocols. The Snort IDS was configured with the subscription ruleset, which is "the same Snort ruleset developed for the NGIPS" and used by businesses, according to [31]. The Snort IDS was configured to use all categories of signatures published in the ruleset. This proprietary enterprise solution was complemented with Suricata, "a high performance open-source Network IDS, IPS and Network Security Monitoring engine" [33]. All the features of Suricata as described in [32] were enabled.

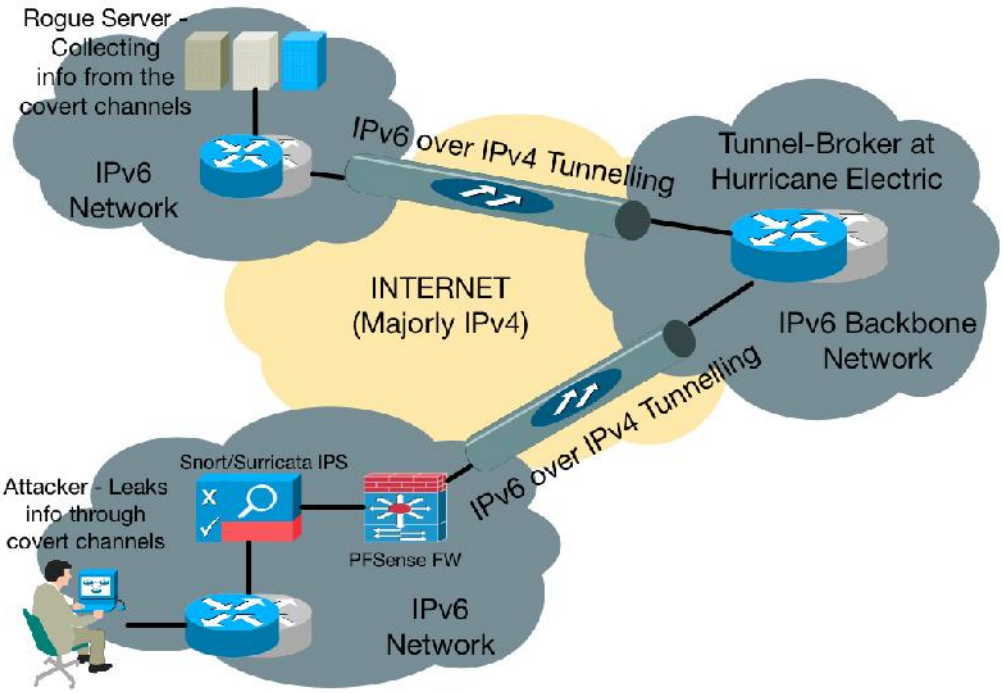

Figure 1 - Network Architecture for Testing

5. SOFTWARE DESIGN

The software tool called "Covertv6" was developed in Python using Scapy library, "a powerful interactive packet manipulation program that enables the user to send, sniff, dissect and forge network packets" [6]. Covertv6 was designed to exploit the following covert channels:

1. 1. IPv6 Flow Label [29, 16]
2. 2. IPv6 Extension Headers [5, 18, 4, 16]
3. 3. ICMPv6 [11, 34, 16]

Covertv6 was designed to run both as a client and a server in one of the following communication modes:

1. Beacon: where the server usually sends a beacon packet and the client leaks the data as a response to the beacon packet, thereby hiding the data in a legitimate communication channel.
2. Transfer: where the server is in listening mode all the time and the client leaks the data based on the configuration.

Covertv6 requires as input a file or a directory to be transferred and has the ability to specify the size of the payload (for a payload based exploit, otherwise uses constant values) and the rate at which the data should be leaked. The tool could also randomize the data leakage rate. The client initially sends a beacon packet to test the liveliness of the server and the reachability through the network. The beacon is usually an ICMPv6 echo request, but could also be a UDP packet with a zero checksum. The server sends a response and this signals the client to start the transfer. The client archives the file/folder in '.zip' format, to preserve the integrity of the files contained, throughout the binary data transfer process. The client then reads the bitstream of the zipped data and packs into a buffer set, each of the size as mentioned earlier. This buffered data is then encoded differently based on various exploits as follows.

The client sends a 'start of leak' beacon denoting the start of the data leak.

5.1 IPv6 Flow Label

The client encodes the 20-bit flow label value in an ICMPv6 echo request packet with the buffered data and sends it to the server at the rate con- figured (or at random intervals).

5.2 IPv6 Extension Headers

The client encodes the buffered data into one of the following extension headers as described below and sends it to the server at the rate configured (or at random intervals):

The client encodes the buffered data into one of the following extension headers as described below and sends it to the server at the rate configured (or at random intervals):

1. Hop-by-Hop Options Header - Using PadN to create jumbograms - This is a type of payload based covert channel
2. Destination Options Header - Encoding the data into the options field of the IPv6 header
3. Routing Header - Using experimental Routing Type to send type-specific data
4. Authentication Header - Encoding the data into the payload field of this header - This is a type of payload based covert channel
5. Encapsulating Security Payload Header - Encoding the data into the payload field of this header - This is a type of payload based covert channel
6. Custom Extension Header - Using the experimental header type values and encoding the data into the payload/value field of the header - This is a type of payload based covert channel

5.3 ICMPv6

While the previous two exploits worked on both modes of communication of the tool such as transfer and beacon modes, exploiting ICMPv6 required a different approach. Most of the ICMPv6 based covert channels worked based on the beacon mode where the covert packets would be created as a response to a packet they received. This required configuring the Firewall to allow ICMPv6 beacons from the server, which is often not the case in an enterprise network. The firewalls allowed packets through a pre-established connection. The tool exploited this by sending an ICMPv6 Echo Request packet first from the client, to which the server responds with an ICMPv6 Echo Reply, which acts as a beacon, thereby allowing the client to respond with one of the following ICMPv6 covert packets:

1. Destination Unreachable - The client sends the buffered data as pay- load in the 'destination unreachable' message with ICMPv6 type 1 code 0 - This is a type of payload based covert channel
2. Packet Too Big - The client sends the buffered data as payload in the ICMPv6 type 2 code 0 message - This is a type of payload based covert channel
3. Time Exceeded - The client sends the buffered data as payload in the ICMPv6 type 3 code 0 message - This is a type of payload based covert channel
4. Parameter Problem - The client sends the buffered data as payload in the ICMPv6 type 4 code 0 message, with the pointer set to '0xFF' - This is a type of payload based covert channel
5. Echo Request - The exploit using ICMPv6 Echo Request preferred the transfer mode and used the payload field to covertly send the data

6. Echo Reply - The exploit using ICMPv6 Echo Reply required a beacon with ICMPv6 being allowed through the Firewall, then sending the covert data in the payload

The server receives the packets and decodes the exploited field and adds it to its buffer for data reconstruction. As soon as all the data is transferred, the client sends a 'end of leak' beacon, and the server starts the reconstruction process. The server reconstructs the '.zip' file from the bitstream and extracts the archive to get the transferred data.

6. TESTING METHODOLOGY

The first step in the methodology was to determine if a covert channel could be established using each of the three IPv6 or ICMPv6 options described above. Once the establishment of each of those had been confirmed, each of the three main covert channels were tested further and their feasibility was observed under three conditions:

1. 1. Varying the size of the leaked data in the covert packets
2. 2. Varying the time interval of transfer between packets
3. 3. Varying the time randomness of transfer of the packets

For the purpose of the proof of concept, an ISO file of size 1MB and a directory of size 1MB containing text documents, were transferred through the covert channel. Figure 2 shows the working of Covertv6, while figure 3 shows a capture of one of the covert channel, with the 'start of leak' beacon packet.

```
lginod@ubuntu:~/Desktop$
lginod@ubuntu:~/Desktop$ sudo python covertv6.py
[sudo] password for lginod:
usage: covertv6.py [-h] [-l] [-x] [-i ICMP_OPT] [-e] [-s SRC] -d DST [-n SIZE]
                   [-t TIME_INTERVAL] -o OPMODE -m MODE -f FILENAME
covertv6.py: error: argument -d/--dst is required
lginod@ubuntu:~/Desktop$
lginod@ubuntu:~/Desktop$ sudo python covertv6.py -h
usage: covertv6.py [-h] [-l] [-x] [-i ICMP_OPT] [-e] [-s SRC] -d DST [-n SIZE]
                   [-t TIME_INTERVAL] -o OPMODE -m MODE -f FILENAME

optional arguments:
  -h, --help            show this help message and exit
  -l, --flow_label      Exploit using IPv6 Flow label
  -x, --ext_hdr         Exploit using IPv6 Extension headers
  -i ICMP_OPT, --icmp_opt ICMP_OPT
                        Exploit using ICMPv6 Options: 1. Destination
                        Unreachable 2. Packet too big 3. Time Exceeded 4.
                        Parameter Problem 128. Echo Request 129. Echo Reply
  -e, --encryption      Encryption On/Off
  -s SRC, --src SRC     Source IPv6 address
  -d DST, --dst DST     Destination IPv6 address
  -n SIZE, --size SIZE  Size of the payload (only if -i is set, default
                        1400Bytes)
  -t TIME_INTERVAL, --time_interval TIME_INTERVAL
                        Time interval
  -o OPMODE, --opMode OPMODE
                        Mode of Operation: 0. Client 1. Server 2. Forwarder
  -m MODE, --mode MODE  Mode of Transfer: 0. Transfer 1. Beacon 2. Forward
  -f FILENAME, --filename FILENAME
                        Path of the file to be transferred
lginod@ubuntu:~/Desktop$
lginod@ubuntu:~/Desktop$ sudo python covertv6.py -d 2004::128 -o 0 -m 0 -f doc/ -i 128 -t 1
.
Sent 1 packets.
Exploiting using ICMPv6:
Using ICMPv6 Echo Request
```

Figure 2. Covert6 Tool Options

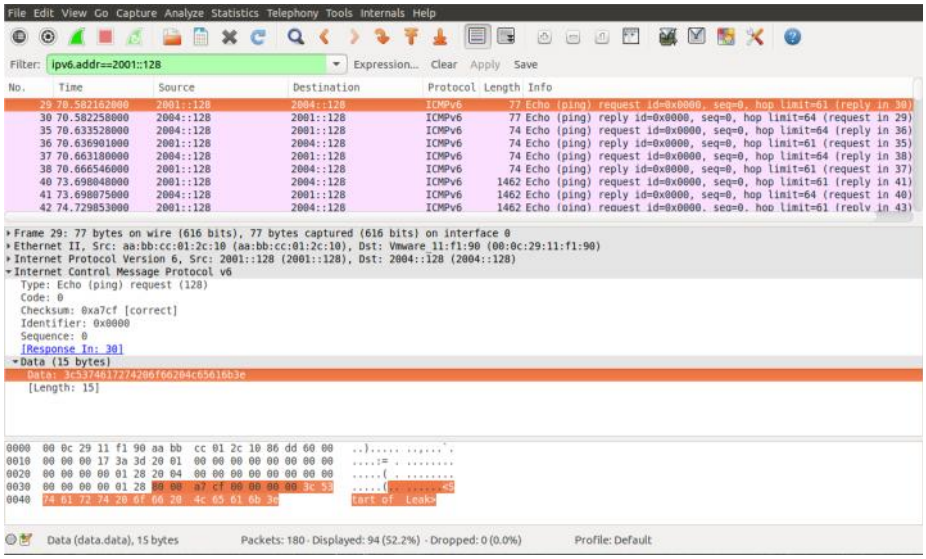

Figure 3. Start of Leak Frame

7. RESULTS AND DISCUSSION

The preliminary findings from the proof of concept are shown in Table 1. This shows the different attributes that were used as exploits in order to create the covert channels in IPv6, the communication modes of the tool that the exploits used, the possibility of a successful data transfer over the setup in the above three conditions.

Table 1
Test Results of Covert6

	Exploits for Covert Channels	Mode	Variable Payload Size	Variable Transfer Rate
1	*IPv6 Flow Label*	Both	✓	✓
2	*IPv6 Extension Headers*			
	Hop-by-Hop Options	Both	✓	✓
	Destination Options	Both	✓	✓
	Routing	Both	✓	✓
	Authentication	Both	✓	✓
	Encapsulate Security Payload	Both	✓	✓
	Custom Extension	Both	✓	✓

	Exploits for Covert Channels	Mode	Variable Payload Size	Variable Transfer Rate
3	*ICMPv6*			
	Destination Unreachable	Both[1]	✓	✓
	Packet Too Big	Both[1]	✓	✓
	Time Exceeded	Both[1]	✓	✓
	Parameter Problem	Both[1]	✓	✓
	Echo Request	Transfer	✓	✓
	Echo Reply	Both[1]	✓	✓

[1]Beacon Mode Preferred

7.1 IPv6 Flow Label

In RFC 6437, [29] established the ability to exploit the 20-bit flow label in IPv6 header to create covert channels and how it would not be detected by conventional means. The results of this study corroborate the exploit of the flow label field to create covert channels.

7.2 IPv6 Extension Headers

[13], [20], [3], [21], and [18] made changes to the IPv6 Extension Headers defined by RFC 2460 and noted that a forwarding device in the path of a packet containing extension headers must conform to the standards and not drop the packets if they do not understand those headers (for example, experimental headers). Taking that into consideration, the tool exploited various fields in the Extension Headers. The findings suggest that the creation of covert channels by exploiting these Extension Headers is still possible.

7.3 ICMPv6

The findings suggest that the potential creation of covert channels by exploiting various field and messages of ICMPv6. The following observations can also be made:

1. Although RFC 4443 [11] mandates that "for security reasons, it is recommended that implementations SHOULD allow sending of ICMP destination unreachable messages to be disabled, preferably on a per-interface basis," it is observed that implementations still allow 'destination unreachable' messages. Also, it could be observed that most implementations do not check the payload field of the message which should ideally be part of the invoking packet
2. RFC 4443 [11] states that "unlike other messages, a 'packet too big' message is sent in response to a packet received with an IPv6 multicast destination address, or with a link-layer multicast or link- layer broadcast address," thereby making this exploit unfeasible. But RFC 6946 [14] explains that a huge packet could still invoke the 'packet too big' message if it has to be fragmented

by an intermediate node. This helps in exploiting the payload of the message

3. Since RFC 4443 [11] states that "the pointer in the 'parameter problem' message will point beyond the end of the ICMPv6 packet if the field in error is beyond what can fit in the maximum size of an ICMPv6 error message," it could be observed that most implementations allow any data in the payload of the 'parameter problem' message
4. The RFC 4443 did not mandate implementations to allow/block the 'time exceeded', 'echo request', and 'echo reply' and it depended on the network configuration of the path.
5. All the ICMPv6 based covert channels worked in transfer mode through a poorly configured Firewall/IDS.

8. CONCLUSION

The current research strived to demonstrate a proof of concept by designing a tool to create covert channels in IPv6, thereby providing corroboration of its existence, in a simulated enterprise infrastructure consisting of enterprise Routers, Firewalls, IDS, and Internet. The preliminary findings from the research suggest the possible creation of covert channels by exploiting the IPv6 'flow label' field, IPv6 extension headers such as, the 'hop-by-hop options' header, 'destination options' header, 'routing' header, 'authentication' header, 'encapsulating security payload' header and custom extension header, and various message types of ICMPv6 such as, the 'destination unreachable,' 'packet too big,' 'time exceeded,' 'parameter problem,' 'echo request,' and 'echo reply.' The current study did not exploit all possible covert channels in IPv6 and did not attempt to encrypt the covert channel, since they are out of scope of the current research. Future work will include a study on the behavior and response of the enterprise Firewall and IDS solution to covert channels in IPv6, an attempt at corroborating other possible covert channels in IPv6 and a research on encrypted covert channels.

REFERENCES

[1] Joe Abley, Pekka Savola, and George Neville-Neil. Rfc 5095 deprecation of type 0 routing headers in ipv6. Technical report, 2007.

[2] Kamran Ahsan. Covert channel analysis and data hiding in TCP/IP. PhD thesis, University of Toronto, 2002.

[3] Jari Arkko and Scott Bradner. Rfc 5871 iana allocation guidelines for the ipv6 routing header. 2010.

[4] Antonios Atlasis. Security impacts of abusing ipv6 extension headers. In Black Hat security conference, pages 1–10, 2012.

[5] Manav Bhatia, Suresh Krishnan, James Hoagland, and Erik Kline. Rfc 6564 a uniform format for ipv6 extension headers. 2012.

[6] Philippe Biondi. About scapy. Retrieved from http://www.secdev.org/pro jects/scapy/doc/introduction.ht ml#about-scapy, apr 2010.

[7] R Bonica, D Gan, D Tappan, and C Pignataro. Rfc 4884 extended icmp to support multi-part messages. Technical report, 2007.

[8] Ron Bonica, Vishwas Manral, and Fernando Gont. Rfc 7112 implications of oversized ipv6 header chains. 2014.

[9] Phil Chimento, Marshall Eubanks, and Magnus Westerlund. Rfc 6935 ipv6 and udp checksums for tunneled packets. 2013.

[10] Lorenzo Colitti and Steinar H Gunderson. Global ipv6 statistics- measuring the current state of ipv6 for ordinary users. In RIPE 57 Meeting, 2008.

[11] Alex Conta and Mukesh Gupta. Rfc 4443 internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. 2006.

[12] Hurricane Electric. Hurricane electric tunnel broker. Retrieved from https://tunnelbroker.net, aug 2015.

[13] Bill Fenner. Rfc 4727 experimental values in ipv4, ipv6, icmpv4, icmpv6, udp, and tcp headers. 2006. 12

[14] Fernando Gont. Rfc 6946 processing of ipv6 atomic fragments. 2013.

[15] Thomas Graf. Messaging over ipv6 destination options. Retrieved from http://gray-world.net/papers/messip6.txt, 2003.

[16] Robert Hinden and S Deering. Rfc 2460 internet protocol, version 6 (ipv6) specification. 1998.

[17] Ed Jankiewicz, John Loughney, and Thomas Narten. Rfc 6434 ipv6 node requirements. 2011.

[18] Sheng Jiang and Brian Carpenter. Rfc 7045 transmission and processing of ipv6 extension headers. 2013.

[19] Merike Kaeo. Ipv6 routing header security - ripe54 - tallinn, estonia. Retrieved from http://meetings.ripe.net/ripe-54/presentations/IPv6 Routing Header.pdf, may 2007.

[20] Suresh Krishnan. Rfc 5722 handling of overlapping ipv6 fragments. 2009.

[21] Suresh Krishnan, J Woodyatt, Erik Kline, James Hoagland, and Manav Bhatia. Rfc

6564 a uniform format for ipv6 extension headers. Technical report, 2012.

[22] Butler W Lampson. A note on the confinement problem. Communications of the ACM, 16(10):613–615, 1973.

[23] Donald C Latham. Department of defense trusted computer system evaluation criteria. Department of Defense, 1986.

[24] David Llamas, C Allison, and A Miller. Covert channels in internet protocols: A survey. In Proceedings of the 6th Annual Postgraduate Symposium about the Convergence of Telecommunications, Networking and Broadcasting, PGNET, volume 2005, 2005.

[25] Norka B Lucena, Grzegorz Lewandowski, and Steve J Chapin. Covert channels in ipv6. In Privacy Enhancing Technologies, pages 147–166. Springer, 2006.

[26] RP Murphy. Ipv6/icmpv6 covert channels, 2006.

[27] Number Resource Organization. Free pool of ipv4 address space depleted. Retrieved from https://www.nro.net/news/ipv4-free-pool- depleted, feb 2011.

[28] J Rajahalme, A Conta, and B Carpenter. Rfc 3697 ipv6 flow label specification. 2004.

[29] Jarno Rajahalme, Shane Amante, Sheng Jiang, and Brian Carpenter. Rfc 6437 ipv6 flow label specification. 2011.

[30] Craig H Rowland. Covert channels in the tcp/ip protocol suite. First Monday, 2(5), 1997.

[31] Snort. Snort - explanation of rules. Retrieved from https://www.snort.org/rules explanation, aug 2015.

[32] Suricata. Complete list of suricata features. Retrieved from http://suricata-ids.org/features/all-features/, aug 2015.

[33] Suricata. Suricata ids. Retrieved from http://suricata-ids.org, aug 2015. 13 [34] Daniel C Tappan, Carlos Pignataro, Ronald P Bonica, and Der-Hwa Gan. Rfc 4884 extended icmp to support multi-part messages. 2007.

