Hedgehog in the Fog: Creating and Detecting IPv6 Transition Mechanism-Based Information Exfiltration Covert Channels

Bernhards Blumbergs, Mauno Pihelgas, Markus Kont, Olaf M. Maennel and Risto Vaarandi

Hedgehog in the Fog: Creating and Detecting IPv6 Transition Mechanism-Based Information Exfiltration Covert Channels

(Author copy, December 20, 2016)

Bernhards Blumbergs*, Mauno Pihelgas*, Markus Kont*, Olaf M. Maennel† and Risto Vaarandi†
*NATO Cooperative Cyber Defence Centre of Excellence
Email: name.surname[a]ccdcoe.org
†Tallinn University of Technology
Email: name.surname[a]ttu.ee

***Abstract*—The Internet Protocol Version 6 (IPv6) transition opens a wide scope for potential attack vectors. Tunnel-based IPv6 transition mechanisms could allow the set-up of egress communication channels over an IPv4-only or dual-stack network while evading detection by a network intrusion detection system (NIDS). Increased usage of IPv6 in attacks results in long-term persistence, sensitive information exfiltration, or system remote control. Effective tools are required for the execution of security operations for assessment of possible attack vectors related to IPv6 security.**

In this paper, we review relevant transition technologies, describe two newly-developed IPv6 transition mechanism-based proof-of-concept tools for the establishment of covert information exfiltration channels, and compare their performance against common tunneling mechanisms. We evaluated commonly used exfiltration tools in an automated and virtualized environment, and assessed covert channel detection methods in the context of insider threat.

An analysis of the generated test cases confirms that IPv6 and various evasion techniques pose a difficult task for network security monitoring. While detection of various transition mechanisms is relatively straightforward, other evasion methods prove more challenging. Additionally, some solutions do not yet fully support IPv6.

***Index Terms*—Computer network operations, Evasion techniques, Information exfiltration, IPv6 security, IPv6 transition, Covert channels, Monitoring, Red teaming**

I. Introduction

In this work we explore possible uses of IPv6 transition technologies for creation of covert channels over dual-stack and native IPv4 connectivity to exfiltrate information for red teaming [1] purposes. An analysis in Section II shows that this approach is novel and no implementations of such newly-developed tools have been identified previously.

The main contributions of this paper are:

1) two novel approaches for covert channel creation with IPv6 transition mechanisms;
2) development of proof-of-concept tools that implement the proposed methods (nc64 and tun64);
3) commonly-used protocol tunneling and developed proof-of-concept tool detection comparison table (Appendix A); and
4) a reproducible virtual lab environment providing detection results using open-source network security monitoring tools.

The Internet is in a period of tremendous growth, currently evolving toward the Internet of Anything (IoA). Early 2011 saw depletion of the pool of Internet Protocol Version 4 (IPv4) addresses available from the Internet Assigned Numbers Authority (IANA), an expansion which was anticipated and preceded by IPv6 specification [2] in 1998. The more widely-deployed IPv4 standard and IPv6 are incompatible, and they can communicate only via transition mechanisms and technologies [3] [4]. This introduces an additional layer of complexity and inherent security concerns for the transition and co-existence period [5]. Approaches for the IPv6 transition include dual-stack, address translation, and configured and dynamic tunneling [6], as well as other common encapsulation and packet transmission mechanisms, such as generic routing encapsulation (GRE) and multi-protocol label switching (MPLS).

The adoption of IPv6, and availability per the core backbone of the Internet infrastructure and edge networks, varies [7]. It has been observed that "[...] IPv6 is largely deployed at the core but lags in edge networks [...]" [8]. Alongside IPv6 launch campaigns (e.g. World IPv6 Day in 2011, and World IPv6 Launch in 2012), the autonomous systems (AS) announcing IPv6 prefix are rapidly increasing[1] [2]. Even though IPv6 usage is increasing with the yearly rate of over 400%, it still makes up around 1% of measured Internet packets [8], and IPv6 accessible content on the Internet is a minority [4] [9] [10]. In this context, connecting to the IPv6 Internet while maintaining scalability and minimal overall complexity would require edge networks to depend on transition mechanisms [4]. IPv6 deployment is based on Internet Service Provider (ISP) and Internet Content Provider (ICP) technical readiness, implementation knowledge, commercial considerations and consumer demands [8], possibly meaning that local area

[1]IPv6 Enabled Networks, RIPE NCC. http://v6asns.ripe.net/v/6 (Accessed 15/04/2016)

[2]IPv6 CIDR Report. http://www.cidr-report.org/v6/as2.0/ (Accessed 15/04/2016)

networks (LAN) will continue to use primary IPv4 for an undefined period.

IPv6 protocol implementations and security solutions are relatively new, already supported by default by modern operating systems, and have not yet reached the level of acceptable quality and maturity [4] [11]. In many cases, system engineers are not fully aware of them [12]. This lack of expertise and technological maturity result in IPv6 being considered in most cases as a "back-door" protocol, allowing bypass of network access controls, evasion of security mechanisms, and circumvention of security policies [13] [14]. This is important particularly when an attack originates from inside the network, as network security devices are commonly configured and placed on the perimeter under the assumption that intruders will always come from outside [15], making insider attacks more severe, damaging, and harder to detect [16].

In the age of advanced high-profile targeted attacks [17] [18] executed by sophisticated and resourceful adversaries, IPv6 is seen as an additional vector for persistent and covert attacks. Such threats include malware [19] [11] and advanced persistent threat (APT) actors [20]. Advanced threats typically pursue cyber-espionage campaigns for collection and exfiltration of sensitive information, gaining control over target network systems, and executing long-term persistence or fast-paced "hit-and-run" operations. The length of the transition period cannot be estimated, and it can be assumed that even once the entire Internet is native IPv6, there will still be systems running deprecated IPv6 functionality specifications, heritage transition mechanisms, or even IPv4 which already at that time could be potentially used as a "back-door" protocol.

Our research shows that current Network Intrusion Detection System (NIDS) solutions have serious drawbacks for handling IPv6 traffic. Addressing these shortcomings would require redevelopment of the principles how NIDSs reassemble packet streams, and correlation of distinct sessions. The described IPv6 transition-based methods (i.e. nc64 and tun64) use both IP version implementations in the same protocol stack. Attribution of these connections to a covert channel is therefore difficult. By comparison, common protocol tunneling approaches (e.g. SSH, DNS) would be easier to detect by an automated solution or human analyst since their behavior pattern is well known and understood.

In this paper, Section II reviews background and related work on IPv6 based network security, device evasion mechanisms, and covert channels; Section III describes common protocol tunneling approaches and newly-developed attack tool implementation and design; Section IV describes the attack scenario, simulation environment, and generated test cases; Section V discusses experiment execution results (presented in Table II), summarized discussions with commercial intrusion detection and data leakage prevention (DLP) system vendors, and additionally gives recommendations for such attack detection and mitigation possibilities; and Section VI offers conclusions.

Throughout, we adhere to IETF RFC2119 terminology, being aware of inherent discrepancies and ambiguities in this terminology across technical research papers and IETF documents [21].

II. Background and related previous work

The aim for IPv6 was to to evolve and eliminate the technical drawbacks and limitations of the IPv4 standard. However, IPv6 reintroduced almost the same security issues and, moreover, added new security concerns and vulnerabilities [22]. Various publications on IPv6 security acknowledge at least the following known security issues:

1) extension header-based attacks (e.g. extension header chaining, large extension headers, unused headers, deprecated headers) [23] [24] [25];
2) fragmentation (e.g. atomic fragments, overlapping fragments) [11] [24] [25] [14];
3) ICMPv6 security issues (e.g. Neighbor Discovery, Router Advertisements, multicast listen discovery) [26] [23] [24] [27] [14];
4) stateless automatic address configuration (SLAAC) [26] [28] [29] [30] [14];
5) unauthorized IPv6 clients [26];
6) tunneling (e.g. 6over4, ISATAP, IPSec tunneling) [11] [23] [31] [27] [29] [32] [14];
7) transition mechanisms (e.g. tunneling, translation and dual-stack) [11] [23] [28] [22] [24] [31] [29] [14];
8) address space size (e.g. address allocations per subnet and their uniqueness) [23] [28] [22] [24] [30];
9) privacy concerns (e.g. EUI-64 addressing) [26] [14];
10) mobile IPv6 [24];
11) IPv6 preference over IPv4 (e.g. on dual-stack hosts) [23];
12) service misconfiguration (e.g. exposing internal hosts globally) [12] [25];
13) multiple IPv6 addresses per single host [12];
14) inconsistent IPv6 support [29] [31] [27]; and
15) protocol specification implementation ambiguities [23].

Current IPv6 attack tools, such as the *THC-IPv6* [13], *SI6-IPv6*[3], *Topera*[4], and *Chiron*[5] toolkits, include the majority of techniques for abuse of IPv6 vulnerabilities, and can be utilized for network security assessment and IPv6 implementation verification.

Within the scope of this paper, a covert channel is understood as "a network connection that disguises its byte stream as normal traffic" [33]. Protocol steganography [34] for hiding and side-channeling data in unused fields or encoding data in existing field values can be considered a valid technique for covert information exfiltration. However, for newly-developed tool implementations described in this paper, exfiltrated data is directly stored in the protocol payload. This being done in order to test and verify the developed techniques in principle without using additional obfuscation approaches, which could

[3]SI6 Networks' IPv6 Toolkit. http://www.si6networks.com/tools/ipv6toolkit/ (Accessed 10/11/2015)

[4]Topera IPv6 analysis tool: the other side. http://toperaproject.github.io/topera/ (Accessed 10/11/2015)

[5]Chiron. http://www.secfu.net/tools-scripts/ (Accessed 10/11/2015)

be implemented at later stages. Mileva et al. offer an up-to-date, comprehensive survey of the covert channels in the TCP/IP protocol stack [34] classified by the affected layer and protocol, assessing advantages, disadvantages, and defense mechanisms, if they exist.

A proof-of-concept tool, *v00d00N3t*, for establishment of covert channels over ICMPv6 [35] has demonstrated the potential for such approach, though it has not been released publicly. Techniques for evading NIDS based on mobile IPv6 implementations reveal that it is possible to trick NIDS using dynamically-changing communication channels [36]. Also, it could be viable to create a covert channel by hiding information within IPv6 and its extension headers [37]. Network intrusion detection system (NIDS) and firewall evasions based on IPv6 packet fragmentation and extension header chaining attacks, have been acknowledged [5] [38] [13]. Although current Requests for Comments (RFCs) have updated the processing of IPv6 atomic fragments [39], discarding overlapping fragments [40] and enforcing security requirements for extension headers [41] [30], these attacks will remain possible in the years ahead as vendors and developers sometimes fail to follow the RFC requirements or implement their own interpretation of them. General approaches for NIDS evasions have been described and analyzed [42] [43] [44] [45], with the basic principles behind evasions based on the entire TCP/IP protocol stack. Advanced evasion techniques (AETs) involve creating combinations of multiple atomic evasion techniques, potentially allowing evasion of detection by the majority of NIDS solutions [46]. Evasions are possible due to NIDS design, implementation and configuration specifics, and low network latency requirements [11]. This is partially related to a lack of accepted NIDS standards, although there have been some standardization and community-based attempts to establish common frameworks by the Defense Advanced Research Projects Agency (DARPA) (the Common Intrusion Detection Framework, or CIDF) and Internet Engineering Task Force (IETF) (the Intrusion Detection Working Group).

Existing approaches and technologies consider native IPv6 network implementation and connectivity, and do not take into account possible methods for network security device evasions and covert channel establishment over IPv6 transition mechanisms, in order to reach the command and control (CnC) servers over IPv4 only or dual-stack Internet connectivity. Moreover, to our knowledge no publicly available tools directly implement transition technology-based attacks. Here we address this gap and advance beyond previous work.

III. Covert channel implementations

A. Protocol tunneling

Protocol tunneling and IPv6 tunneling-based transition mechanisms pose a major security risk, as they allow bypassing of improperly-configured or IPv4-only network security devices [27] [22] [26] [14] [32]. IPv6 tunnel-based transition mechanisms, as well as general tunneling approaches (e.g. HTTP, SSH, DNS, ICMP, IPsec), can bypass network protection mechanisms. However, IPv6 tunnels simply add to the heap of possible tunneling mechanisms, leading to unmanaged and insecure IPv6 connections [27]. Moreover, dual-stack hosts and Internet browsers favor IPv6 over IPv4 [7], which in some cases raises security concerns as this may not be anticipated by network security personnel. Various protocol tunneling approaches can be used to set up a covert channel by encapsulating exfiltrated information in networking protocols, such as DNS [33], HTTP(S) [33], SSH [47] [48] [49] [50], ICMP [33], RTP [51], FTP [51], SSH over HTTP [52], and peer-to-peer [33].

Technologies such as VPN, TOR, i2p and related peer-to-peer mechanisms are not considered here, as they rely on dedicated sets of protocols, are highly specific, and infrastructure dependent. Their presence in a network can be detected (e.g. TOR exit nodes, i2p router) and might not be allowed by the network egress firewall policy, unless they are encapsulated in other protocols or otherwise tunneled.

Covert channels based on DNS, HTTP(S), SSH, and ICMP implementations are acknowledged here as the most common approaches for eluding network detection mechanisms, due to both their frequent use and standard network policy, which allows outbound protocols and ports for user requirements and remote network administration needs. The following approaches and their tool implementations have been selected for their availability in distributions for network security penetration testing (e.g. Kali Linux[6]):

1) DNS tunneling (iodine[7], dnscat2[8]);
2) HTTP tunneling (httptunnel[9]);
3) SSH tunneling (OpenSSH client and server[10], sshuttle[11], PuTTY[12]);
4) ICMP tunneling (ptunnel[13]); and
5) Netcat tunneling (Ncat[14]).

Although it is possible to manually script missing tool implementations (e.g. HTTP(S) or DNS tunneling over IPv6), we consider mature and publicly available tools herein.

B. Proof-of-concept nc64 tool

We have developed a proof-of-concept tool, nc64[15], for the creation of information exfiltration channel over dual-stack networks using sequential IPv4 and IPv6 sessions. The tool's source code is publicly available under MIT license[16].

Signature-based IDSs reassemble packets and data flows, in order to conduct inspection against a known signature

[6]Kali Linux 2.0. https://www.kali.org/ (Accessed 20/11/2015)

[7]Iodine. http://code.kryo.se/iodine/ (Accessed 20/11/2015)

[8]dnscat2. https://github.com/iagox86/dnscat2 (Accessed 20/11/2015)

[9]Arch Linux httptunnel repository. https://www.archlinux.org/packages/?name=httptunnel (Accessed 20/11/2015)

[10]OpenSSH. http://www.openssh.com/ (Accessed 20/11/2015)

[11]sshuttle. https://github.com/apenwarr/sshuttle (Accessed 20/11/2015)

[12]PuTTY: A Free Telnet/SSH Client. http://www.chiark.greenend.org.uk/~sgtatham/putty/ (Accessed 20/11/2015)

[13]Ping Tunnel. http://www.cs.uit.no/~daniels/PingTunnel/ (Accessed 20/11/2015)

[14]Nmap Ncat. https://nmap.org/ncat/ (Accessed 20/11/2015)

[15]nc64 https://github.com/lockout/nc64 (Accessed 12/03/2016)

[16]Open Source Initiative. The MIT License (MIT). https://opensource.org/licenses/MIT (Accessed 16/05/2016)

database. This is done on per-session basis (e.g. a TCP session). If the data is fragmented across multiple sessions, then the IDS cannot retrieve the full information to evaluate whether the traffic is malicious. In such scenario NIDS has to be context aware in order to be able to correlate and reconstruct the original stream from multiple sequential ones. This is very challenging due to performance considerations. While any set of networking protocols could be used for a sequential session creation, the security, transition, and immaturity of IPv6 makes it a preferred choice. When considering NIDS separate session correlation possibilities, IP protocol switching would make it harder since destination IPv4 and IPv6 addresses are different. In a dual-stack operating system, IPv4 and IPv6 protocols are implemented side by side, thus adding a layer of separation between the two standards and making it more difficult for IDSs to reassemble data. Additionally, a single host can have multiple global IPv6 addresses, making the correlation to a single host even harder.

Having IPv4 and IPv6 connections to the same remote host can be regarded as normal, as the resources on the destination server could be accessible over both IPv4 and IPv6. For example, a website might serve its main content over IPv6 and third-party content, such as banners, over IPv4, or vice-versa. Additionally, a single host can have multiple global IPv6 addresses, making the correlation to a single host harder, unless a single reverse DNS record exists for those addresses.

To exfiltrate data from the source host to a destination CnC server over sequential IPv4 and IPv6 sessions, the data must be split into smaller chunks in compliance with the dual-stack network MTU of 1500B. However, chunk size can be varied to achieve coherent information exfiltration in relation to specifics of the network and data. Alternation between IPv4 and IPv6 per session has to be controlled to minimize the amount of information that is sent over a single IP protocol in successive sessions (e.g. not allowing three or more sequential IPv4 sessions). This control would avoid partial reassembly and deny successful payload inspection by NIDS. Therefore, in our implementation, the default threshold is set to three successive same IP version sessions, with overall probability of IP version selection of 0.5.

A CnC server has both IPv4 and IPv6 addresses on which it listens for incoming connections. Once the connection is established, the listener service receives sessions and reassembles data in sequence of reception (Figure 1). This can be hard to accomplish if a stateless transport layer protocol is being used (i.e. UDP) or data chunk size exceeds the maximum path MTU (e.g. causing IPv4 packet fragmentation).

Another plausible option would be to establish multiple CnC servers and have each session redirected to a different one via a single IP version. However, such an approach would be less scalable, require more server management overhead, and make the data reassembly less efficient and reliable.

Our proof-of-concept tool, nc64, is written in Python version 3 using standard libraries. It implements the aforementioned principles, and additionally:

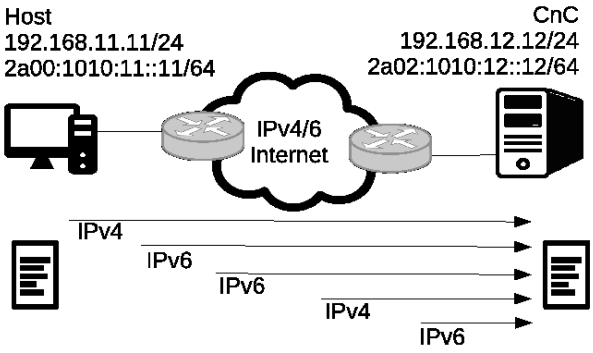

Fig. 1. nc64 sequential IPv4/6 information exfiltration

1) provides both the listener server and client part in one Python module;
2) accepts user-specified data from a standard input, which provides flexibility and freedom of usage;
3) requires both IPv4 and IPv6 addresses for the destination CnC listener, and can have a list of IPv6 addresses in case the CnC server has multiple IPv6 addresses configured;
4) supports UDP and TCP transport layer protocols, as these are the main ones used in computer networks;
5) enables the destination port to be freely selected to comply with firewall egress rules and match the most common outbound protocol ports (e.g. HTTP(S), DNS), and also allows for setting and randomizing of the source port for UDP-based communications;
6) provides payload Base64 encoding for binary data transmission, and to some degree can be treated as obfuscation if the IDS does not support encoding detection and decoding. It has to be noted that Base64-encoded traffic might reveal the exfiltrated data in the overall traffic since it would stand out, which would also apply when using payload encryption;
7) allows for the setting and randomizing of timing intervals between sequential sessions for an additional layer of covertness and to mitigate possible timing pattern prediction and detection by NIDS;
8) implements control over how many sequential sessions of the same protocol can be tolerated before forcing a switch to the other protocol, ensuring that small files are sent over both IP protocols; and
9) supports additional debugging features, exfiltrated data hash calculation, and transmission statistics.

Future improvements to the nc64 tool should implement additional control mechanisms, such as codes for identifying the exfiltrated block sequence number for correct reassembly and verification; variable payload block size specification and synchronization; legitimate traffic emulation (e.g. HTTP GET requests for HTTP traffic, SSL handshakes and certificates for HTTPS and SSH traffic); payload encryption; and transport layer protocol switching.

C. Proof-of-concept tun64 tool

We have developed a second proof-of-concept tool, tun64[17], which exfiltrates information by abusing tunneling-based IPv6 transition mechanism capabilities over the IPv4-only computer network. The tool's source code is publicly available under MIT license.

Most tunneling-based IPv6 transition mechanisms rely on IPv4 as a link layer by using 6in4 encapsulation [6], whereby an IPv6 packet is encapsulated in IPv4 and the protocol number is set to decimal value 41 (the IANA-assigned payload type number for IPv6). In 2013, 6in4-encapsulated traffic was estimated to make 90% of IPv6-tunneled traffic [8]. Besides 6in4 encapsulation, we also acknowledge GRE (protocol-47) [53] as an applicable encapsulation mechanism for 6in4-in-GRE double encapsulation. When 6in4 (protocol-41) encapsulation is used, duplex connectivity might not be possible if the network relies on strict NAT, and therefore stateless connections may be used over UDP instead of TCP or SCTP. Techniques for NAT hole punching [54] or autonomous NAT traversal [55] could potentially traverse NAT devices. However, for the attack scenario considered in this paper, a one-way communication channel for information exfiltration to the CnC server is sufficient, making UDP the preferred transport layer protocol [10].

Generic packet tunneling in IPv6 is described in RFC2473 [56], an overview and comparison of tunneling mechanisms are provided in RFC7059 [57], and security concerns for IP tunneling are reviewed in RFC6169 [58]. RFC4213 [6] defines two tunneling mechanisms: configured (i.e. static), with both tunnel endpoint IPv4 addresses predefined, and dynamic (i.e. automatic), whereby the tunnel source and destination IPv4 addresses are derived from IPv6 addresses. Based on specification, these can be divided into router-to-router, host-to-router, and router-to-host transition mechanisms [59] [14]. These include transition mechanisms such as: 6to4 automatic tunnel [60] (with anycast 6to4 relay routers deprecated by RFC7526 [61]), 6rd configured tunnel [62] [63], Softwire Mesh (RFC5565), 6over4 configured tunnel [64], ISATAP automatic tunnel [65] [57], Teredo automatic tunnel [66], SIT [67], and AYIYA (Anything-in-Anything) [68]. Additionally, tunnel brokers [69] (e.g. Hurricane Electric, SIXXS) provide IPv6 Internet access to hosts in the IPv4 Internet.

Most of the transition techniques cannot solve transition problems and hence are not appropriate for real-world implementation and widespread deployment [4]. Tunnel-based transition approaches are considered deprecated by the IETF, but nevertheless, some of these technologies continue to be supported by modern operating systems and ISPs.

The 6over4, ISATAP (figure 2), and 6to4 (figure 3) transition mechanisms were selected for implementation in our proof-of-concept tool for tunneling-based information exfiltration. Selection of these mechanisms was based upon the tunnel establishment from the target host or network, their support by either operating systems or network infrastructure devices

[17]tun64 https://github.com/lockout/tun64 (Accessed 12/03/2016)

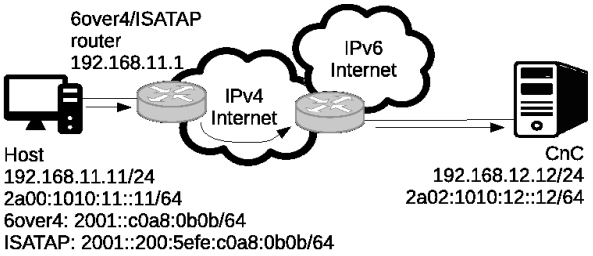

Fig. 2. 6over4 and ISATAP tunnel emulation

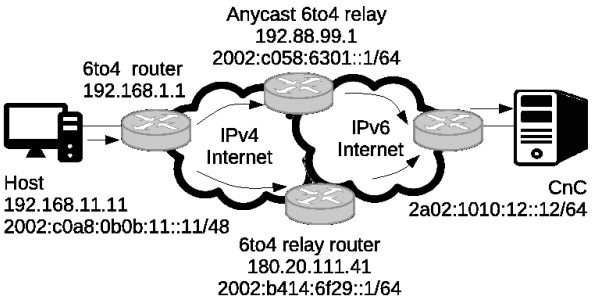

Fig. 3. 6to4 tunnel emulation

[57], and being independent of transition mechanisms implemented by an upstream ISP (e.g. 6rd).

Our proof-of-concept tool, tun64, is written in Python version 2 using the Scapy library[18]. It implements the aforementioned principles and additionally:

1) provides only the client part, thus relying on standard packet capture tools for reception and reassembly (e.g. tcpdump, Wireshark, tshark);
2) supports TCP, UDP, and SCTP as transport layer protocols;
3) emulates 6over4, 6to4, and ISATAP tunneling by assigning source and destination IPv6 addresses according to the transition protocol specification;
4) enables usage of 6to4 anycast relay routers if the tool is being tested in real Internet conditions, although in our simulated network, 6to4 relay routers or agents are not implemented;
5) allows additional GRE encapsulation to create a 6in4-in-GRE double encapsulated packet, which may allow obfuscation if the NIDS is not performing a full packet decapsulation and analysis;
6) gives an option to freely specify source and destination ports, in order to comply with firewall egress rules; and
7) supports sending a single message instead of files or standard input, a functionality designed with proof-of-concept approach in mind.

When observing tun64 operations by an advanced packet-

[18]Scapy project. http://www.secdev.org/projects/scapy/ (Accessed 10/11/2015)

capture tool, such as Wireshark[19], the traffic is represented as IPv6 or IPv6-GRE rather than IPv4-based 6in4 encapsulated traffic due to how Wireshark uses its sophisticated display filters and packet decoders. In some cases this might give an advantage to the attackers, as network monitoring personnel might misinterpret the actual traffic relying on the sophisticated capabilities of the packet capture tool.

IV. Testing environment and test description

A. Attack scenario

Our testing environment and experiments are designed according to the following scenario. The attack target is a small- to medium-sized research organization (up to 100 network nodes). Research organization assumes it is running an IPv4-only network, even though all the network hosts are dual-stack and their ISP just recently started to provide also IPv6 connectivity. Network administrators have implemented IPv4 security policies and only the following most common egress ports and services are allowed through the firewall: DNS (udp/53, tcp/53), HTTP (tcp/80), HTTPS (tcp/443), SSH (tcp/22), and ICMP (echo). All network hosts can establish a direct connection to the Internet without proxies or any other connection handlers. This organization was recently contracted by government to conduct advanced technological research and therefore has sensitive (i.e. not restricted or confidential) information processed and stored on the network hosts and servers.

The government hired a red team to perform a technical security test on the research organization in order to assess the possibilities for exfiltrating information and gaining infrastructure control. A red team, assuming the role of reasonably sophisticated attacker, has the goal of retrieving sensitive information related to recent advanced research projects or gaining unhindered access to the network services that store and process such information. For this purpose, the red team has already gained a persistent foothold within the computer network of the target organization by fully compromising at least one network host (e.g. a workstation, internal or edge server, or core router) via network infrastructure device targeting, client-side attacks, pivoting, and lateral movement. The tools for information exfiltration are deployed on this compromised host, which serves as the source node. The red team has a dual-stack CnC server deployed on the Internet under its full control, and this serves as the destination node. Since the team is interested in exfiltrating information, it might fall-back to a unidirectional communication channel if duplex connectivity cannot be established or is not desired due to operational requirements.

The red team has a selection of tools available at its disposal for the establishment of a covert information exfiltration channel, as described in Section III. Although the red team intends to perform data exfiltration over a long period of time, a requirement for immediate data extraction can be met if network security devices would become able to detect the covert channel.

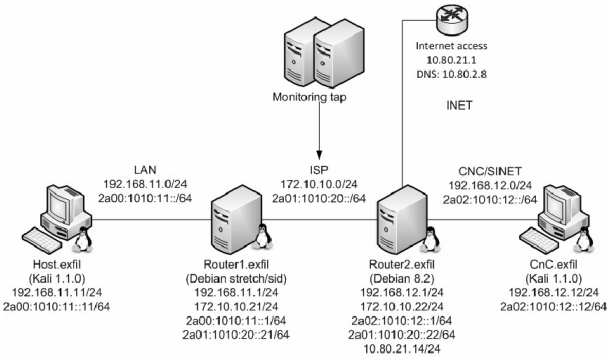

Fig. 4. Testing environment network map

B. Testing environment

To ensure reproducibility of the testbed, we created several *bash*[20] scripts that leverage the Vagrant[21] environment automation tool. The scripts are publicly available in a GitHub repository[22]. A network map of the virtual testing environment is presented in Figure 4.

The host and CnC devices were built on 32-bit Kali Linux 2.0, which comes bundled with several tunneling tools. Router1 served as the gateway for the target organization, and Router2 as an ISP node in the simulated Internet (SINET). Both routers were also built as authoritative DNS servers to facilitate usage of the Iodine tool, which was explicitly configured to query them during the tests. Two monitoring machines were built to provide detection capability. The first node was connected with a tap to the network link between the routers and all packets were copied to its monitoring interface. Second node was created to avoid conflicts between monitoring tools, and was therefore not used for capture.

In order to create identical testing conditions, we decided to store a packet capture (PCAP) file for each combination of the exfiltration tool, destination port number, transport layer protocol, and IP version. Additionally, several distinct operation modes were tested for the nc64 (e.g. both plaintext and base64 encoded payload) and tun64 (e.g. ISATAP, 6to4, and 6over4 tunneling mechanism emulation) tools, as these significantly impact the nature of the network traffic. Overall, 126 packet capture files were generated to be used as test cases. In the next phase we used the same monitoring nodes to run a selection of popular detection tools which would analyze these PCAP files, produce connection logs, and possibly generate alerts for suspicious activity.

We considered a number of open-source monitoring tools that are often used for network security analysis. These include

[19]Wireshark network protocol analyzer. https://www.wireshark.org/ (Accessed 10/11/2015)

[20]GNU Bourne-Again SHell. https://www.gnu.org/software/bash/ (Accessed 07/12/2015)

[21]Vagrant. https://www.vagrantup.com/ (Accessed 07/12/2015)

[22]Automated virtual testing environment. https://github.com/markuskont/exfil-testbench (Accessed 07/12/2015)

the signature-based NIDSs Snort[23] and Suricata[24], as well as the network traffic analyzers Bro[25] and Moloch[26]. For Suricata, we used the Emerging Threats (ET) ruleset, while for Snort we experimented with rulesets from both SourceFire (SF) and ET signature providers.

In our tests, the data exfiltrated from the host system comprise the highly sensitive */etc/shadow* file and the *root* user's private *SSH* cryptographic keys. Both of which could be used for gaining unauthorized access to potentially many other systems in the organization.

V. Experiment execution and discussion of results

The results of the experiments are summarised in Table I. We also present the comprehensive results in an extensive table (see Table II) in Appendix A. Each row in Table II describes a single attack, while the columns represent a detection tool that was used to attempt its detection.

TABLE I
SUMMARISED RESULTS OF THE EXPERIMENTS

Type	**#**	**%**
Positive matches	122	19.4%
Partial matches	92	14.6%
Potential visual matches	117	18.6%
Failed detection	299	47.4%
Total test iterations	*630*	*100%*

In our results, we distinguished four potential outcomes for a test:

1) a positive match (denoted by letter *Y* and a green cell in table II) was clearly identified as malicious activity with appropriate alerts;
2) a partial or abnormal footprint (*P* and yellow cell) which raised an alert, but the alert did not describe the activity appropriately;
3) a potential visible match (*V* and orange cell) from connection logs which requires human analysis or sophisticated anomaly detection for a positive match; and
4) in the worst case, no visible alerts nor connection logs were generated (*N* and red cell).

Firstly, we observed that any exfiltration tool utilizing a specific application layer protocol should adhere to its standard port numbers if the malicious user aims to evade detection. For example, a HTTP tunnel on port 22 triggered an *outbound SSH Scan* alert with the ET ruleset, whereas when port 80 was used, only HTTP connection logs were generated such that we classified the attack as being only *visible*. Note that we marked the *outbound SSH Scan* alert for the HTTP tunnel on port 22 only as a *partial* match because it was incorrectly identified as an outbound SSH connection. Additionally, the same rule was responsible for a partial match against the nc64 technique on port 22. Furthermore, an alert was raised if a SSH header was detected on port 443, or if that port was used to send unencrypted HTTP traffic. Similarly, if abnormal (non-DNS) traffic was identified on UDP port 53, the ET ruleset triggered alerts for either *non-compliant traffic to DNS protocol*, or for being *overly aggressive* (i.e., having too many connections). These signatures were easily bypassed if TCP port 53 was used.

However, it has to be noted that most server applications can be bound to any applicable port number (e.g. SSH on tcp/2022, HTTPS console over tcp/8443), and thus can potentially be used to avoid or obscure detection.

The difference between SF and ET rulesets in their default configurations is significant. The former seems to focus solely on perimeter intrusions, and hence could not detect any malicious outbound traffic in our tests. Note, that the poor results from our experiments with the SF ruleset severely influenced the overall statistics. Taking this side note into consideration, the other tool and ruleset combinations produced much better detection rates. Furthermore, the ET ruleset produced slightly different results in Snort and Suricata. Most importantly, the former could clearly identify ICMP Ptunnel as the tool used for traffic exfiltration.

Bro does not employ any traditional signatures like Snort or Suricata, but does create logs for all identified connections. As such, it was able to produce log records of all test cases. However, although Bro does not generate alerts, it does have an interesting log file named *weird.log* wherein a record of detected anomalous connections is kept. In fact, during our attacks, several *weird.log* records were generated for non-compliant traffic on port 53. Additionally, Bro's SSH connection parser malfunctioned while processing non-standard traffic, and abnormal logs could be observed in the detection system.

Moloch provides no alerts, but is designed as a packet capture, indexing and visualization tool. In the most recent release, at the time of conducting the experiment, Moloch does not support IPv6 due to various limitations when indexing 128-bit IP addresses[27]. Therefore, IPv6-only iterations were unnoticed while IPv4 sessions generated by nc64 in dual-stack configuration were *visible*. The *t6to4* mode in tun64 encapsulates the IPv6 packet as payload making it visible in IPv4 indexing system. This was observed only in cases of TCP connections without additional GRE encapsulation.

From the executed test results, detection of malicious activity by NIDS rules was based predominantly on the direction of network traffic, protocol, and destination port. This detection approach is generally favored because it uses resources (e.g. CPU, RAM) efficiently, with an expensive payload analysis attempted only after the preceding match conditions are achieved.

[23]Snort v2.9.8.0. http://manual.snort.org/ (Accessed 07/12/2015)

[24]Suricata v2.1beta4. http://suricata-ids.org/docs/ (Accessed 07/12/2015)

[25]Bro v2.4.1 https://www.bro.org/documentation/index.html (Accessed 07/12/2015)

[26]Moloch v0.12.1. https://github.com/aol/moloch (Accessed 07/12/2015)

[27]Moloch 0.14.0 2016/05/08 CHANGELOG specifies a notice that "[IPv6] support is experimental, and will change with ES 5.0." https://github.com/aol/moloch/blob/master/CHANGELOG (Accessed 16/08/2016)

In most cases, the nc64 tool avoided being detected, and Table II shows which protocol/port combinations can be used to minimize detection by selected NIDS solutions. In comparison with other exfiltration tools, nc64 performed very well on avoiding rule-based detection, and moreover could potentially elude payload inspection.

In contrast, the tun64 tool was detected in the majority of cases, since protocol-41 and protocol-47 triggered the rules and generated warning messages by NIDSs. 6to4 tunneling emulation was detected when TCP or 6in4-in-GRE encapsulation was used, suggesting that double encapsulation is considered more suspicious. However, if an organization relies on IPv6 tunneling-based transition mechanisms utilizing 6in4 or GRE encapsulation, such warnings might be silenced or ignored by network-monitoring personnel. In contrast to other tunneling tools the approach taken by tun64 is feasible only if the network conditions comply with the specific operational requirements.

A. Discussion with vendors

In order to compare and assess how commercial solutions perform in comparison with the open-source ones used in the experiment, vendors were approached to get their virtual appliances for a limited trial period. Three major NIDS and DLP product vendors agreed on supporting the research to execute additional tests against the generated PCAP test cases. These tests were conducted separately, outside the developed testing environment, in close collaboration with the vendors.

An anonymous summary of the additional experiment results and discussions is the following:

1) IPv6 support is often not implemented due to customers not requesting it;
2) Many monitoring tools were initially built with no IPv6 support in mind, and enabling it requires a fundamental redesign of how 128-bit IPv6 addresses are processed (e.g. not as strings or numbers in hexadecimal or decimal form);
3) Vendors often use open-source detection tools in their products;
4) Automated near real-time detection, traffic decapsulation, and payload decoding is not performed due to high resource requirements and introduction of additional network latency;
5) Protocol analysis, payload inspection and extraction is supported only for commonly used plain-text protocols (e.g. HTTP, SMTP, FTP);
6) Manual asynchronous traffic analysis could be done by human analyst whenever detection algorithms are incapable of parsing the data or sorting out the false positives; and
7) Warnings on suspicious protocol behavior (e.g. no SSH traffic on port tcp/22) or RFC non-compliant traffic (e.g. no TCP three way handshake completed) could potentially be configured to be silently discarded or ignored by system administrator, thus leading to attacks not being observed.

It has to be noted that any commercial product which uses an open-source tool for data acquisition is subjected to same limitations of the respective tool. Also, the lack of knowledge regarding IPv6 exploitation methods translate into low customer demand which leads to insufficient IPv6 support in final products. Furthermore, any reasonably sophisticated data exfiltration method which splits that data into smaller chunks and extracts the resulting pieces using different connections/flows (e.g. IPv4 and IPv6) will be very hard to detect in real-time by existing NIDS, which typically lack any capability to correlate across different connections/flows. Finally, commercial tools are often too expensive for small and medium sized organizations. Therefore, we did not consider these products in our final evaluation.

B. Cyber defense exercise results

To further test and verify the detection possibilities of nc64 created covert channels by other technical solutions and expert analysts, it was integrated as a side challenge into a bigger digital forensic challenge. This was executed within the technical cyber defense exercise Locked Shields 2016 which is "[...] the biggest and most advanced international live-fire cyber defence exercise in the world [...], organised [annually] since 2010 by the Tallinn-based NATO Cooperative Cyber Defence Centre of Excellence [...]."[28] Network digital forensics challenge consisted of approximately 4GB large PCAP file, where traffic between numerous hosts on different networks was captured. Among other activities this packet capture contained multiple exfiltrations of sensitive private SSH keys via nc64 tool.

Exercise participants, twenty teams (i.e. the Blue Teams) from NATO nations and Partner countries, each consisting of lead cyber defense and digital forensic experts where given the chance to participate in digital forensics challenge. Upon completion of the forensic challenge, Blue Teams had to prepare and submit reports. To verify the covert channel detection, Blue Teams were provided with the following questions to be answered in their report:

1) which IP addresses are engaged in exfiltration execution (please specify all identified source and destination IP addresses);
2) what approach is used to create a covert channel used to exfiltrate information from the target network;
3) if possible, can you identify and extract the exfiltrated sensitive administrative information?

The results of the digital forensics challenge for covert channel identification and analysis is summarized in the table III in appendix B. In the table, the following notation is used: 'Yes' denotes a correct answer, 'No' – an incorrect one, '–' means that no answer was provided, and 'Partial' represents that answer is partially correct or only some parts of technical information were correctly presented.

[28]NATO CCD CoE, Locked Shields 2016. https://ccdcoe.org/locked-shields-2016.html (Accessed 24/11/2016)

From the results it can be seen, that only one team out of twenty was able to fully present all technical information and identify the principle for covert channel established by nc64 tool. Three other teams were able to fully extract technical information, but were not able to identify or correctly formulate what principle is being used for such channel establishment. Also, it can be determined, that nearly half of the teams did not even attempt solving this particular digital forensic challenge, which might mean either their lack of expertise or this challenge being prioritized as low and resources allocated to main exercise engagement. Furthermore, the remaining teams mostly provided no or partial technical information, either being able to extract only IPv4 or IPv6 addresses, or just addresses for the destination CnC host; as well as presenting inaccurate identification of exfiltration operation general principle, such as, base64 encoded exfiltration, SSL over UDP tunneling, multiple machines (when actually only two hosts were engaged in exfiltration), or usage of UDP over common ports allowed by firewall.

The main point of this forensic challenge was to verify if experienced human analyst or group of experts would be able to properly identify, analyze and defend against an unknown threat. This turned out to be very difficult as the teams were struggling with this task.

C. Anomaly detection considerations

According to experiments described in [70] and [71] for organizational networks, most users of such networks consistently access the same set of services over longer periods of time, and services accessed in the recent past by the user can be leveraged for predicting the future behavior of that user. In paper [70], an algorithm is described which maintains a list of frequently accessed services for each user, and employs these lists for detecting user behavior anomalies in private networks. To verify how well such approach might work in the context of Internet services, we obtained a HTTP traffic log from a large EU institution that covers 60 days and describes accessing of Internet websites on 1379 workstations. We discovered that each workstation contacted an average of 135.57 hosts per day over HTTP (we identified hosts by their IP addresses). When we investigated how frequently the same hosts were accessed by workstations, we found that from all hosts contacted by a workstation during the 60-day period, 32.85% hosts were accessed on at least 10% of the days (6 days), 20.45% are accessed on at least 25% of the days (15 days), and 13.03% of hosts are accessed during on at least 50% of the days (30 days). In contrast, a typical workstation accesses an average of 64.73% hosts only during 1 day. These findings suggest that although long-term HTTP usage patterns can be identified for workstations, such patterns nevertheless do not describe a significant part of the workstation HTTP traffic, and pattern detection cannot predict the entire future behavior of a workstation.

VI. Conclusions and further work

In this paper, the authors addressed a fundamental problem which could allow to bypass NIDSs by using the IPv6 tunneling-based and dual-stack transition mechanisms in a certain way. The proof-of-concept tools were prototyped to further verify under which circumstances the evasion of major open-source and commercial NIDS and monitoring solutions would be possible. Developed tools, tested along side with other well known protocol tunneling tools, proved to be able to evade detection and addressed certain shortcomings in the core principles of how modern NIDSs work.

It has to be noted, that any reasonably sophisticated method for exfiltrating data will be hard to detect in real-time by existing NIDSs, especially in situations where the data is split into smaller chunks and the resulting pieces use different connections or protocols (e.g. IPv4 and IPv6). Detecting such activity would require the capability to correlate the detection information in near real-time across different flows. This is theoretically possible, but would most likely incur a significant performance penalty and an increased number of false positives. There are several possibilities to attempt correlating flows using both IPv4 and IPv6 protocols. If the destination host (i.e. CnC) used in multi-protocol exfiltration has a DNS entry for both A and AAAA records, it would be possible to perform a reverse lookup to identify that the connections are going to the same domain name using IPv4 and IPv6 protocols simultaneously. This should not happen under normal circumstances, since IPv6 is usually the preferred protocol on dual-stack hosts. If no IPv6 privacy extensions (RFC4941) are used, then host's NIC MAC address would be used to generate a unique IPv6 SLAAC EUI-64 (RFC2373) address. Such kind of generated address in certain conditions could be used in order to link data connections to a physical host. Another option would be to rely solely on source NIC MAC address for aggregating and correlating flows from both IPv4 and IPv6 which are originating from the the same network interface. Note, that this requires capturing the traffic from the network segment where the actual source node resides, otherwise source MAC address might get overwritten by network devices in transit. One caveat still remains — distinguishing the flows which are belonging together, especially on busy hosts with many simultaneous connections. Finally, behavior based detection (e.g. unexpected traffic, malformed packets, specification non-compliance) would provide a way to detect such evasions, at the same time introducing a significant amount of false positives.

Authors acknowledge, that the tendency of use of IPv6 in attack campaigns conducted by sophisticated malicious actors is going to increase. Since IPv6 security aspects are being addressed by protocol RFC updates and deprecation of obsolete transition mechanisms, it would be required to focus on these issues at the security solution developer (i.e. vendor) and implementer (i.e. consumer) levels. Adding IPv6 support to the security devices would not solve this problem, since fundamental changes would be required in the way how

network traffic is interpreted and parsed, while being able to trace the context of various data streams and perform their correlation. Also, end-users should know how to properly configure, deploy and monitor security solutions in order to gain maximum awareness of the computer network flows under their direct supervision.

Potential future research directions would include advanced insider threat detection, IPv6 protocol stack implementation analysis in the modern operating system kernels and in embedded device micro-kernels.

VII. Acknowledgements

This research was conducted with the support of NATO Cooperative Cyber Defence Centre of Excellence, Tallinn University of Technology, and Estonian IT Academy (StudyITin.ee). The authors would like to acknowledge the valuable contribution of Leo Trukšāns, Walter Willinger, and Merike Käo.

This research has been presented and published in 21st Nordic Conference NordSec 2016 proceedings Springer Lecture Notes in Computer Science (LNCS): doi:10.1007/978-3-319-47560-8_6.

References

[1] P. Brangetto, E. Çalişkan, and H. Rõigas, *Cyber Red Teaming - Organisational, technical and legal implications in a military context*. NATO CCD CoE, 2015.

[2] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, IETF Secretariat, December 1998. Standards Track.

[3] R. Tadayoni and A. Henten, "Transition from IPv4 to IPv6," in *23rd European Regional Conference of the International Telecommunication Society*, July 2012.

[4] P. Wu, Y. Cui, J. Wu, J. Liu, and C. Metz, "Transition from IPv4 to IPv6: A State-of-the-Art Survey," *IEEE Comm. Surveys and Tutorials*, vol. 15, no. 3, pp. 1407–1424, 2013.

[5] A. Atlasis, "Attacking IPv6 Implementation Using Fragmentation," tech. rep., Centre for Strategic Cyberspace + Security Science, 2011.

[6] E. Nordmark and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers," RFC 4213, IETF Secretariat, October 2005. Standards Track.

[7] L. Colitti, S. H. Gunderson, E. Kline, and T. Refice, "Evaluating IPv6 Adoption in the Internet," in *PAM 2010*, pp. 141–150, Springer-Verlag, 2010.

[8] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, and M. Bailey, "Measuring IPv6 Adoption," in *ACM SIGCOMM14*, 2014.

[9] M. Nikkhah, R. Guérin, Y. Lee, and R. Woundy, "Assessing IPv6 Through Web Access: A Measurement Study and Its Findings," in *ACM CoNEXT 2011*, December 2011.

[10] N. Sarrar, G. Maier, B. Ager, R. Sommer, and S. Uhlig, "Investigating IPv6 Traffic: What Happened at the World IPv6 Day?," in *13th International Conference, PAM 2012* (N. Taft and F. Ricciato, eds.), pp. 11–20, Springer-Verlag, March 2012.

[11] Fortinet, "Biting the Bullet: A Practical Guide for Beginning the Migration to IPv6," white paper, Fortinet Inc., 2011.

[12] A. Turiel, "IPv6: new technology, new threats," *Network Security*, pp. 13–15, August 2011.

[13] F. Gont and M. Heuse, "Security Assessments of IPv6 Networks and Firewalls." IPv6 Congress 2013, 2013. Presentation.

[14] S. Hogg and E. Vyncke, *IPv6 Security*. Cisco Press, 2009.

[15] A. H. M. Taib and R. Budiarto, "Evaluating IPv6 Adoption in the Internet," in *5th Student Conference on Research and Development*, IEEE, December 2007.

[16] U.S. DHS, "Common Cybersecurity Vulnerabilities in Industrial Control Systems," tech. rep., U.S. Department of Homeland Security, May 2011.

[17] TrendLabs, "Targeted Attack Trends 2014 Report," tech. rep., TrendMicro, 2015.

[18] National Cybersecurity and Communications Integration Center, "ICS-CERT Monitor," tech. rep., US Dep. of Homeland Security, December 2013.

[19] G Data SecurityLabs, "Uroburos: Highly complex espionage software with Russian roots," tech. rep., G Data Software AG, February 2014.

[20] B. Blumbergs, "Technical Analysis of Advanced Threat Tactics Targeting Critical Information Infrastructure," *Cyber Security Review*, pp. 25–36, 2014.

[21] J. Palet, "6in4 versus 6over4 terminology," tech. rep., IETF Secretariat, November 2005. Internet Draft.

[22] S. Convery and D. Miller, "IPv6 and IPv4 Threat Comparison and Best-Practice Evaluation," white paper, Cisco Systems, March 2004.

[23] D. McPherson, "Eight Security Considerations for IPv6 Deployment," tech. rep., Verisign Inc., 2011.

[24] J. Ullrich, K. Krombholz, H. Hobel, A. Dabrowski, and E. Weippl, "IPv6 Security: Attacks and Countermeasures in a Nutshell," in *USENIX WOOT'14*, SBA Research, 2014.

[25] C. Ottow, F. van Vliet, P. T. de Boer, and A. Pras, "The Impact of IPv6 on Penetration Testing," in *18th IFIP International EUNICE Conference on Information and Communications Technologies*, August 2012.

[26] G. of the HKSAR, "IPV6 Security," tech. rep., The Government of the Hong Kong Special Administrative Region, May 2011.

[27] S.Degen et.al., "Testing the security of IPv6 implementations," tech. rep., Ministry of Economic Affairs of the Netherlands, March 2014.

[28] F. Gont, "Results of a Security Assessment of the Internet Protocol version 6 (IPv6)." Hack.lu 2011 Conference, 2011. Presentation.

[29] US-CERT, "Malware Tunneling in IPv6," security advisory, US-CERT, May 2005.

[30] F. Gont, W. Liu, and R. Bonica, "Transmission and Processing of IPv6 Options," tech. rep., IETF Secretariat, March 2015. Best Current Practice.

[31] S. Frankel, R. Graveman, J. Pearce, and M. Rooks, "SP 800-119: Guidelines for the Secure Deployment of IPv6," special publication, National Institute of Standards and Technology, December 2010.

[32] F. Gont, "Security Implications of IPv6 on IPv4 Networks," RFC 7123, Feb 2014.

[33] B. Blunden, *The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System, 2nd ed.*, ch. 14. Covert Channels. Jones and Bartlett Learning, 2013.

[34] A. Mileva and B. Panajotov, "Covert channels in TCP/IP protocol stack," *Central European Journal of Computer Science*, vol. 4, no. 2, pp. 45–66, 2014.

[35] R. Murphy, "IPv6 / ICMPv6 Covert Channels." DEF CON'14, 2014. Presentation.

[36] M. Colajanni, L. D. Zotto, M. Marchetti, and M. Messori, "Defeating NIDS evasion in Mobile IPv6 networks," in *IEEE*, 2011.

[37] N. B. Lucena, G. Lewandowski, and S. J. Chapin, "Covert Channels in IPv6," in *PET 2005* (G. Danezis and D. Martin, eds.), pp. 147–166, Springer-Verlag, 2006.

[38] A. Atlasis, "Security Impacts of Abusing IPv6 Extension Headers," tech. rep., Centre for Strategic Cyberspace + Security Science, 2012.

[39] F. Gont, "Processing of IPv6 "Atomic" Fragments," RFC 6946, May 2013.

[40] S. Krishnan, "Handling of Overlapping IPv6 Fragments," RFC 5722, IETF Secretariat, December 2009. Standards Track. Updates RFC 2460.

[41] S. Krishnan, J. Woodyatt, E. Kline, J. Hoagland, and M. Bhatia, "A Uniform Format for IPv6 Extension Headers," tech. rep.

[42] S. Pastrana, J. Montero-Castillo, and A. Orfila, *Advances in Security Information Management: Perceptions and Outcomes*, ch. 7. Evading IDSs and firewalls as fundamental sources of information in SIEMS. Nova Science Publishers, Jan 2013.

[43] A. Atlasis and E. Rey, "Evasion of High-End IPS Devices in the Age of IPv6," tech. rep., secfu.net, 2014.

[44] J. M. Vidal, J. D. M. Castro, A. L. S. Orozco, and L. J. G. Villalba, "Evolutions of Evasion Techniques Aigainst Network Intrusion Detection Systems," in *ICIT 2013, The 6th International Conference on Information Technology*, May 2013.

[45] V. Bukač, "IDS System Evasion Techniques," Master's thesis, Masarykova Univerzita Fakulta Informatiky, 2010.

[46] O. P. Niemi, A. Levomki, and J. Manner, "Dismantling Intrusion Prevention Systems," in *ACM SIGCOMM12*, August 2012.

[47] W. Ellens, P. Żuraniewski, A. Sperotto, H. Schotanus, M. Mandjes, and E. Meeuwissen, "Covert Channels in IPv6," in *Flow-Based Detection of DNS Tunnels* (G. Doyen, M. Waldburger, P. Čeleda, A. Sperotto,

and B. Stiller, eds.), vol. 7943 of *Lecture Notes in Computer Science*, pp. 124–135, Springer Berlin Heidelberg, 2013.
[48] P. Butler, K. Xu, and D. D. Yao, "Quantitatively analyzing stealthy communication channels," in *Proceedings of the 9th International Conference on Applied Cryptography and Network Security*, ACNS'11, (Berlin, Heidelberg), pp. 238–254, Springer-Verlag, 2011.
[49] H. Bhanu, J. Schwier, R. Craven, R. Brooks, K. Hempstalk, D. Gunetti, and C. Griffin, "Side-Channel Analysis for Detecting Protocol Tunneling," in *Advances in Internet of Things*, vol. 1, pp. 13–26, July 2011.
[50] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Detection of Encrypted Tunnels Across Network Boundaries," in *International Conference on Communications*, pp. 1738–1744, IEEE, May 2008.
[51] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *Communications Surveys Tutorials, IEEE*, vol. 9, pp. 44–57, September 2007.
[52] S. Prowell, R. Kraus, and M. Borkin, *Seven Deadliest Network Attacks*, ch. 4. Protocol Tunneling, pp. 59–73. Syngress, 2010.
[53] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784, IETF Secretariat, March 2000. Standards Track. Supplemented with RFC2890.
[54] D. Maier, O. Haase, J. Wäsch, and M. Waldvogel, "NAT Hole Punching Revisited," in *36th Annual IEEE Conference on Local Computer Networks*, pp. 147–150, 2011.
[55] A. Müller, N. Evans, C. Grothoff, and S. Kamkar, "Autonomous NAT Traversal," in *IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, August 2010.
[56] A. Conta and S. Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473, IETF Secretariat, December 1998. Standards Track.
[57] S. Steffann, I. van Beijnum, and R. van Rein, "A Comparison of IPv6-over-IPv4 Tunnel Mechanisms," RFC 7059, IETF Secretariat, November 2013. Informational.
[58] S. Krishnan, D. Thaler, and J. Hoagland, "Security Concerns with IP Tunneling," RFC 6169, IETF Secretariat, April 2011. Informational.
[59] Microsoft, "IPv6 Transition Technologies," windows server 2008 white paper, Microsoft Corp., 2008.
[60] K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," RFC 3056, IETF Secretariat, February 2001. Standards Track.
[61] O. Troan and B. Carpenter, "Deprecating the Anycast Prefix for 6to4 Relay Routers," RFC 7526, IETF Secretariat, May 2015. Best Current Practice.
[62] W. Townsley and O. Troan, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) –Protocol Specification," RFC 5969, IETF Secretariat, August 2010. Standards Track.
[63] R. Despres, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)," RFC 5569, IETF Secretariat, January 2010. Informational.
[64] B. Carpenter and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," RFC 2529, IETF Secretariat, March 1999. Standards Track.
[65] F. Templin, T. Gleeson, and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)," RFC 5214, IETF Secretariat, March 2008. Informational.
[66] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," RFC 4380, IETF Secretariat, February 2006. Standards Track.
[67] R. E. Gilligan, "Simple Internet Transition Overview," tech. rep., IETF Secretariat, November 1994. Internet Draft.
[68] J. Massar, "AYIYA: Anything In Anything," tech. rep., IETF Secretariat, July 2004. Internet Draft.
[69] A. Durand, P. Fasano, I. Guardini, and D. Lento, "IPv6 Tunnel Broker," RFC 3053, IETF Secretariat, Jauary 2001. Informational.
[70] R. Vaarandi, "Detecting Anomalous Network Traffic in Organizational Private Networks," in *IEEE CogSIMA Conference*, pp. 285–292, 2013.
[71] J. McHugh and C. Gates, "Locality - A New Paradigm for Thinking About Normal Behavior and Outsider Threat," in *New Security Paradigms Workshop*, pp. 3–10, 2003.

APPENDIX A

TABLE II: Protocol tunneling and data exfiltration tool assessment

Iteration	IP Version	Protocol	Port	Snort SF	Snort ET	Suricata	Bro	Moloch
http-22	4	TCP	22	N	P	P	P	V
http-443	4	TCP	443	N	Y	Y	V	V
http-53	4	TCP	53	N	Y	Y	P	V
http-80	4	TCP	80	N	N	V	V	V
Iodine	4	UDP	53	N	N	Y	P	V
nc64-t-22-4-b64	4	TCP	22	N	P	P	V	V
nc64-t-22-4	4	TCP	22	N	P	P	V	V
nc64-t-22-64-b64	4+6	TCP	22	N	P	P	V	V
nc64-t-22-64	4+6	TCP	22	N	P	P	V	V
nc64-t-22-6-b64	6	TCP	22	N	P	P	V	N
nc64-t-22-6	6	TCP	22	N	P	P	V	N
nc64-t-443-4-b64	4	TCP	443	N	N	N	V	V
nc64-t-443-4	4	TCP	443	N	N	N	V	V
nc64-t-443-64-b64	4+6	TCP	443	N	N	N	V	V
nc64-t-443-64	4+6	TCP	443	N	N	N	V	V
nc64-t-443-6-b64	6	TCP	443	N	N	N	V	N
nc64-t-443-6	6	TCP	443	N	N	N	V	N
nc64-t-53-4-b64	4	TCP	53	N	N	N	P	V
nc64-t-53-4	4	TCP	53	N	N	N	P	V
nc64-t-53-64-b64	4+6	TCP	53	N	N	N	P	V
nc64-t-53-64	4+6	TCP	53	N	N	N	P	V
nc64-t-53-6-b64	6	TCP	53	N	N	N	P	N
nc64-t-53-6	6	TCP	53	N	N	N	P	N
nc64-t-80-4-b64	4	TCP	80	N	N	N	P	V
nc64-t-80-4	4	TCP	80	N	N	N	P	V
nc64-t-80-64-b64	4+6	TCP	80	N	N	N	P	V
nc64-t-80-64	4+6	TCP	80	N	N	N	P	V
nc64-t-80-6-b64	6	TCP	80	N	N	N	P	N
nc64-t-80-6	6	TCP	80	N	N	N	P	N
nc64-u-22-4-b64	4	UDP	22	N	N	N	V	V
nc64-u-22-4	4	UDP	22	N	N	N	V	V
nc64-u-22-64-b64	4+6	UDP	22	N	N	N	V	V
nc64-u-22-64	4+6	UDP	22	N	N	N	V	V
nc64-u-22-6-b64	6	UDP	22	N	N	N	V	N
nc64-u-22-6	6	UDP	22	N	N	N	V	N
nc64-u-443-4-b64	4	UDP	443	N	N	N	V	V
nc64-u-443-4	4	UDP	443	N	N	N	V	V
nc64-u-443-64-b64	4+6	UDP	443	N	N	N	V	V
nc64-u-443-64	4+6	UDP	443	N	N	N	V	V
nc64-u-443-6-b64	6	UDP	443	N	N	N	V	N
nc64-u-443-6	6	UDP	443	N	N	N	V	N
nc64-u-53-4-b64	4	UDP	53	N	Y	Y	P	V
nc64-u-53-4	4	UDP	53	N	Y	Y	P	V
nc64-u-53-64-b64	4+6	UDP	53	N	Y	Y	P	V
nc64-u-53-64	4+6	UDP	53	N	Y	Y	P	V
nc64-u-53-6-b64	6	UDP	53	N	Y	Y	P	N
nc64-u-53-6	6	UDP	53	N	Y	Y	P	N
nc64-u-80-4-b64	4	UDP	80	N	N	N	V	V
nc64-u-80-4	4	UDP	80	N	N	N	V	V

TABLE II: Protocol tunneling and data exfiltration tool assessment

Iteration	IP Version	Protocol	Port	Snort SF	Snort ET	Suricata	Bro	Moloch
nc64-u-80-64-b64	4+6	UDP	80	N	N	N	V	V
nc64-u-80-64	4+6	UDP	80	N	N	N	V	V
nc64-u-80-6-b64	6	UDP	80	N	N	N	V	N
nc64-u-80-6	6	UDP	80	N	N	N	V	N
netcat-t-22-4	4	TCP	22	N	N	N	V	V
netcat-t-22-6	6	TCP	22	N	N	N	V	N
netcat-t-443-4	4	TCP	443	N	N	N	V	V
netcat-t-443-6	6	TCP	443	N	N	N	V	N
netcat-t-53-4	4	TCP	53	N	N	N	P	V
netcat-t-53-6	6	TCP	53	N	N	N	P	N
netcat-t-80-4	4	TCP	80	N	N	N	V	V
netcat-t-80-6	6	TCP	80	N	N	N	V	N
netcat-u-22-4	4	UDP	22	N	N	N	V	V
netcat-u-22-6	6	UDP	22	N	N	N	V	N
netcat-u-443-4	4	UDP	443	N	N	N	V	V
netcat-u-443-6	6	UDP	443	N	N	N	V	N
netcat-u-53-4	4	UDP	53	N	Y	Y	P	V
netcat-u-53-6	6	UDP	53	N	Y	Y	P	N
netcat-u-80-4	4	UDP	80	N	N	N	V	V
netcat-u-80-6	6	UDP	80	N	N	N	V	N
ptunnel	4	ICMP		N	Y	N	V	V
ssh-4-22	4	TCP	22	N	N	V	V	V
ssh-4-443	4	TCP	443	N	Y	Y	V	V
ssh-4-53	4	TCP	53	N	N	V	V	V
ssh-4-80	4	TCP	80	N	N	V	P	V
ssh-6-22	6	TCP	22	N	N	V	P	N
ssh-6-443	6	TCP	443	N	Y	Y	P	N
ssh-6-53	6	TCP	53	N	N	V	P	N
ssh-6-80	6	TCP	80	N	N	V	P	N
tun64-t-22-isatap	4	TCP	22	N	Y	Y	P	N
tun64-t-22-t6over4	4	TCP	22	N	Y	Y	P	N
tun64-t-22-t6to4	4	TCP	22	N	Y	Y	P	V
tun64-t-443-isatap	4	TCP	443	N	Y	Y	P	N
tun64-t-443-t6over4	4	TCP	443	N	Y	Y	P	N
tun64-t-443-t6to4	4	TCP	443	N	Y	Y	P	V
tun64-t-53-isatap	4	TCP	53	N	Y	Y	P	N
tun64-t-53-t6over4	4	TCP	53	N	Y	Y	P	N
tun64-t-53-t6to4	4	TCP	53	N	Y	Y	P	V
tun64-t-80-isatap	4	TCP	80	N	Y	Y	P	N
tun64-t-80-t6over4	4	TCP	80	N	Y	Y	P	N
tun64-t-80-t6to4	4	TCP	80	N	Y	Y	P	V
tun64-u-22-isatap	4	UDP	22	N	Y	Y	P	N
tun64-u-22-t6over4	4	UDP	22	N	Y	Y	P	N
tun64-u-22-t6to4	4	UDP	22	N	Y	Y	P	N
tun64-u-443-isatap	4	UDP	443	N	Y	Y	P	N
tun64-u-443-t6over4	4	UDP	443	N	Y	Y	P	N
tun64-u-443-t6to4	4	UDP	443	N	Y	Y	P	N
tun64-u-53-isatap	4	UDP	53	N	Y	Y	P	N
tun64-u-53-t6over4	4	UDP	53	N	Y	Y	P	N
tun64-u-53-t6to4	4	UDP	53	N	Y	Y	P	N
tun64-u-80-isatap	4	UDP	80	N	Y	Y	P	N

TABLE II: Protocol tunneling and data exfiltration tool assessment

Iteration	IP Version	Protocol	Port	Snort SF	Snort ET	Suricata	Bro	Moloch
tun64-u-80-t6over4	4	UDP	80	N	Y	Y	P	N
tun64-u-80-t6to4	4	UDP	80	N	Y	Y	P	N
tun64-t-22-isatap-gre	4	TCP	22	N	Y	Y	P	N
tun64-t-22-t6over4-gre	4	TCP	22	N	Y	Y	P	N
tun64-t-22-t6to4-gre	4	TCP	22	N	Y	Y	P	V
tun64-t-443-isatap-gre	4	TCP	443	N	Y	Y	P	N
tun64-t-443-t6over4-gre	4	TCP	443	N	Y	Y	P	N
tun64-t-443-t6to4-gre	4	TCP	443	N	Y	Y	P	V
tun64-t-53-isatap-gre	4	TCP	53	N	Y	Y	P	N
tun64-t-53-t6over4-gre	4	TCP	53	N	Y	Y	P	N
tun64-t-53-t6to4-gre	4	TCP	53	N	Y	Y	P	V
tun64-t-80-isatap-gre	4	TCP	80	N	Y	Y	P	N
tun64-t-80-t6over4-gre	4	TCP	80	N	Y	Y	P	N
tun64-t-80-t6to4-gre	4	TCP	80	N	Y	Y	P	V
tun64-u-22-isatap-gre	4	UDP	22	N	Y	Y	P	N
tun64-u-22-t6over4-gre	4	UDP	22	N	Y	Y	P	N
tun64-u-22-t6to4-gre	4	UDP	22	N	Y	Y	P	V
tun64-u-443-isatap-gre	4	UDP	443	N	Y	Y	P	N
tun64-u-443-t6over4-gre	4	UDP	443	N	Y	Y	P	N
tun64-u-443-t6to4-gre	4	UDP	443	N	Y	Y	P	V
tun64-u-53-isatap-gre	4	UDP	53	N	Y	Y	P	N
tun64-u-53-t6over4-gre	4	UDP	53	N	Y	Y	P	N
tun64-u-53-t6to4-gre	4	UDP	53	N	Y	Y	P	V
tun64-u-80-isatap-gre	4	UDP	80	N	Y	Y	P	N
tun64-u-80-t6over4-gre	4	UDP	80	N	Y	Y	P	N
tun64-u-80-t6to4-gre	4	UDP	80	N	Y	Y	P	V

APPENDIX B

TABLE III
LOCKED SHIELDS 2016 DIGITAL FORENSIC EXFILTRATION CHALLENGE OVERVIEW

Blue Team No.	Identified IPv4 addresses	Identified IPv6 addresses	nc64 approach described	Exfiltrated data extracted
1	No	No	No	Partial
2	–	–	–	–
3	**Yes**	**Yes**	No	**Yes**
4	No	**Yes**	No	Partial
5	–	–	–	–
6	–	–	–	–
7	**Yes**	**Yes**	No	**Yes**
8	**Yes**	**Yes**	**Yes**	**Yes**
9	Partial	No	No	No
10	–	–	–	–
11	**Yes**	Partial	No	Partial
12	–	–	–	–
13	–	–	–	–
14	No	Partial	No	Partial
15	–	–	–	–
16	**Yes**	No	No	Partial
17	–	–	–	–
18	**Yes**	**Yes**	Yes	**Yes**
19	–	–	–	–
20	**Yes**	No	No	No