Creating and Detecting IPv6 Transition Mechanism-Based Information Exfiltration Covert Channels

Bernhards Blumbergs[1(✉)], Mauno Pihelgas[1], Markus Kont[1], Olaf Maennel[2], and Risto Vaarandi[2]

[1] NATO Cooperative Cyber Defense Center of Excellence, Tallinn, Estonia
{bernhards.blumbergs,mauno.pihelgas,markus.kont}@ccdcoe.org
[2] Tallinn University of Technology, Tallinn, Estonia
{olaf.maennel,risto.vaarandi}@ttu.ee

Abstract. The Internet Protocol Version 6 (IPv6) transition opens a wide scope for potential attack vectors. IPv6 transition mechanisms could allow the set-up of covert egress communication channels over an IPv4-only or dual-stack network, resulting in full compromise of a target network. Therefore 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 and analyze two newly-developed IPv6 transition mechanism-based proof-of-concept tools for the establishment of covert information exfiltration channels. The 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.

Keywords: IPv6 security · IPv6 transition · Covert channels · Computer network operations · Red teaming · Monitoring and detection

1 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 [6] purposes. An analysis in Sect. 2 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. fully self-developed 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). The more widely-deployed IPv4 standard and IPv6 are incompatible, and they can communicate only via transition mechanisms and technologies [38,44]. This introduces an additional layer of complexity and inherent security concerns for the transition and co-existence period [1]. The adoption of IPv6, and availability per the core backbone of the Internet infrastructure and edge networks, varies [10,12]. IPv6 launch campaigns rapidly increased the number of autonomous systems (AS) announcing IPv6 prefix[1,2]. Nevertheless, connecting to the IPv6 Internet while maintaining scalability and minimal overall complexity often means that edge networks deploy various transition mechanisms [36,44], possibly meaning that local area networks (LANs) 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 [15,44]. The lack of expertise and technological maturity result in IPv6 being considered in most cases as a "back-door" protocol, allowing evasion of security mechanisms [21,23]. 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 [39].

In the age of advanced high-profile targeted attacks executed by sophisticated and resourceful adversaries, IPv6 is seen as an additional vector for persistent and covert attacks [29,41]. 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, or heritage transition mechanisms.

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, Sect. 2 reviews background and related work, evasion mechanisms, and covert channels; Sect. 3 describes common protocol tunneling approaches and newly-developed attack tool implementation and design; Sect. 4

[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).

describes the attack scenario, simulation environment, and generated test cases; Sect. 5 discusses experiment execution results (presented in Table 1), and additionally gives recommendations for such attack detection and mitigation possibilities; and Sect. 6 offers conclusions and future research directions.

2 Background and Related Previous Work

The aim for IPv6 was 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 [11,19]. Current IPv6 attack tools, such as the *THC-IPv6* [21], *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.

Already in 1998, Ptacek and Newsham in their research paper [33] showed that NIDS evasions are possible and pose a serious threat. A proof-of-concept tool, *v00d00N3t*, for establishment of covert channels over ICMPv6 [28] 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 [9]. Also, it could be viable to create a covert channel by hiding information within IPv6 and its extension headers [26]. Network intrusion detection system (NIDS) and firewall evasions based on IPv6 packet fragmentation and extension header chaining attacks, have been acknowledged [1,2,21]. Although current Requests for Comments (RFCs) have updated the processing of IPv6 atomic fragments [17], discarding overlapping fragments [24] and enforcing security requirements for extension headers [20,25], 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 [3,7,32,43], 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 [30]. Evasions are possible due to NIDS design, implementation and configuration specifics, and low network latency requirements [15].

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

[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).

over IPv4 only or dual-stack Internet connectivity. To the best of our knowledge no publicly available tool implements transition technology-based attacks.

3 Covert Channel Implementations

3.1 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 [11,18,22,23,35]. 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 add to the heap of possible tunneling mechanisms, leading to unmanaged and insecure IPv6 connections [35]. Moreover, dual-stack hosts and Internet browsers favor IPv6 over IPv4 [10]. Various protocol tunneling approaches can be used to set up a covert channel by encapsulating exfiltrated information in networking protocols. Covert channels based on DNS, HTTP(S), ICMP [5], and SSH [13] protocol tunneling 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. For the purposes of the test cases we consider mature and publicly available tools for protocol tunneling establishment.

3.2 Proof-of-Concept Nc64 Tool

We have developed a proof-of-concept tool, nc64[6], 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.

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

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

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 (i.e. up to IPv6 MTU of 1500B). 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 version 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.

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. 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 packet fragmentation).

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

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 an optional 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.

3.3 Proof-of-Concept Tun64 Tool

We have developed a second proof-of-concept tool, tun64[7], which exfiltrates information by abusing tunneling-based IPv6 transition mechanism capabilities

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

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 [31], 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). Besides 6in4 encapsulation, we also acknowledge GRE (protocol-47) [14] 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. However, for the attack scenario considered in this paper (see Sect. 4.1), a one-way communication channel for information exfiltration to the CnC server is sufficient, making UDP the preferred transport layer protocol [34].

Most of the transition techniques cannot solve transition problems and hence are not appropriate for real-world implementation and widespread deployment [44]. Although tunnel-based transition approaches are considered deprecated by the IETF, some of these technologies continue to be supported by modern operating systems and ISPs. The 6over4 [8], ISATAP [37,40], and 6to4 [27,42] 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 local network infrastructure devices [37].

Our proof-of-concept tool, tun64, is written in Python 2.7 using the Scapy library[8]. 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.

4 Testing Environment and Test Description

4.1 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

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

(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 information processed and stored on the network hosts and servers. A red team, assuming the role of reasonably sophisticated attacker with persistent foothold in the research organization's network, is tasked to exfiltrate sensitive information from the target network. The red team has a selection of tools available at its disposal for the establishment of a covert information exfiltration channel, as described in Sect. 3.

4.2 Testing Environment

To ensure reproducibility of the testbed, we created several *bash* scripts that leverage the Vagrant[9] environment automation tool. The scripts are publicly available in a GitHub repository[10]. A network map of the virtual testing environment is presented in Fig. 1.

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 plain-text 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.

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

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

We considered a number of open-source monitoring tools that are often used for network security analysis. These include the signature-based NIDSs Snort[11] and Suricata[12], as well as the network traffic analyzers Bro[13] and Moloch[14]. For Suricata, we used the Emerging Threats (ET) ruleset, while for Snort we experimented with rulesets from both SourceFire (SF) and ET signature providers. Furthermore, we consulted with security vendors. In some cases their solutions were based on the same open-source tools, albeit lacking IPv6 support due to small customer demand. Thus, we decided to focus only on evaluating open-source network detection tools. 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.

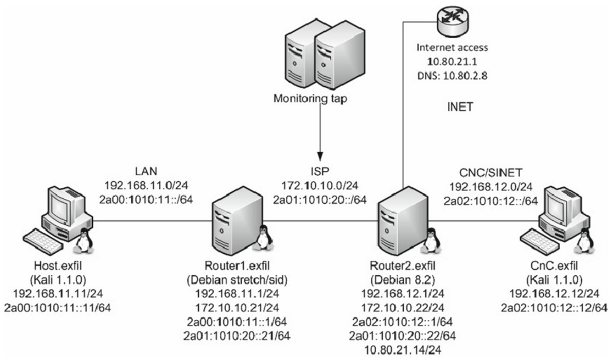

Fig. 1. Testing environment network map

5 Experiment Execution and Discussion of Results

The results of the experiments are presented in an extensive table (see Table 1 in Appendix A). Each row in the table describes a single attack, while the columns represent a detection tool that was used to attempt its detection. In our results, we distinguished four potential outcomes for a test:

1. a positive match (denoted by letter Y and a green cell in the table) 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;

[11] Snort v2.9.8.0. http://manual.snort.org/ (Accessed 07/12/2015).
[12] Suricata v2.1beta4. http://suricata-ids.org/docs/ (Accessed 07/12/2015).
[13] Bro v2.4.1 https://www.bro.org/documentation/index.html (Accessed 07/12/2015).
[14] Moloch v0.12.1. https://github.com/aol/moloch (Accessed 07/12/2015).

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. 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[15]. 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

[15] 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).

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. In most cases, the nc64 tool avoided being detected, and Table 1 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.

6 Conclusions

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 alongside 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 connections/flows. And current NIDS solutions typically lack such capabilities. 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. Another option would be to rely on source NIC MAC address for aggregating and correlating flows from both IPv4 and IPv6 which are originating from 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 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.

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. Finally, commercial tools are often too expensive for small and medium sized organizations. Therefore, we did not consider these products in our final evaluation.

Authors believe, that the tendency of use of IPv6 in attack campaigns conducted by sophisticated malicious actors is going to increase; this is also recognized as an increasing trend by the security reports and articles [4,15,16]. 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.

Acknowledgements. This research was conducted with the support of NATO Cooperative Cyber Defense Center of Excellence. The authors would like to acknowledge the valuable contribution of Leo Trukšāns, Walter Willinger, and Merike Käo.

A Appendix

(See Table 1)

Table 1. 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

(*continued*)

Table 1. (*continued*)

Iteration	IP Version	Protocol	Port	Snort SF	Snort ET	Suricata	Bro	Moloch
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
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

(*continued*)

Table 1. (*continued*)

Iteration	IP Version	Protocol	Port	Snort SF	Snort ET	Suricata	Bro	Moloch
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
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

References

1. Atlasis, A.: Attacking IPv6 implementation using fragmentation. Technical report, Centre for Strategic Cyberspace + Security Science (2011)
2. Atlasis, A.: Security impacts of abusing IPv6 extension headers. Technical report, Centre for Strategic Cyberspace + Security Science (2012)
3. Atlasis, A., Rey, E.: Evasion of high-end IPS devices in the age of IPv6. Technical report, secfu.net (2014)
4. Blumbergs, B.: Technical analysis of advanced threat tactics targeting critical information infrastructure. Cyber Security Review, pp. 25–36 (2014)
5. Blunden, B.: Covert Channels. In: Blunden, B. (ed.) The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System, 2nd edn. Jones and Bartlett Learning, Burlington (2013)
6. Brangetto, P., Çalişkan, E., Rõigas, H.: Cyber Red Teaming - Organisational, technical and legal implications in a military context. NATO CCD CoE (2015)
7. Bukač, V.: IDS system evasion techniques. Master's thesis, Masarykova Univerzita Fakulta Informatiky (2010)
8. Carpenter, B., Jung, C.: Transmission of IPv6 over IPv4 Domains without Explicit Tunnels. RFC 2529, IETF Secretariat, standards Track, March 1999
9. Colajanni, M., Zotto, L.D., Marchetti, M., Messori, M.: Defeating NIDS evasion in Mobile IPv6 networks. In: IEEE (2011)
10. Colitti, L., Gunderson, S.H., Kline, E., Refice, T.: Evaluating IPv6 adoption in the internet. In: Krishnamurthy, A., Plattner, B. (eds.) PAM 2010. LNCS, vol. 6032, pp. 141–150. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12334-4_15
11. Convery, S., Miller, D.: IPv6 and IPv4 Threat Comparison and Best-Practice Evaluation. White paper, Cisco Systems, March 2004
12. Czyz, J., Allman, M., Zhang, J., Iekel-Johnson, S., Osterweil, E., Bailey, M.: Measuring IPv6 adoption. In: ACM SIGCOMM14 (2014)
13. Ellens, W., Żuraniewski, P., Sperotto, A., Schotanus, H., Mandjes, M., Meeuwissen, E.: Flow-based detection of DNS tunnels. In: Doyen, G., Waldburger, M., Čeleda, P., Sperotto, A., Stiller, B. (eds.) AIMS 2013. LNCS, vol. 7943, pp. 124–135. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38998-6_16
14. Farinacci, D., Li, T., Hanks, S., Meyer, D., Traina, P.: Generic Routing Encapsulation (GRE). RFC 2784, IETF Secretariat, March 2000. (standards Track. Supplemented with RFC2890)
15. Fortinet: Biting the Bullet: A Practical Guide for Beginning the Migration to IPv6. white paper, Fortinet Inc. (2011)
16. Data SecurityLabs, G.: Uroburos: Highly complex espionage software with Russian roots. Technical report, G Data Software AG, February 2014
17. Gont, F.: Processing of IPv6 "Atomic" Fragments. RFC 6946, May 2013
18. Gont, F.: Security Implications of IPv6 on IPv4 Networks. RFC 7123, February 2014
19. Gont, F., Chown, T.: Network Reconnaissance in IPv6 Networks. Technical report, IETF Secretariat, February 2015. (internet Draft)
20. Gont, F., Liu, W., Bonica, R.: Transmission and processing of IPv6 options. Technical report, IETF Secretariat, March 2015. (best Current Practice)
21. Gont, F., Heuse, M.: Security assessments of IPv6 networks and firewalls. IPv6 Congress 2013 (2013). (presentation)
22. The Government of HKSAR: IPV6 security. Technical report, The Government of the Hong Kong Special Administrative Region, May 2011

23. Hogg, S., Vyncke, E.: IPv6 Security. Cisco Press, Indianapolis (2009)
24. Krishnan, S.: Handling of Overlapping IPv6 Fragments. RFC 5722, IETF Secretariat, December 2009. (standards Track. Updates RFC 2460)
25. Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., Bhatia, M.: A uniform format for IPv6 extension headers. Technical report
26. Lucena, N.B., Lewandowski, G., Chapin, S.J.: Covert channels in IPv6. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 147–166. Springer, Heidelberg (2006). doi:10.1007/11767831_10
27. Moore, K.: Connection of IPv6 Domains via IPv4 Clouds. RFC 3056, IETF Secretariat, February 2001. (standards Track)
28. Murphy, R.: IPv6 / ICMPv6 Covert Channels. DEF CON 2014 (2014). (presentation)
29. National Cybersecurity and Communications Integration Center: ICS-CERT Monitor. Technical report, US Dep. of Homeland Security, December 2013
30. Niemi, O.P., Levomki, A., Manner, J.: Dismantling intrusion prevention systems. In: ACM SIGCOMM 2012, August 2012
31. Nordmark, E., Gilligan, R.: Basic transition mechanisms for IPv6 hosts and routers. RFC 4213, IETF Secretariat, October 2005. (standards Track)
32. Pastrana, S., Montero-Castillo, J., Orfila, A.: Evading IDSs and firewalls as fundamental sources of information in SIEMS. In: Pastrana, S., Montero-Castillo, J., Orfila, A. (eds.) Advances in Security Information Management: Perceptions and Outcomes. Nova Science Publishers, New York (2013)
33. Ptacek, T.H., Newsham, T.N.: Insertion, evasion, and denial of service: eluding network intrusion detection. Technica report, DTIC Document, January 1998
34. Sarrar, N., Maier, G., Ager, B., Sommer, R., Uhlig, S.: Investigating IPv6 traffic: What happened at the world IPv6 day? In: Taft, N., Ricciato, F. (eds.) PAM 2012. LNCS, vol. 7192, pp. 11–20. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28537-0_2
35. Degen, S., et al.: Testing the security of IPv6 implementations. Technical report, Ministryof Economic Affairs of the Netherlands, March 2014
36. Skoberne, N., Maennel, O., Phillips, I., Bush, R., Zorz, J., Ciglaric, M.: Ipv4 address sharing mechanism classification and tradeoff analysis. IEEE/ACM Trans. Netw. **22**(2), 391–404 (2014)
37. Steffann, S., van Beijnum, I., van Rein, R.: A comparison of IPv6-over-IPv4 tunnel mechanisms. RFC 7059, IETF Secretariat, November 2013. (informational)
38. Tadayoni, R., Henten, A.: Transition from IPv4 to IPv6. In: 23rd European Regional Conference of the International Telecommunication Society, July 2012
39. Taib, A.H.M., Budiarto, R.: Evaluating IPv6 Adoption in the Internet. In: 5th Student Conference on Research and Development. IEEE, December 2007
40. Templin, F., Gleeson, T., Thaler, D.: Intra-site automatic tunnel addressing protocol (ISATAP). RFC 5214, IETF Secretariat, March 2008. (informational)
41. TrendLabs: targeted attack trends 2014 Report. Technical report, TrendMicro (2015)
42. Troan, O., Carpenter, B.: Deprecating the Anycast Prefix for 6to4 Relay Routers. RFC 7526, IETF Secretariat, May 2015. (best Current Practice)
43. Vidal, J.M., Castro, J.D.M., Orozco, A.L.S., Villalba, L.J.G.: Evolutions of evasion techniques aigainst network intrusion detection systems. In: ICIT 2013, The 6th International Conference on Information Technology, May 2013
44. Wu, P., Cui, Y., Wu, J., Liu, J., Metz, C.: Transition from IPv4 to IPv6: a state-of-the-art survey. IEEE Comm. Surv. Tutorials **15**(3), 1407–1424 (2013)