The Impact of IPv6 on Penetration Testing

Christiaan Ottow[1], Frank van Vliet[2], Pieter-Tjerk de Boer[1], Aiko Pras[1]

[1] University of Twente, Enschede, The Netherlands,
chris@6core.net, {p.t.deboer,a.pras}@utwente.nl
[2] Pine Digital Security, The Hague, The Netherlands,
frank.vanvliet@pine.nl

Abstract. In this paper we discuss the impact the use of IPv6 has on remote penetration testing of servers and web applications. Several modifications to the penetration testing process are proposed to accommodate IPv6. Among these modifications are ways of performing fragmentation attacks, host discovery and brute-force protection. We also propose new checks for IPv6-specific vulnerabilities, such as bypassing firewalls using extension headers and reaching internal hosts through available transition mechanisms.
The changes to the penetration testing process proposed in this paper can be used by security companies to make their penetration testing process applicable to IPv6 targets.

Keywords: IPv6, security, penetration testing, host discovery, transition mechanisms

1 Introduction

IPv4 address space is nearing depletion in some regions [2, 25] and adoption of its successor, IPv6, is increasing [17]. At recent security conferences, security researchers have discussed flaws in both the IPv6 protocol and its implementations [3, 10, 14]. Since IPv6 replaces IPv4 as network layer protocol, future network security activities can also be subject to change. Penetration testing is one such activity; companies that provide penetration testing services need to know how to change their process in order to perform penetration tests on IPv6 targets.

In this paper we will answer the following question:

> "How will the use of IPv6 change remote penetration testing of web-applications and servers?"

Penetration testing can be performed in many ways, from physically breaking into structures to social engineering attacks and the hacking of web applications. Although there are some efforts [13, 27, 29] to standardize the process of penetration testing, none of them is commonly accepted as such. We therefore base our discussion on the case study of Pine Digital Security. Pine is a company in the Netherlands that performs remote penetration testing on web applications

and servers. They use public checklists of vulnerabilities, combined with clear defined methods to check for each vulnerability on those checklists. This results in a penetration testing process that can be used in this research to determine the impact IPv6 will have.

The following sub-questions will be used to answer the main question:

- What components of the current penetration testing process need to be changed for IPv6 targets?
- What new components can be added to the penetration testing process for IPv6 targets?

The first sub-question concerns how the current penetration testing process is affected when IPv6 is used: what components need to be modified in order to achieve their goal, or can be removed because they become obsolete. For each component of Pine's penetration testing process, the performed activities are examined to determine how they are affected by a change to IPv6. This is done by analyzing the dependencies of every action performed, to see if functions specific to IPv4 are relied upon. A suggestion for removal from the process or modification is made per activity that is found to be affected. By following this approach, we determine what components of the current penetration testing process are affected, but we do not discover new additions.

The second sub-question concerns what new vulnerabilities IPv6 introduces that are not currently tested for in Pine's penetration testing process. From existing literature and community sources such as protocol standards, conference presentations and weblogs, we collect documented IPv6 vulnerabilities. When these would be applicable within the scope of penetration testing as defined in this research (see Section 2), we propose a check for them as an addition to the process.

In the next section, we discuss our choice of Pine as a case study and Pine's penetration testing process. Components of this process that should be modified for IPv6 or can be removed are discussed in Section 3. Section 4 discusses proposed additions to the process. Finally, conclusions and suggestions for future work are presented in Section 5.

2 Backgrounds on Penetration Testing and the Case Study

In this section we discuss our choice of case study and describe their process of penetration testing for servers and web applications.

Pine Digital Security performs various security services to companies, among which penetration testing. Penetration testing at Pine is described as finding vulnerabilities in servers and web applications using checklists of vulnerability classes, from the perspective of certain types of attackers. Most often this is the perspective of an outsider at a remote location (somewhere on the Internet). Vulnerabilities qualify if they can be used to compromise the confidentiality or integrity of the protected assets. This implies that the penetration test does not

focus on availability, although exceptions are made for availability vulnerabilities that can be exploited without large force, and that can be tested for in a non-destructive manner.

The process of penetration testing at Pine follows the phases described by several sources [29, 32]: planning, execution, post-execution. In the planning phase, apart from commercial activities, an assessment of the customer's systems and risks is made to determine the test targets and scope. Sometimes it is necessary to perform host and service discovery on the target to determine the size of the attack surface. In the execution phase, the penetration tester finds and exploits vulnerabilities in the targets, making sure that every vulnerability on the applicable checklists is checked for. Discovery and attack are executed in cycles, as described in [28]. In the post-execution phase, the results are formulated along with recommendations for mitigation, and are discussed with the client. Cleaning up and quality assurance also take place in this phase.

Pine is representative as a case study of remote penetration testing of web applications and servers. Since Pine uses public [6] checklists and the description of their penetration testing process is available, the impact that IPv6 will have can be determined in a structured way. As described, the phases of Pine's penetration testing process follow a recognized model. The checklists have been assembled by independent security experts with years of penetration testing experience. They contain the most prevalent security issues such as those from the OWASP Top Ten [26] and those exploited by popular automated testing tools, as well as lesser-known issues.

3 Impact of IPv6 on the current penetration testing process

In this section we discuss the components of Pine's current penetration testing process that either need to be changed in order to be applicable to IPv6, or become obsolete and can be removed from the process. We go through the three phases described in the previous section, and discuss the activities that take place. For each of these activities, we analyze its dependencies to see if it relies on features specific to IPv4. We only discuss those items that would need change or can be removed.

3.1 Preparation phase

In the preparation phase, little activity involving the target systems of the penetration test takes place. Normally the customer will give an exact list of IP addresses that form the scope of the penetration test. When the customer however gives a network range instead of an exact list, a scan to detect the number of hosts in a network needs to be performed.

Host and service discovery When performing host discovery in a subnet, all TCP ports and some UDP ports on all hosts are probed. This is done since

many hosts do not respond to various kinds of alive polls (like ping requests), but do offer services on some ports. Alive detection can stop once a proof of aliveness is received from the server. A server without firewall will respond to a TCP connection request to a closed port by sending a TCP packet with the RST flag set, which indicates its aliveness. However, many servers have firewalls that are configured to ignore connection requests to all ports by default, and make exceptions for ports that are on a whitelist. In this case, scanning will have to continue until the first whitelisted port is probed, which will evoke a proof of aliveness from the server in the form of a TCP packet with either the SYN and ACK flags or the RST flag set. Scanning in this case can take very long since the scanner cannot distinguish a lost transmission from a dropped one and has to retry probes multiple times after waiting for a timeout.

This type of scanning is especially problematic in IPv6 networks [7, 31] because of the large IP space typically in use. The standardised netmask size for an IPv6 subnet is 64 bits, leaving 64 bits for numbering hosts [16]. To scan those 64 bits of address space as described would take an infeasible amount of time: 28 years under conditions that favor the scanner [31]. Therefore, exhaustive scanning of all addresses in the network is infeasible within the scope of a penetration test, unless an extremely small prefix has been given.

There are alternatives to exhaustive scanning. Address information can often be retrieved from external sources such as the Domain Name System (DNS) and online directories. Several websites track changes in WHOIS and DNS, and map virtual hosts to IP addresses. Other services index networks by found services. Address information can be taken from these websites. DNS can be used in multiple ways to find addresses. Some servers allow AXFR queries that enable an attacker to download the whole zone. Guessing hostnames using wordlists can also be used. When DNSSEC is used, both NSEC and its successor NSEC3 [20] offer possibilities to enumerate zones [4]. Additionally, the tree of PTR records can be queried to find addresses that are in use when the nameserver returns NOERROR for empty non-terminals. This behavior is RFC-compliant, and implemented by most nameservers [9]. PTR records need to be configured for this enumeration technique to work however, which is not the case in all networks.

Furthermore, assumptions can be made about address assignment policies. Different sources suggest that servers and routers are usually numbered through manual assignment or DHCPv6, not using SLAAC or privacy extensions [10, 14, 21]. In fact, DHCPv6 and manual assignment are even recommended [30]. Research indicates that manual assignment is often based on sequential numbering, numbering with the service port (e.g. assigning an address ending in ::80 to a webserver), wordiness (using hexadecimal characters to form words) and IPv4 addresses in IPv6 addresses. Since the focus of our host discovery is on servers and web applications, manual assignment and DHCPv6 are expected to be prevalent among the targets [7,14]. To find alive hosts, one could assume that these schemes are used for assignment, and guess what other addresses are in use. Based on given or found addresses, address ranges can be found in which

sequential numbering, hex-word, service port and IPv4-address based addresses are in use. For each of the suggestions that this "smart guessing" algorithm generates, alive detection needs to be performed. For a small number of suggestions, traditional alive detection could be performed by scanning all TCP and UDP ports. As the number of suggestions increases, this approach becomes less feasible. A limited number of ports could be probed, alongside generic methods (ICMP ping for instance).

An algorithm using the above methods for "smart guessing" cannot guarantee to find all hosts in a network. If a host has a DNS record that is not easily guessable or no DNS record at all, and has an unpredictable IPv6 address, it may not be found. An early test with such an algorithm reports finding 90-95% of servers on public networks [15]. If a guarantee is required that 100% of the hosts have been included in the test, the only option as of yet is to require an exhaustive list of IP addresses from the customer.

3.2 Execution phase

In the execution phase, the penetration tester attempts to hack the targets and makes sure that each vulnerability on the checklists is checked for. Pine has defined methods for each check, and has documented tools that can be used. These tools are free, commercial or in-house developed. In order to use the tools on IPv6 targets, they need to have basic IPv6 support that includes specifying IPv6 hosts, resolving names to IPv6 addresses and transmitting data over IPv6 sockets. This is a change that affects every tool used, and will not be mentioned separately for every checklist item.

DNS information gathering In two items on the checklists, the Domain Name System (DNS) is queried for (possibly sensitive) information. The goal is not host discovery as was covered in the previous section, but to check if the DNS contains information that should not be stored there, and if the nameservers don't leak the entire zone to attackers. These items are checked by querying the nameservers for the domains in scope, and attempting to download the complete zone (AXFR) or failing that, find certain hosts by guessing hostnames. Reverse hostnames of known IP addresses are resolved as well.

A target that uses IPv6 might have nameservers that are accessible over IPv6, and might have AAAA records in associated zones. The checklist item can remain the same, but the procedure for testing should be changed to include checking both the IPv6 addresses and IPv4 addresses for the nameservers, as they might point to different systems. For all records that are resolved, both the AAAA and A records should be queried. Reverse lookups for both IPv4 and IPv6 addresses should be performed. Wordlists used for brute-force guessing of system names should be updated to contain names relevant to IPv6, such as words containing "ipv6" and the names of popular transition mechanisms such as Teredo and ISATAP.

Fragmentation to bypass firewalls Fragmentation in IPv6 works similar to fragmentation in IPv4. In IPv6 however, fragment information is transmitted in an extension header that is only present when an IP packet is a fragment, and only end nodes perform fragmentation and reassembly.

Fragmentation attacks in IPv4 have been known for years. RFC 1858 [34] describes two attacks that can be used to bypass firewalls using fragmentation: tiny fragments and overlapping fragments. The RFC also suggests mitigations against these attacks.

In the case of tiny fragments, the packet that contains the first step of the TCP handshake (SYN flag) is fragmented across two packets such that the SYN flag, which is at the end of the TCP header, falls within the second fragment. A firewall that tests if a packet constitutes a TCP connection request, will not be able to judge this having just the first fragment. It is possible that no rule will be matched, and the packet let through. The suggested mitigation against this attack is to discard fragments smaller than a minimum size to make sure the TCP flags always fall in the first fragment.

Overlapping fragments can be used to smuggle a connection request past a firewall as well. The first fragment can contain a TCP header with a certain source and destination port but no SYN flag. The firewall may allow this fragment to pass since it is not a connection request. The offset of the second fragment can be tailored to cause reassembly to overwrite the flags field of the TCP header with the data from the second fragment. In this second fragment, a flags field could be set that includes a SYN flag. The firewall might allow the fragment through since it is not the first fragment, while the host reassembling the packet (and parsing the overlap in a favorable way) will read a connection request. The suggested mitigation is to drop fragments with an offset value of 1, since higher offsets do not cause the flags field to be overwritten.

The original IPv6 specification allows overlapping fragments. A later RFC [19] forbids overlapping fragments, but as this is a relatively recent change, not all implementations follow it. As for tiny fragment attacks, many IPv6 implementations accept non-last fragments smaller than 1280 bytes [3]. Additionally, the IPv6 specification dictates that extension headers occurring any number of times in the same packet must be parsed. This requirement can be used to fill a first fragment with extension headers, and put the TCP header in the second fragment. Many implementations already follow a proposed standard [12] the specifies that all headers, including the upper layer header, must be in the first fragment. This mitigates the tiny fragment attack, but is not the standard yet. Other solutions to the tiny fragment problem are being discussed in [22].

When using fragmentation attacks in IPv6, extension headers can be used as described above in addition to the known IPv4 methods.

3.3 Evaluation phase

In the evaluation phase, the report is written and discussed with the customer. This report includes recommendations for mitigation of the found vulnerabilities.

Like the checklist items, these recommendations may have to change to achieve their goal on an IPv6 target.

Automated attacks The recommendation made by Pine against brute-force attacks is to implement an anti-automation measure in the form of rate-limiting or CAPTCHA solving, based on the client's IP address. Several items on the checklist have this recommendation, for instance those that check for brute-forcing user names and passwords or spamming via feedback forms.

This recommendation works in an IPv4 world where an IP address can be assumed to identify a user, since most users have at most one address per person at a point in time. With IPv6 however, a standard network has a 64-bit netmask (a "/64"), and a standard end-user assignment is a multiple of that [24]. An organization is likely to receive a /48, while a home user is more likely to receive a /56. Since March 2011, the exact allocations are left up to the ISP and are no longer defined in an RFC [24]. In any case, an attacker has a very large IPv6 address space compared to the IPv4 situation. Therefore, acting on a single IP address is not enough: the attacker can change IP addresses very often to circumvent anti-automation measures.

We suggest using a "smarter" algorithm for anti-automation measures, that takes into account the large address space an attacker has. This algorithm is based on "taint" actions: actions that are a trigger to the anti-automation system, such as a failed login attempt. After a defined number of taints in a /64, the whole /64 is blocked. Say the attacker has multiple /64 ranges, he can use multiple to begin with, or move to the next after the first has been blocked. Therefore, a threshold is also set for a /56, for a /48, and so on. After each taint, the algorithm checks for all prefix sizes if its taint threshold has been reached. If so, the prefix can be added to a blacklist.

Session hijacking Another recommendation that needs to change, is the recommendation made for web applications to allow the use of a session from only one IP address. Pine recommends that when a session is started (a session identifier is generated and sent to the user), the IP address of the user is stored on the server along with the session. Upon each subsequent request with that session identifier, the server checks if the IP address where the request came from is the same as the stored IP address. If not, access to the session is denied.

With IPv6, privacy extensions [23] are often used for end-user systems [18]. The addresses that are generated have a limited preferred lifetime, which means they are used for outgoing connections for a limited time of often only hours. When the preferred lifetime is over, a new address is generated and the address is not used for outgoing connections anymore.

The described behavior poses a problem for session management according to Pine's recommendation. A user may use different addresses for his requests to the web application during his session, since sessions typically run for longer than a few hours. Therefore, the IP address cannot be used to lock the user's session anymore. The /64 prefix in which the random addresses are generated,

remains constant in the address, but may be used by many more users. A trade-off has to be made between usability and security: if the current recommendation is applied to IPv6, users with privacy addresses will have trouble using the web application. If the /64 prefix is used, abuse of the session by remote attackers is prevented, but attackers on the same LAN may still hijack the session. With IPv4 this situation often already exists, since many networks use NAT, which results in a network segment having one external address.

We recommend tracking the session using the /64 prefix. The measure is for defense-in-depth: stealing of the session identifier should be made impossible by other means such as proper output escaping to prevent Cross-site Scripting (XSS), the use of SSL and setting proper flags on session cookies. Additionally, in IPv4, many users share the same public IP address with the whole LAN already, so the use of the IPv6 /64 does not offer less protection.

4 Additions to the penetration testing process

In this section we describe additions that should be made to the penetration testing process, based on existing literature, protocol standards and community sources. Proposed additions need to be within our scope of penetration testing. As such, vulnerabilities that are specific to the LAN (such as Neighbor Discovery and Multicast Listener Discovery vulnerabilities) are not considered, since they cannot be abused from a remote perspective. Vulnerabilities that can only be used for denial of service attacks and cannot be tested for in a non-destructive manner are out of scope as well.

4.1 Using extension headers to bypass router ACLs

Extension headers in IPv6 packets are placed between the fixed-size IPv6 header and the payload. The IPv6 standard dictates that any number of extension headers must be processed by receiving nodes.

Most routers can enforce packet filtering policies based on Access Control Lists (ACLs). Filters are usually composed of not only source and destination IP address, but also TCP/UDP ports and flags. To read this information, the router needs to read past the complete IP header in the packet, including extension headers. High-end routers achieve high throughput by implementing the forwarding plane (where ACL checking takes place) in hardware instead of in software. This hardware has a fixed-size view on the packet: often only the first 64, 128 or 256 bytes are evaluated to make a routing or policy decision [33].

This limited packet view can introduce a security problem. A packet can have so many extension headers that the upper layer header is moved outside of the view of the forwarding plane, meaning that the information required for a policy decision is not available. Some routers choose to pass such a packet on to the control plane, which is software-based and can handle dynamic lengths. Others just let the packet pass [33], which provides an easy way to bypass policy enforcement.

An addition should be made to the checklists for checking whether adding a large number of extension headers leads to bypasses ACL restrictions as described above.

4.2 Unintended exposure due to transition mechanisms

Various mechanisms have been defined to ease the transition from IPv4 to IPv6. Some of these transition mechanisms provide automatic tunneling for hosts that want to use IPv6 in an IPv4-only network. These mechanisms are either based on existing IPv4 tunneling techniques such as IP-in-IP and GRE or encapsulating IPv6 packets inside layer 4 protocols such as UDP.

Hosts that are not reachable by a global IPv4 address, might become globally reachable over IPv6 by the use of these transition mechanisms. This poses a security risk [8, 11]. An addition should be made to the checklists to check for the reachability of hosts in the target network using IPv6 transition mechanisms. This could be achieved by performing alive detection on guessed addresses, based on assumptions about the transition mechanism used. Assumptions about the addresses can be made when ISATAP, Teredo or 6to4 are in use, since these transition mechanisms use predictable information in the IPv6 addresses that are generated.

ISATAP ISATAP is used to provide hosts in a network with IPv6 support, while not migrating the entire network to IPv6. The network itself must have an IPv6 prefix routed to it, but not all infrastructure inside the network needs to support IPv6. The IPv4 network is used as a link layer for IPv6. Hosts in the network use the IPv6 prefix combined with their own IPv4 address *v4addr* to generate an IPv6 address, that ends with the 48 bits :5efe:*v4addr*.

If the network prefix is known, for instance from other IPv6 enabled hosts or a WHOIS service, the generated IPv6 addresses can be guessed based on the IPv4 addresses that are in use. Private IPv4 addresses can even be accessed using this method: if a host has IP address 10.0.0.2 on the LAN and it uses ISATAP in prefix 2001:db8::/32, it will have IPv6 address 2001:db8::5efe:a00:2 (0a00:0002 is 10.0.0.2 in hexadecimal notation) which is globally reachable by default.

6to4 A host that has IPv4 address *v4addr* can have the prefix 2002:*v4addr*::/48 routed to it by setting up a 6to4 tunnel. By substituting known IPv4 addresses of the target into 2002:*v4addr*::/48, prefixes are acquired wherein host discovery can be performed.

Teredo Teredo addresses contain more variables than just the *v4addr*, such as the UDP port number. By making assumptions about these variables, addresses could be guessed. However, since there are 2^{16} possible port numbers, one would need to test $\frac{2^{16}}{2}n$ guesses on average, where n is the number of IPv4 addresses in use. This will very quickly become infeasible.

An addition to the penetration testing process should be made to discover hosts reachable via the 6to4 and ISATAP transition mechanisms. Within ISATAP prefixes, known IPv4 addresses and commonly used private ranges should be tried. Within 6to4, the IPv4 addresses of border routers in the target network should be used to generate prefixes in which host discovery can be performed.

4.3 Evading policy enforcement using routing headers

One of the IPv6 extension headers is the Routing Header (RH). There are two types of routing headers, of which RH0 (type 0) can be used to specify a list of addresses that should be visited by the packet.

This feature offers the same functionality as the "Loose Source Route Record (LSRR)" option in IPv4, and poses the same security risk. With RH0 enabled, an attacker could craft a packet that follows a certain path into a network, bypassing a firewall. Or, the destination according to the destination address could be allowed by the firewall, while the real final destination specified inside the RH0 header is not [1, 5]. The first destination host would then forward the packet over the internal network.

Because of the vulnerabilities identified in the RH0 functionality, the IPv6 standard was updated in 2007 [1] to deprecate RH0 extension headers. However, many IPv6 implementations already contained the RH0 functionality, and there is no guarantee that all of them have been updated to reflect the change. Therefore an addition to the penetration testing process should be made to check for firewall bypassing using RH0 headers.

5 Conclusions and Future Work

This paper shows that when IPv6 is used, changes to the penetration testing process are needed. Some activities need to be performed in a different way for IPv6 targets, and some new checks should be added to the process. None of the activities of the current penetration testing process need to be removed from the process.

The following elements of the process need to be changed. Host discovery by exhaustively scanning the entire IP space will not work for normal-sized IPv6 networks. We have described an algorithm that combines different sources of address information with assumptions about address assignment, to replace exhaustive scanning. DNS information gathering should be changed to also search for IPv6 (AAAA) records, and include IPv6-specific wordlists. Fragmentation attacks that aim to bypass firewalls should be changed to include IPv6-specific fragmentation attacks. Recommendations for mitigation of vulnerabilities are also affected: brute-force attacks cannot be stopped by blocking single IPv6 addresses. We described an algorithm that takes into account the address space an attacker may have. Furthermore, users cannot be expected to keep the same IPv6 address throughout the lifetime of their session, so the user's network prefix must be taken into account when restricting access to the session.

Additions to the penetration testing process should be made to target several IPv6-specific vulnerabilities. The use of IPv6 extension headers in general allows for router ACL bypassing on some systems. If Routing Headers are allowed, firewalls might be bypassed as well. When certain transition mechanisms are in use, internal systems may be directly accessible to attackers. Additions to the checklists are proposed for these IPv6-specific vulnerabilities.

Using Pine's penetration testing process provided us with a structured way to examine the impact IPv6 may have, but may not be applicable to penetration testing as performed by others than Pine. As future work, the impact of IPv6 on other forms of penetration testing, such as Local Area Network penetration testing, could be investigated. Future research could also verify our findings by performing penetration tests on IPv6 targets with the checks and algorithms proposed here.

Acknowledgements The authors would like to thank Peter van Dijk, Frank Kargl and Job Snijders for their valuable feedback and suggestions, and Pine Digital Security for their support of this work.

References

1. Abley, J., Savola, P., Neville-Neil, G.: Deprecation of type 0 routing headers in IPv6. http://tools.ietf.org/html/rfc5095 (Dec 2007)
2. APNIC: APNIC IPv4 address pool reaches final /8. http://www.apnic.net/publications/news/2011/final-8 (Apr 2011)
3. Atlasis, A.: Attacking ipv6 implementation using fragmentation. http://media.blackhat.com/bh-eu-12/Atlasis/bh-eu-12-Atlasis-Attacking_IPv6-WP.pdf (Mar 2012)
4. Bernstein, D.: Breaking dnssec. http://cr.yp.to/talks/2009.08.10/slides.pdf (2009 Aug)
5. Biondi, P., Ebalard, A.: IPv6 routing header security. http://cansecwest.com/csw07/csw07-ebalard-biondi.pdf (Apr 2007)
6. Certified Secure: Certified Secure Checklists. https://www.certifiedsecure.com/checklists/
7. Chown, T.: RFC 5157: IPv6 implications for network scanning. http://www.rfc-editor.org/rfc/rfc5157.txt (Mar 2008)
8. Davies, E., Krishnan, S., Savola, P.: IPv6 transition/coexistence security considerations. http://tools.ietf.org/html/rfc4942 (Sept 2007)
9. van Dijk, P.: Finding v6 hosts by efficiently mapping ip6.arpa. http://7bits.nl/blog/2012/03/26/finding-v6-hosts-by-efficiently-mapping-ip6-arpa (Mar 2012)
10. Gont, F.: Results of a security assessment of the internet protocol version 6. http://www.si6networks.com/presentations/hacklu2011/fgont-hacklu2011-ipv6-security.pdf (Sept 2011)
11. Gont, F.: Security implications of ipv6 on ipv4 networks. http://www.ietf.org/id/draft-gont-opsec-ipv6-implications-on-ipv4-nets-00.txt (Apr 2012)
12. Gont, F., Manral, V.: Security and interoperability implications of oversized ipv6 header chains.

http://tools.ietf.org/html/gont-6man-oversized-header-chain-01 (Apr 2012)
13. Herzog, P.: The Open Source Security Testing Methodology Manual. ISECOM (2010)
14. Heuse, M.: Recent advances in IPv6 insecurities. http://events.ccc.de/congress/2010/Fahrplan/events/3957.en.html (Dec 2010)
15. Heuse, M.: Vulnerabilities, failures - and a future? http://www.mh-sec.de/downloads/mh-ipv6_vulnerabilities.pdf (Nov 2011)
16. Hinden, R., Deering, S.: RFC 4291: IP version 6 addressing architecture. http://tools.ietf.org/html/rfc4291 (Feb 2006)
17. Huston, G.: Active BGP entries (FIB). http://bgp.potaroo.net/v6/as2.0/index.html
18. Kaps, R.: Ipv6: Privacy extensions einschalten. http://www.heise.de/netze/artikel/IPv6-Privacy-Extensions-einschalten-1204783.html (Mar 2011)
19. Krishnan, S.: RFC 5722 - Handling of overlapping IPv6 fragments. http://tools.ietf.org/html/rfc5722 (Dec 2009)
20. Laurie, B., Sisson, G., Arends, R., Blacka, D.: RFC 5155: DNS security (DNSSEC) hashed authenticated denial of existence. http://tools.ietf.org/html/rfc5155 (Mar 2008)
21. Malone, D.: Observations of IPv6 addresses. In: PAM'08 Proceedings of the 9th international conference on Passive and active network measurement. Springer-Verlag Berlin (2008)
22. Manral, V.: Tiny fragments in ipv6. http://tools.ietf.org/html/draft-manral-6man-tiny-fragments-issues-00 (Feb 2012)
23. Narten, T., Draves, R., Krishnan, S.: RFC 4941: Privacy extensions for stateless address autoconfiguration in IPv6. http://tools.ietf.org/html/rfc4941 (Sept 2007)
24. Narten, T., Huston, G., Roberts, L.: RFC 6177 - IPv6 address assignments to end sites. http://tools.ietf.org/html/rfc6177 (Mar 2011)
25. NCC, R.: IPv4 exhaustion. http://www.ripe.net/internet-coordination/ipv4-exhaustion (2012)
26. OWASP: OWASP top ten. https://www.owasp.org/index.php/Top_10_2010 (2010)
27. PTES: The Penetration Testing Execution Standard. http://www.pentest-standard.org/ (2012)
28. Saindane, M.S.: Penetration testing – a systematic approach. Tech. rep., infosecwriters.com (2006)
29. Scarfone, K., Souppaya, M., Cody, A., Orebaugh, A.: Technical guide to information security testing and assessment. Tech. rep., NIST (2008)
30. SURFnet: IPv6 numberplan. http://www.surfnet.nl/nl/nieuws/Pages/HandleidingIPv6-nummerplanverschenen.aspx (Feb 2011)
31. Vyncke, E.: IPv6 Security. Cisco Press (2009)
32. Wai, C.T.: Conducting a penetration test on an organization. http://www.sans.org/reading_room/whitepapers/auditing/conducting-penetration-test-organization_67 (2002)
33. Ytti, S.: IPv6 ACL bypass. http://blog.ip.fi/2011/08/ipv6-acl-bypass.html (Aug 2011)
34. Ziemba, G., Reed, D., Traina, P.: RFC 1858 - security considerations for IP fragment filtering. http://tools.ietf.org/html/rfc1858 (Oct 1995)