ERNW Newsletter 45 / December 2014

Penetration Testing Tools that (do not) Support IPv6

ERNW Enno Rey Netzwerke GmbH
Carl-Bosch-Str. 4

69115 Heidelberg

Tel. +49 6221 480390
Fax +49 6221 419008
www.ernw.de

Version: 1.1

Date: 12/11/2014

Author(s): Dr. Antonios Atlasis

ERNW
providing security.

Table of Content

1 Introduction 7
1.1 Lab Setup 7
1.2 Families of Penetration Testing Tools 8
2 Information Gathering 9
2.1 Robtex 9
2.1.1 Conclusion 9
2.2 Shodan 10
2.2.1 Conclusion 10
2.3 Maltego 11
2.3.1 Conclusion 11
2.4 Dradis 11
2.4.1 Conclusion 12
3 DNS Enumeration 13
3.1 Fierce 13
3.1.1 Conclusion 13
3.2 DNSrecon 13
3.2.1 Conclusion 15
3.3 Traceroute - Tcptraceroute 15
3.3.1 Conclusion 15
3.4 Traceroute6 / Traceroute 15
3.4.1 Conclusion 15
3.5 Firewalk 16
3.5.1 Conclusion 16
4 Network / Port Scanning 17
4.1 Unicornscan 17
4.1.1 Conclusion 17
4.2 Nmap 17
4.2.1 Conclusion 18
5 IPv6 fingerprinting 19
5.1 Nmap 19
5.1.1 Conclusion 19
5.2 Xprobe2 19
5.2.1 Conclusion 20
5.3 p0f 20
5.3.1 Conclusion 20
5.4 Amap 20
5.4.1 Conclusion 21

6 BRUTE-FORCING 22
6.1 HYDRA 22
6.1.1 Conclusion 23
6.2 MEDUSA 23
6.2.1 Conclusion 23
6.3 NCRACK 23
6.3.1 Conclusion 23
7 PACKET CRAFTING 24
7.1 HPING 24
7.1.1 Conclusion 24
7.2 NPING 24
7.2.1 Conclusion 25
7.3 SCAPY 25
7.3.1 Conclusion 25
8 REMOTE SHELLS 26
8.1 NCAT 26
8.1.1 Conclusion 27
9 LAN / MITM ATTACKS & OTHER 28
9.1 NSE SCRIPTS 28
9.1.1 Conclusion 33
9.2 ETTERCAP 34
9.2.1 Conclusion 35
9.3 CAIN & ABEL 35
9.3.1 Conclusion 35
9.4 NET-SNMP 35
9.4.1 Conclusion 35
10 VULNERABILTY SCANNERS 36
10.1 NESSUS 36
10.1.1 Conclusion 40
11 WEB PENETRATION TESTING 41
11.1 BURPSUITE 41
11.1.1 Conclusion 41
11.2 ZAPROXY 41
11.2.1 Conclusion 42
11.3 NIKTO 42
11.3.1 Conclusion 43
11.4 SKIPFISH 43
11.4.1 Conclusion 43
11.5 SQLMAP 43
11.5.1 Conclusion 44

11.6 SQLNINJA 44
11.6.1 Conclusion 46
11.7 W3AF 46
11.7.1 Conclusion 46
11.8 ARACHNI 46
11.8.1 Conclusion 47
12 EXPLOITATION FRAMEWORKS 48
12.1 METASPLOIT 48
12.1.1 Conclusion 51
13 WHEN OUR FAVORITE HACKING TOOL DOES NOT SUPPORT IPV6 52
13.1 FAST AND EASY 52
13.2 EXPLOITING IPV6 FEATURES WITH YOUR IPV4 TOOLS 53
14 CONCLUSION 55
15 APPENDIX 56
15.1 LIST OF THE TESTED TOOLS 56
15.2 METASPLOIT MODULES 60
15.3 REFERENCES 61
15.4 DISCLAIMER 61

ERNW
providing security.

List of Figures

Figure 1 Robtex Output "ernw.de" 9
Figure 2 IPv6 Search Results Shodan 10
Figure 3 Maltego Target www.ernw.de 11
Figure 4 Dradis Nmap Output 11
Figure 5 Ncat Unencrypted Communication – Example 1 26
Figure 6 Ncat Unencrypted Communication – Example 2 27
Figure 7 RFC 4620 30
Figure 8 Wireshark Output nmap scan 31
Figure 9 Flood router impact on a Windows 7 Operating System 31
Figure 10 Wireshark Output targets-ipv6-multicast-invalid-dst 32
Figure 11 Targets-ipv6-multicast-mld 33
Figure 12 IPv6 Hosts List Ettercap 34
Figure 13 Wireshark Output SNMP Walk 35
Figure 14 Nessus IPv6 Targets 36
Figure 15 Wireshark Output ICMPv6 Echo Request 37
Figure 16 Nessus IPv6 Host Discovery 37
Figure 17 Detailed Nessus Output IPv6 Host Discovery 38
Figure 18 Nessus Invalid Target Settings 38
Figure 19 Advanced Search Nessus 39
Figure 20 Nessus Plugins 39
Figure 21 Nessus Enumeration IPv6 Interfaces via SSH 39
Figure 22 Burp Suite IPv6 Targets 41
Figure 23 Zaproxy IPv6 URL To Attack 42
Figure 24 Zaproxy Invalid Address Proxy Error 42
Figure 25 Wireshark Output SqlMap 44
Figure 26 w3af Invalid URL Error 46
Figure 27 ArachniIPv6 URL 47
Figure 28 Arachni Issues 47
Figure 29 Wireshark Output Multicast Ping 49
Figure 30 Wireshark Output IPv6 Neighbor Router Advertisement 51
Figure 31 Chiron Workflow 53
Figure 32 IP6 Tables Firewall Configuration 54

TABLE DIRECTORY

Table 1 List of Tested Tools 59

1 INTRODUCTION

The goal of this study was to:

- Find out which of our favorite penetration testing tools can be used natively using IPv6 as an underlying layer-3 protocol.
- Find alternative solutions for the rest.

During our tests, only open-source or free versions of commercial tools were considered. Moreover, extensive (e.g. non network functionality-related) testing was not performed (it was out of the scope of this study); instead, just the extent of IPv6 support for each one of them was examined. Finally, by no means this is (or can be) an exhaustive list of penetration testing tools.

Out of the scope of this document were also IPv6-specific tools or frameworks, such as:

- THC-IPV6[1]
- SI6 Networks' IPv6 Toolkit[2]
- Chiron[3]

During our tests, the following methodology was used:

- The tools were tested in a virtual IPv6 lab, using various operating systems as targets and using a Linux box as a router. The reason for doing so is because we did not want to mess with real targets. However, in some cases (DNS resolving, trace routing), real IPv6 "targets" were tested.
- The tools were tested just about whether they can operate over IPv6 natively and not about their effectiveness regarding penetration testing functionalities.
- Most tools were installed by using the latest available source code (since packages provided by several Linux distributions may not be the latest one). However, there are a very few exceptions to this rule (which are noted, per case).
- The focus was on tools known from IPv4 and not on similar features covered from IPv6-specific tools.

1.1 Lab Setup

During the tests, the following targets were used in a virtual environment deployed by using VirtualBox (for reasons of completeness, the corresponding IPv6 addresses are also displayed):

Operating System	IPv6 Address
Fedora 20	fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
Centos 6.5	fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a
OpenBSD 5.5	fdf3:f0c0:2567:7fe4:a00:27ff:fe6a:ca6a
FreeBSD 10	fdf3:f0c0:2567:7fe4:a00:27ff:fefc:f99a
Windows 7.1	fdf3:f0c0:2567:7fe4:c0c6:5389:f6e:99c0

[1] *https://www.thc.org/thc-ipv6/*
[2] *http://www.si6networks.com/tools/ipv6toolkit/*
[3] *http://www.secfu.net/tools-scripts/*

Kali Linux 1.0.8	fdf3:f0c0:2567:7fe4:a00:27ff:fedd:77f4

Fedora 20 host also served as an IPv6 router by using radvd[4].

When required (as for example in case where DNS resolving or tracerouting were tested) real IPv6 addresses/"targets" were used.

1.2 Families of Penetration Testing Tools

During this study, tools from the following families were tested:

- Information Collaboration
- Reconnaissance – port scanning
- Nmap IPv6-specific scripts
- Fingerprinting
- Brute-Forcing
- Remote Shells
- Packet Crafting
- Vulnerability Scanning
- LAN attacks / MITM
- Web Penetration Testing
- Exploitation frameworks

[4] *http://www.litech.org/radvd/*

2 INFORMATION GATHERING

2.1 Robtex

Let's start by trying to check if Robtex[5] provides also IPv6-related information. For instance, by checking for ernw.de we get the results displayed in the next figure. As we can easily observe, IPv6 addresses are also included in the findings (circled in red)

Base	Record	Preference	Name	IP Number	Reverse	Routes	AS	Location
WWW.ERNW.DE	A		WWW.ERNW.DE	2003:60:4010:1090::11		2003::/19 2003::/23 DTAG European region optimized	AS3320 DTAG Deutsche Telekom AG	
				62.159.96.68	NG.ERNW.DE WWW.ERNW.DE	62.156.0.0/14 Deutsche Telekom AG, Internet service provider ERNW-NET TSBS GmbH fuer Oberberg OnlineInformationssysteme GmbH		Germany
ERNW.DE			ERNW.DE	2003:60:4010:1090::13	ERNW.DE TROOPERS.DE TROOPERS.NET WWW.TROOPERS.NET	2003::/19 2003::/23 DTAG European region optimized		
				62.159.96.70		62.156.0.0/14 Deutsche Telekom AG, Internet service provider ERNW-NET TSBS GmbH fuer Oberberg OnlineInformationssysteme GmbH		Germany
	NS (missing in zone)		NS1.ERNW.DE	62.159.96.78	MX1.ERNW.NET			
			NS2.ERNW.DE	212.102.247.186	MX2.ERNW.NET	212.102.224.0/19 Oberberg-Online Informationssysteme GmbH DE-OBERBERGONLINE-20000530 Oberberg-Online Informationssysteme GmbH Provider Local Internet Registry	AS15415 OBIS Oberberg-Online Informationssysteme	Oberberg, Germany
	MX	10	MX1.ERNW.NET	2003:60:4010:10A0::11		2003::/19 2003::/23 DTAG European region optimized	AS3320 DTAG Deutsche Telekom AG	
				62.159.96.78	MX1.ERNW.NET	62.156.0.0/14 Deutsche Telekom AG, Internet service provider ERNW-NET TSBS GmbH fuer Oberberg OnlineInformationssysteme GmbH		Germany
		15	MX2.ERNW.NET	212.102.247.186	MX2.ERNW.NET	212.102.224.0/19 Oberberg-Online Informationssysteme GmbH DE-OBERBERGONLINE-20000530 Oberberg-Online Informationssysteme GmbH Provider Local Internet Registry	AS15415 OBIS Oberberg-Online Informationssysteme	Oberberg, Germany
	NS (primary, but missing in delegation)		NS1.ERNW.NET	62.159.96.78	MX1.ERNW.NET	62.156.0.0/14 Deutsche Telekom AG, Internet service provider ERNW-NET TSBS GmbH fuer Oberberg OnlineInformationssysteme GmbH	AS3320 DTAG Deutsche Telekom AG	Germany
	NS					212.102.224.0/19 Oberberg-Online Informationssysteme GmbH		

Figure 1 Robtex Output "ernw.de"

2.1.1 Conclusion

Robtext can be used (as seen in the figure above) for IPv6 reconnaissance purposes.

[5] *https://www.robtex.com*

2.2 Shodan

Now, let's continue with Shodan[6] for searching about IPv6-related findings. An example output is displayed below:

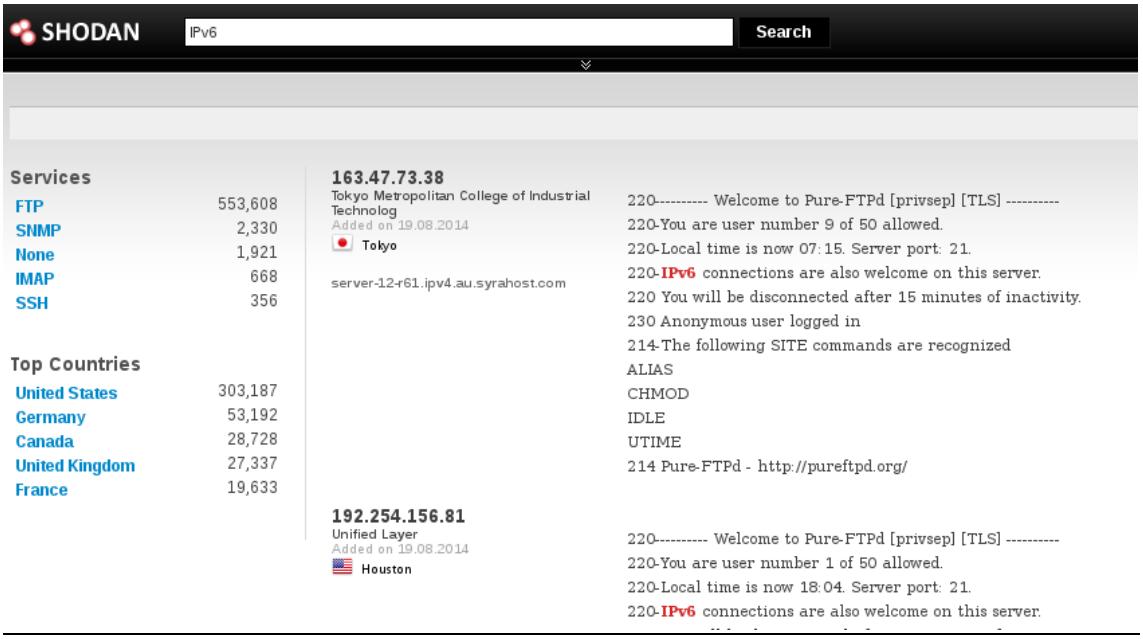

Figure 2 IPv6 Search Results Shodan

As we can see, there are IPV6-related findings. However, they must be examined and analyzed carefully. For instance, the displayed choice is a FTP server which, according to its banner, welcomes also IPv6 connections, but when SSH servers were chosen, ftp servers were also displayed, just because they were configured to use port 22.

2.2.1 Conclusion

So, it seems that some information can be obtained regarding IPv6 from Shodan, but this info does not seem to be extracted in a very reliable and sophisticated way. Certainly, digging further is required.

[6] *http://www.shodanhq.com*

2.3 Maltego

For Maltego, an open source intelligence and forensics application, we used the community edition version 3.4.0. Again, let's search for our favorite "target", *www.ernw.de*. A sample output is displayed below:

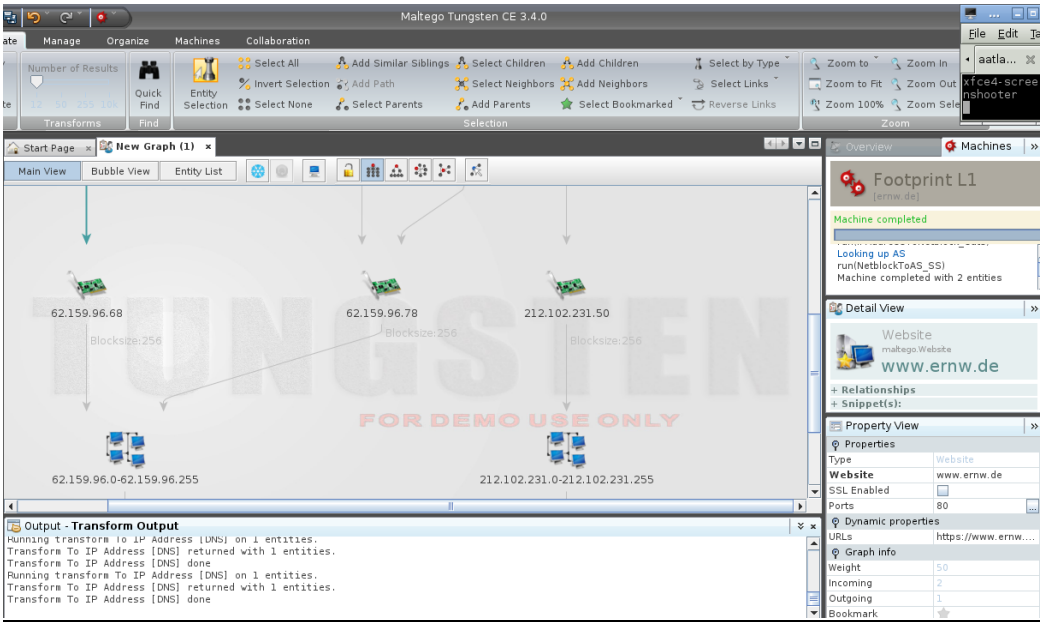

Figure 3 Maltego Target www.ernw.de

As we can see, although we used the provided transformations for obtaining IP addresses, these results are just IPv4 addresses. We don't know if the commercial version offers some IPv6-specific transformations, but, at least the community one seems that does not offer such.

2.3.1 Conclusion

So, as far as IPv6 is concerned, it seems that Maltego is not an option.

2.4 Dradis

Dradis is an open source framework to enable effective information sharing, especially during security assessments. The tested version was 2.9.0, running on Kali Linux 1.0.8.

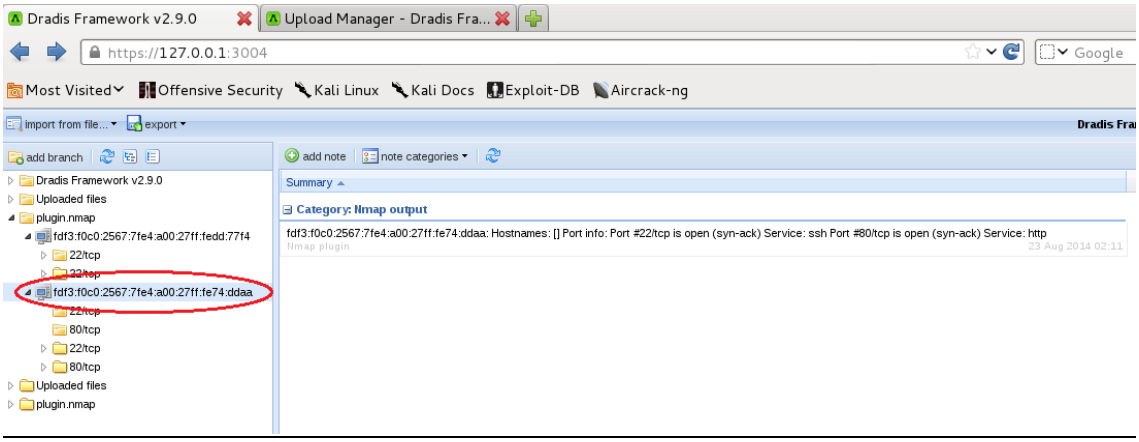

Figure 4 Dradis Nmap Output

As we can see, we can import IPv6 addresses e.g. from nmap outputs for further collaboration. The same applies for burpsuite.

2.4.1 Conclusion

Dradis can be used for information collaboration in the IPv6 era.

3 DNS ENUMERATION

3.1 Fierce

Regarding DNS enumeration, let's start with the "old" but classic fierce. A sample output search for ernw.de is displayed below:

$ perl fierce.pl -dns ernw.de

DNS Servers for ernw.de:

ns2.ernw.net

ns1.ernw.net

Trying zone transfer first...

Testing ns2.ernw.net

Request timed out or transfer not allowed.

Testing ns1.ernw.net

Request timed out or transfer not allowed.

Unsuccessful in zone transfer (it was worth a shot)

Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...

Nope. Good.

Now performing 2898 test(s)...

192.168.99.35 cms.ernw.de

172.31.13.10 crm.ernw.de

172.31.1.10 lists.ernw.de

3.1.1 Conclusion

Based on the above output and taking into account that there are IPv6 addresses which are not displayed, we infer that IPv6 is not supported by fierce. Not surprising, since fierce is a quite "old" DNS recon script.

3.2 DNSrecon

Now, let's try the same using DNSrecon, which is maintained and updated regularly. For the testing purposes, we cloned it from github. For using it, *python-netaddr.noarch* is required.

According to the readme file, since ##Version 0.6.1, ###Date: 1/14/12:

- IPv6 support for ranges in reverse look-up.
- Enhanced parsing of SPF records ranges to cover includes and IPv6.

Let's use it now, "against" ernw.de:

./dnsrecon.py -d ernw.de --lifetime 15

[*] Performing General Enumeration of Domain: ernw.de

[-] DNSSEC is not configured for ernw.de

[*] SOA ns1.ernw.net 62.159.96.78

[*] NS ns1.ernw.net 62.159.96.78

[-] Recursion enabled on NS Server 62.159.96.78

[*] Bind Version for 62.159.96.78 9.6-ESV-R4

[*] NS ns2.ernw.net 212.102.247.186

[-] Recursion enabled on NS Server 212.102.247.186

[*] Bind Version for 212.102.247.186 9.6-ESV-R4

[*] MX mx1.ernw.net 62.159.96.78

[*] MX mx2.ernw.net 212.102.247.186

[*] MX mx1.ernw.net 2003:60:4010:10a0::11

[*] A ernw.de 62.159.96.70

[*] AAAA ernw.de 2003:60:4010:1090::13

[*] TXT ernw.de v=spf1 a mx ptr include:ernw.net -all

[*] Enumerating SRV Records

[-] No SRV Records Found for ernw.de

[*] 0 Records Found

And digging further:

$ ./dnsrecon.py -r 2003:60:4010:1090::8-2003:60:4010:1090::13

[*] Reverse Look-up of a Range

[*] Performing Reverse Lookup from 2003:60:4010:1090::8 to 2003:60:4010:1090::13

[*] PTR www.ernw.de 2003:60:4010:1090::11

[*] PTR www.troopers.de 2003:60:4010:1090::12

[*] 2 Records Found

./dnsrecon.py -r 2003:60:4010:1090::0/120

[*] Reverse Look-up of a Range

[*] Performing Reverse Lookup from 2003:60:4010:1090:: to 2003:60:4010:1090::ff

[*] PTR www.troopers.de 2003:60:4010:1090::12

[*] PTR www.ng.troopers.de 2003:60:4010:1090::30

[*] PTR www.ng.insinuator.net 2003:60:4010:1090::30

[*] PTR www.ng.ernw.de 2003:60:4010:1090::30

[*] 4 Records Found

3.2.1 Conclusion

So, we can put DNSrecon in our quiver with the IPv6 penetration testing tools.

3.3 Tracerouting - Tcptraceroute

Used system package (since it is a very basic one), there is no information regarding the version.

According to man pages:

-6	Explicitly force IPv6 tracerouting.

Test:

tcptraceroute -6 www.google.com
getopt: invalid option – '6'

NOTE: You get the same error if you use an IPv6 address instead of a hostname.

3.3.1 Conclusion

Although IPv6 is advertised, it doesn't seem to work.

3.4 Traceroute6 / Traceroute

traceroute6 www.google.com
traceroute to www.google.com (2a00:1450:4017:800::1012), 30 hops max, 80 byte packets
1 2a02:2149:810b:7200:20d:b9ff:fe28:c214 (2a02:2149:810b:7200:20d:b9ff:fe28:c214) 0.491 ms 0.430 ms 0.588 ms
2 bbras-llu-kln-15L500.forthnet.gr (2a02:2148:77:50:2::5) 36.209 ms 39.093 ms 39.939 ms
3 te0-1-0-0.distr-kln-01.forthnet.gr (2a02:2148:2:32::22) 41.372 ms 42.096 ms 43.735 ms
4 BE-2.core-kln-12.forthnet.gr (2a02:2148:2:9::11) 47.616 ms 48.299 ms 49.022 ms
5 2a02:2148:2:6::22 (2a02:2148:2:6::22) 56.300 ms 61.197 ms 61.938 ms
6 2001:4860:1:1:0:4d9:0:1 (2001:4860:1:1:0:4d9:0:1) 68.665 ms 50.945 ms 51.564 ms
7 2001:4860:0:1::617 (2001:4860:0:1::617) 53.609 ms 50.496 ms 51.813 ms
8 2a00:1450:8000:24::3 (2a00:1450:8000:24::3) 52.440 ms 2a00:1450:4017:800::e (2a00:1450:4017:800::e) 50.945 ms
2a00:1450:8000:24::5 (2a00:1450:8000:24::5) 55.465 ms

traceroute6 --tcp -p 80 www.google.com
traceroute to www.google.com (2a00:1450:4017:800::1014), 30 hops max, 80 byte packets
1 2a02:2149:8601:1c00:20d:b9ff:fe28:c214 (2a02:2149:8601:1c00:20d:b9ff:fe28:c214) 0.492 ms 0.428 ms 0.586 ms
2 bbras-llu-kln-12L500.forthnet.gr (2a02:2148:77:50:2::17) 40.596 ms 40.582 ms 41.146 ms
3 te0-1-0-5.distr-kln-01.forthnet.gr (2a02:2148:2:65::22) 42.024 ms 42.799 ms 43.429 ms
4 BE-2.core-kln-12.forthnet.gr (2a02:2148:2:9::11) 48.154 ms 49.484 ms 51.188 ms
5 2a02:2148:2:6::22 (2a02:2148:2:6::22) 56.648 ms 59.024 ms 60.485 ms
6 2001:4860:1:1:0:4d9:0:1 (2001:4860:1:1:0:4d9:0:1) 69.821 ms 51.040 ms 51.654 ms
7 2001:4860:0:1::617 (2001:4860:0:1::617) 50.795 ms 46.540 ms 51.746 ms
8 sof01s01-in-x14.1e100.net (2a00:1450:4017:800::1014) 49.381 ms 46.101 ms 48.100 ms

3.4.1 Conclusion

IPv6 is supported. No advanced features though (i.e. support of IPv6 Extension headers).

3.5 Firewalk

Firewalk[7] is an active reconnaissance network security tool that attempts to determine what layer 4 protocols a given IP forwarding device will pass. Tested was version 5.

Test:

firewalk -i p10p1 2003:0:4702:c402::2 2003:60:4010:11b0::12

Firewalk 5.0 [gateway ACL scanner]

fw_init_network(): target gateway and metric cannot be the same

Total packets sent:	0
Total packet errors:	0
Total packets caught	0
Total packets caught of interest	0
Total ports scanned	0
Total ports open:	0
Total ports unknown:	0

3.5.1 Conclusion

IPv6 is not supported by firewalk.

[7] *http://packetfactory.openwall.net/projects/firewalk/*

4 NETWORK / PORT SCANNING

4.1 Unicornscan

Unicornscan[8] would be useful for IPv6 network/port scanning due to its very fast performance. Unicornscan version 0.4.7 was tested.

Tests:

[root@localhost ~]# unicornscan -i vboxnet0 -Ir 160 -E fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
Main [Error cidr.c:263] dns lookup fails for `fdf3': Unknown host
Main [Error getconfig.c:434] cant add workunit for argument `fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa': dont understand address `f0c0:2567:7fe4:a00:27ff:fe74:ddaa'
what host(s) should i scan?, ive got nothing to do

4.1.1 Conclusion

IPv6 is not supported by Unicornscan.

4.2 Nmap

Nmap[9] ("Network Mapper") is a free and open source (*license*) utility for network discovery and security auditing. Nmap version 6.46 and 6.47 were tested.

Parameters:

-6, --IPv6	Use IP version 6.

Tests

- **-sn** performs neighbor solicitation, it does not send an ICMPv6 Echo Request, as it as the case in IPv4. Example:
  - *nmap -sn -6 fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa*
- You can define a subnet:
  - *nmap -sn -6 fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a/126*
- If you want to define a set of targets, you can/must use an input file:
  - *nmap -sn -6 -iL IPv6_targets.txt*

[8] *http://www.unicornscan.org/*
[9] *http://nmap.org/*

Starting Nmap 6.46 (http://nmap.org) at 2014-05-31 13:43 EEST
Nmap scan report for fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
Host is up (0.00087s latency).
MAC Address: 08:00:27:74:DD:AA (Cadmus Computer Systems)
Nmap scan report for fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a
Host is up (0.00037s latency).
MAC Address: 08:00:27:D1:D1:7A (Cadmus Computer Systems)
Nmap scan report for fdf3:f0c0:2567:7fe4:a00:27ff:fe6a:ca6a
Host is up (0.00083s latency).
MAC Address: 08:00:27:6A:CA:6A (Cadmus Computer Systems)
Nmap scan report for fdf3:f0c0:2567:7fe4:a00:27ff:fefc:f99a
Host is up (0.00031s latency).
MAC Address: 08:00:27:FC:F9:9A (Cadmus Computer Systems)
Nmap scan report for fdf3:f0c0:2567:7fe4:c0c6:5389:f6e:99c0
Host is up (0.0029s latency).
MAC Address: 08:00:27:8E:96:84 (Cadmus Computer Systems)
Nmap scan report for fdf3:f0c0:2567:7fe4:7874:48a6:2a9d:6a73
Host is up (0.0018s latency).
MAC Address: 08:00:27:4D:30:2F (Cadmus Computer Systems)
Nmap done: 6 IP addresses (6 hosts up) scanned in 0.41 seconds

- Generic (default) port scanning:

nmap -6 -iL IPv6_targets.txt
Starting Nmap 6.46 (http://nmap.org) at 2014-05-31 13:45 EEST
Nmap scan report for fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
...<snipped for brevity>
49159/tcp open unknown
49160/tcp open unknown
MAC Address: 08:00:27:8E:96:84 (Cadmus Computer Systems)
Nmap scan report for fdf3:f0c0:2567:7fe4:7874:48a6:2a9d:6a73
Host is up (0.0013s latency).
All 1000 scanned ports on fdf3:f0c0:2567:7fe4:7874:48a6:2a9d:6a73 are filtered
MAC Address: 08:00:27:4D:30:2F (Cadmus Computer Systems)
Nmap done: 6 IP addresses (6 hosts up) scanned in 202.44 seconds 10

- You CANNOT define a range of IPv6 addresses, i.e.:

./nmap -sn -6 fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa-ddbb

You receive a "Failed to resolve "fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa-ddbb" error message.

- You CANNOT define a list of IPv6 addresses, i.e.:

nmap -sn -6 fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa,fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a

You receive a corresponding "Failed to resolve" message.

4.2.1 Conclusion

Regarding network scanning:

- You CANNOT define a range of IPv6 addresses.
- You CANNOT define a list of IPv6 addresses.

[10]The same port scanning (TCP SYN scan against 6 targets and 1000 ports per target) took about 156 sec in the default configuration).

5 IPv6 FINGERPRINTING

5.1 Nmap

From http://nmap.org/book/osdetect-ipv6-methods.html: "Nmap has a similar but separate OS detection engine specialized for IPv6. At a high level, the technique is the same: send probes, collect responses, and match the set of responses against a database. The differences are in the specific probes used, and in the way they are matched.

IPv6 OS detection is used just like IPv4. Just use the -6 and -O options together. For example, nmap -6 -O <target>."

Nmap 6.47 released 366 new OS fingerprints.

Test:

Our targets are:

- Fedora
- Kali
- Windows 7
- OpenBSD 5.5
- FreeBSD 10
- Centos 6.5.

nmap -6 -iL IPv6_targets.txt -O

Results:

Only Windows was detected.

Running: Microsoft Windows Vista|7|2008

OS CPE: cpe:/o:microsoft:windows_vista::sp2 cpe:/o:microsoft:windows_7::sp1 cpe:/o:microsoft:windows_server_2008:r2:sp1 cpe:/o:microsoft:windows_8

OS details: Microsoft Windows Vista SP2 or Windows 7 SP1 or Windows Server 2008 R2 SP1 or Windows 8 Consumer Preview

In all the other cases:

No OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/.

If we repeat the tests using IPv4, all the OS families are detected (Linux, FreeBSD, OpenBSD, and Windows)

5.1.1 Conclusion

Although IPv6 fingerprinting is supported under IPv6, it is not that effective yet.

5.2 Xprobe2

Xprobe2, used to perform fingerprinting remote TCP/IP stacks was tested in version 0.3.

Test:

xprobe2 fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, meder@o0o.nu

[+] Target is fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
Can not resolve fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa: Unknown host

Same for xprobe2 [fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]

5.2.1 Conclusion

IPv6 is not supported by Xprobe2.

5.3 p0f

p0f[11], a tool that utilizes an array of sophisticated, purely passive traffic fingerprinting mechanisms was tested in version 3.07b.

Tests:

I launched nmap using *nmap -sn -6 -iL IPv6_targets.txt* to generate some traffic.

Used command: */p0f -f p0f.fp -r nmap-O-IPv6.pcap -o results.txt*

5.3.1 Conclusion

- It recognizes IPv6 traffic.
- It seems to recognize Linux (as Linux 2.2.x-3.x) and Windows hosts (as "Windows 7 or 8"), but not BSD hosts.
- More testing on this field is required though using normal traffic, but, definitely, IPv6 fingerprinting is supported. The only question is how effective it really is.

5.4 Amap

Amap[12] is an innovative tool to perform application protocol detection. Tested version was 5.4.

IPv6 is supported. You can use IPv6 by either using the -6 switch, or by using amap6 directly.

Example:

amap6 fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa 22 -bqv
Using trigger file /usr/local/etc/appdefs.trig ... loaded 30 triggers
Using response file /usr/local/etc/appdefs.resp ... loaded 346 responses
Using trigger file /usr/local/etc/appdefs.rpc ... loaded 450 triggers
amap v5.4 (www.thc.org/thc-amap) started at 2014-05-31 18:29:38 - APPLICATION MAPPING mode
Total amount of tasks to perform in plain connect mode: 23
Waiting for timeout on 23 connections ...
Protocol on [fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]:22/tcp matches ssh - banner: SSH-2.0-OpenSSH_6.4\r\nProtocol mismatch.\n
Protocol on [fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]:22/tcp matches ssh-openssh - banner: SSH-2.0-OpenSSH_6.4\r\nProtocol mismatch.\n
amap v5.4 finished at 2014-05-31 18:29:44

Second test: Used as an input file in an Nmap machine readable outputfile to read ports from. This file was produced for our targets using nmap -oM option.

---

[11] *http://lcamtuf.coredump.cx/p0f3*
[12] *https://www.thc.org/thc-amap*

Result:

Amap -6 or amap6 were not able to use this file properly (e.g. Warning: Could not connect (unreachable) to [[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]]:22/tcp, disabling port (EUNKN)).

5.4.1 Conclusion

There shouldn't be any problem by using amap/amap6 with IPv6 when you use just a single address as an input. Its detection performance does not depend on layer-3 and hence, it should be the same as using IPv4. However, its creator, Marc Heuse, recommended amap just for UDP IPv6 scans only. Otherwise, it is considered outdated. Moreover, when you try to read the addresses/ports from an nmap machine readable output file (produced using nmap -oM), this is not performed properly and the service fingerprint fails.

6 BRUTE-FORCING

6.1 Hydra

Hydra[13] is a very fast network logon cracker that supports many different services. Hydra was tested in version 8.1-dev.

Parameters:

-6	prefer IPv6 addresses

Examples:

- Simple usage:

hydra -6 fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa -l root -p mypassword ssh

Hydra v8.1-dev (c) 2014 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2014-06-18 13:01:17
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 tasks, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking service ssh on port 22
[22][ssh] host: fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa login: root password: mypassword
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2014-06-18 13:01:17

- Read the targets from a file:

hydra -6 -M /root/IPv6_targets.txt -l root -p atlas930 ssh

Hydra v8.1-dev (c) 2014 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2014-06-18 13:02:58
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 6 servers, overall 6 tasks, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking service ssh on port 22
[ERROR] could not resolve address: fdf3
[ERROR] could not resolve address: f0c0
[ERROR] could not resolve address: 7fe4
[ERROR] could not resolve address: a00
[ERROR] could not resolve address: 27ff
[ERROR] unknown address string size!
[ERROR] unknown address string size!
[ERROR] could not connect to ssh://(null):22
[ERROR] unknown address string size!
[ERROR] unknown address string size!
[ERROR] could not connect to ssh://(null):2567
[ERROR] could not connect to ssh://0.0.10.7:7
[ERROR] unknown address string size!
[ERROR] unknown address string size!
[ERROR] could not connect to ssh://(null):22
[ERROR] unknown address string size!
[ERROR] unknown address string size!
[ERROR] could not connect to ssh://(null):27
[ERROR] unknown address string size!

[13] *Website: https://www.thc.org/thc-hydra*

[ERROR] could not connect to ssh://(null):22
0 of 6 targets completed, 0 valid passwords found
[ERROR] 6 targets did not resolve or could not be connected
Hydra (http://www.thc.org/thc-hydra) finished at 2014-06-18 13:02:59

6.1.1 Conclusion

Hydra partially supports IPv6. You can define a single IPv6 target using -6, but you cannot define a list of targets in a file using -M. No options for adding IPv6 Extension headers or other IPv6-related capabilities (e.g. for evading purposes).

6.2 Medusa

Medusa[14] is intended to be a speedy, massively parallel, modular, login brute-forcer. Version 2.1.1 was tested.

There is no option/parameter to enable IPv6.

Test:

medusa -h fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa -u root -p pass -M ssh
Medusa v2.1.1 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>
NOTICE: ssh.mod: failed to connect, port 22 was not open on fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa

6.2.1 Conclusion

IPv6 is not supported by medusa.

6.3 Ncrack

Ncrack[15] is a high-speed network authentication cracking tool. Tested version was 0.4ALPHA

Parameters:

-6	Enable IPv6 scanning

Warning: This option was just added and it is currently experimental, so please notify us for any problems and bugs related to it.

The command syntax is the same as usual except that you also add the -6 option. Of course, you must use IPv6 syntax if you specify an address rather than a hostname. An address might look like 3ffe:7501:4819:2000:210:f3ff:fe03:14d0, so hostnames are recommended. The output looks the same as usual, with the IPv6 address on the "Discovered credentials" line being the only IPv6 give away.

In reality it doesn't seem to work (an "invalid port number" error message is displayed).

6.3.1 Conclusion

Although it is claimed to be supported, at least experimentally, IPv6 does not seem to work.

[14] *Website: http://foofus.net/goons/jmk/medusa/medusa.html*
[15] *http://nmap.org/ncrack/*

7 PACKET CRAFTING

7.1 Hping

Hping[16] assembles and sends custom ICMP, UDP, or TCP packets and then displays any replies. The tested version was 20051105-20.

hping 3.0.0-alpha-1 ($Id: release.h,v 1.4 2004/04/09 23:38:56 antirez Exp $)

hping fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
Unable to resolve 'fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa

7.1.1 Conclusion

No support for IPv6.

7.2 Nping

From the man pages: "*Nping[17] is an open-source tool for network packet generation, response analysis and response time measurement. Nping allows users to generate network packets of a wide range of protocols, letting them tune virtually any field of the protocol headers. While Nping can be used as a simple ping utility to detect active hosts, it can also be used as a raw packet generator for network stack stress tests, ARP poisoning, Denial of Service attacks, route tracing, and other purposes.*" The test version was 0.6.46 & 0.6.47.

Parameters:

-6, --IPv6	Use IP version 6.
--dest-ip	Set destination IP address. (Used as an alternative to {target specification}).
--hop-limit	Set hop limit (same as IPv4 TTL).
--traffic-class <class>	Set traffic class.
--flow <label>	Set flow label.

PAYLOAD OPTIONS:

--data <hex string>	: Include a custom payload.
--data-string <text>	: Include a custom ASCII text.
--data-length <len>	: Include <len> random bytes as payload.

[16] *http://sectools.org/tool/hping/*
[17] *http://nmap.org/nping/*

Test:

nping -6 --dest-ip fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa --tcp-connect -p 22

Result:

nping -6 --dest-ip fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa --tcp-connect -p 22
Starting Nping 0.6.46 (http://nmap.org/nping) at 2014-05-31 21:13 EEST
SENT (0.0024s) Starting TCP Handshake > fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22
RECV (0.0029s) Handshake with fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 completed
SENT (1.0041s) Starting TCP Handshake > fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22
RECV (1.0051s) Handshake with fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 completed
SENT (2.0057s) Starting TCP Handshake > fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22
RECV (2.0063s) Handshake with fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 completed
SENT (3.0075s) Starting TCP Handshake > fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22
RECV (3.0080s) Handshake with fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 completed
SENT (4.0090s) Starting TCP Handshake > fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22
RECV (4.0094s) Handshake with fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 completed
Max rtt: 1.035ms | Min rtt: 0.454ms | Avg rtt: 0.640ms
TCP connection attempts: 5 | Successful connections: 5 | Failed: 0 (0.00%)
Nping done: 1 IP address pinged in 4.01 seconds

However, when trying to use a comma-separated list, an error is generated. Moreover, subnets (e.g. /64) are not allowed, as well there is no option to read the IPv6 addresses from an input file.

No support for IPv6 Extension headers either.

7.2.1 Conclusion

Very limited (basic) IPv6 support. If arbitrary packet crafting is required, the use of *Scapy* is highly recommended, because it is much more flexible.

7.3 Scapy

Scapy[18] is a powerful interactive packet manipulation program.

7.3.1 Conclusion

Very good support of IPv6, not all the latest IPv6 Extension headers or protocols are supported though (e.g. MLDv2).

[18] *http://www.secdev.org/projects/scapy/*

8 REMOTE SHELLS

8.1 Ncat

Ncat[19] is a feature-packed networking utility which reads and writes data across networks from the command line. Tested version was 6.47.

Parameters:

-6, --IPv6 Use IP version 6.

IPv6 is supported.

Examples:

Listen mode: *ncat -6 -l*

Connect mode: *ncat -6 fdf3:f0c0:2567:7fe4:800:27ff:fe00:0*

Default port: *31337*

It provides you a typical non-interactive shell.

Transferring a file:

$ ncat -l -6 > test.txt

$ ncat -6 fdf3:f0c0:2567:7fe4:800:27ff:fe00:0 < test.txt

Or, to exec commands remotely:

$ ncat -l -6 -e /bin/sh

An example output of Ncat unencrypted communication:

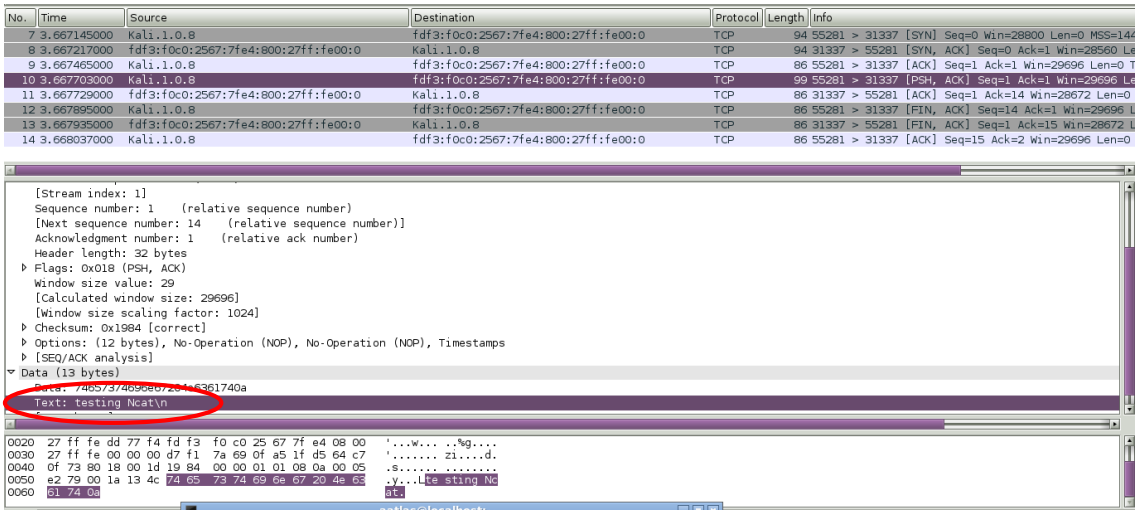

Figure 5 Ncat Unencrypted Communication – Example 1

However, ncat supports many handy features, and one of them is to use SSL for encrypting the communication channel (using the switch –ssl).

[19] *http://nmap.org/ncat/*

Another example output of Ncat encrypted communication is displayed below:

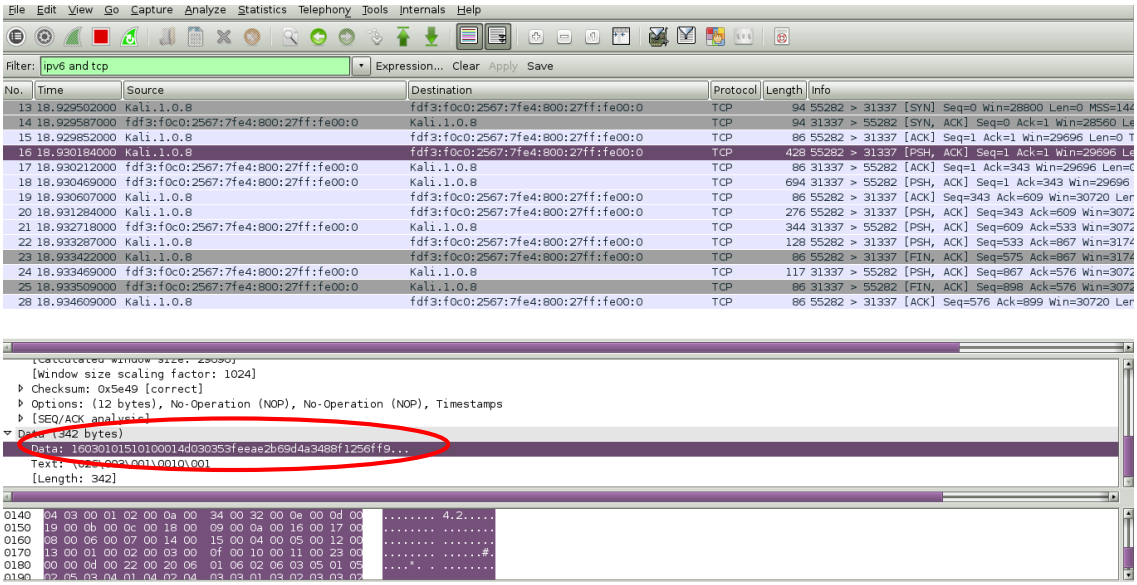

Figure 6 Ncat Unencrypted Communication – Example 2

8.1.1 Conclusion

Ncat works without problems, using IPv6. It also supports some handy features, like SSL encryption, even over IPv6.

9 LAN / MITM ATTACKS & OTHER

9.1 NSE Scripts

Nmap Scripting Engine (NSE)[20] scripts provide different options, introduced below. Let's start with the IPv6-specific NSE scripts provided by nmap.

Script location:

/usr/local/share/nmap/scripts/

Information obtained from:

http://nmap.org/nsedoc/

NSE Scripts specific for IPv6

dns-ip6-arpa-scan	Performs a quick reverse DNS lookup of an IPv6 network using a technique which analyzes DNS server response codes to dramatically reduce the number of queries needed to enumerate large networks.
ipv6-node-info	Obtains hostnames, IPv4 and IPv6 addresses through IPv6 Node Information Queries.
ipv6-ra-flood	Generates a flood of Router Advertisements (RA) with random source MAC addresses and IPv6 prefixes. Computers, which have stateless auto configuration enabled by default (every major OS), will start to compute IPv6 suffix and update their routing table to reflect the accepted announcement. This will cause 100% CPU usage on Windows and platforms, preventing to process other application requests.
targets-ipv6-multicast-echo	Sends an ICMPv6 echo request packet to the all-nodes link-local multicast address (ff02::1) to discover responsive hosts on a LAN without needing to individually ping each IPv6 address.
targets-ipv6-multicast-invalid-dst	Sends an ICMPv6 packet with an invalid extension header to the all-nodes link-local multicast address (ff02::1) to discover (some) available hosts on the LAN. This works because some hosts will respond to this probe with an ICMPv6 Parameter Problem packet.
targets-ipv6-multicast-mld	Attempts to discover available IPv6 hosts on the LAN by sending an MLD (multicast listener discovery) query to the link-local multicast address (ff02::1) and listening for any responses. The query's maximum response delay set to 0 to provoke hosts to respond immediately rather than waiting for other responses from their multicast group.
targets-ipv6-multicast-slaac	Performs IPv6 host discovery by triggering stateless address auto-configuration (SLAAC).

[20] *http://nmap.org/book/nse-scripts.html*

dhcp6	Minimalistic DHCP6 (Dynamic Host Configuration Protocol for IPv6) implementation supporting basic DHCP6 Solicit requests The library is structured around the following classes: ■ DHCP6.Option - DHCP6 options encoders (for requests) and decoders (for responses). ■ DHCP6.Request - DHCP6 request encoder and decoder. ■ DHCP6.Response - DHCP6 response encoder and decoder. ■ Helper - The helper class, primary script interface.
broadcast-dhcp6-discover	Sends a DHCPv6 request (Solicit) to the DHCPv6 multicast address, parses the response, then extracts and prints the address along with any options returned by the server.

NSE scripts that also provide information regarding IPv6:

address-info	Shows extra information about IPv6 addresses, such as embedded MAC or IPv4 addresses when available.
afp-serverinfo	Shows AFP server information. This information includes the server's hostname, IPv4 and IPv6 addresses, and hardware type (for example Macmini or MacBookPro).
resolveall	Resolves hostnames and adds every address (IPv4 or IPv6, depending on Nmap mode) to Nmap's target list. This differs from Nmap's normal host resolution process, which only scans the first address (A or AAAA record) returned for each host name.

Let's see some tests.

Tests:

- IPv6-node-info

Obtains hostnames, IPv4 and IPv6 addresses through IPv6 Node Information Queries. IPv6 Node Information Queries are defined in RFC 4620[21].

[21] *https://tools.ietf.org/html/rfc4620*

1. Introduction

This document specifies a mechanism for discovering information about names and addresses. The applicability of these mechanisms is currently limited to diagnostic and debugging tools and network management (e.g., node discovery). In the global internet, the Domain Name System (DNS) [1][2] is the authoritative source of such information and this specification is not intended to supplant or supersede it. In fact, in a well-supported network, the names and addresses dealt with by this mechanism will be the same ones, with the same relationships, as those listed in the DNS.

This new Node Information protocol provides facilities that are not found in the DNS, for example, discovering relationships between addresses without reference to names. The functions that do overlap with the DNS may be useful in serverless environments, for debugging, or in regard to link-local and unique-local addresses [3] that often will not be listed in the DNS.

Figure 7 RFC 4620

There are three useful types of queries:

qtype=2: Node Name

qtype=3: Node Addresses

qtype=4: IPv4 Addresses

Command example:

nmap -6 fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a --script ipv6-node-info --script-args='interface=vboxnet0' -sn

And this is the information obtained by an OpenBSD system (we displayed it as a Wireshark output).

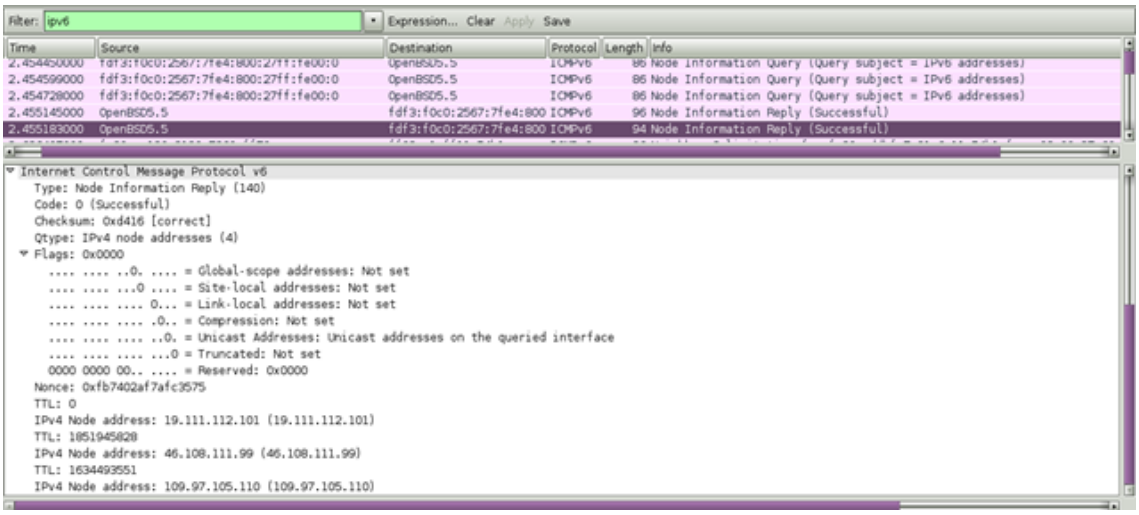

Figure 8 Wireshark Output nmap scan

As we can see, OpenBSD 5.5 happily responds to these queries and provides some information.

- IPv6-ra-flood

Floods the local link with Router Advertisements (RA) with random source MAC addresses and IPv6 prefixes. It increases the CPU load significantly on all the targeted OS making them quite unresponsive. It is quite interesting that this technique can still be effective.

nmap -6 --script ipv6-ra-flood.nse --script-args 'interface=vboxnet0'

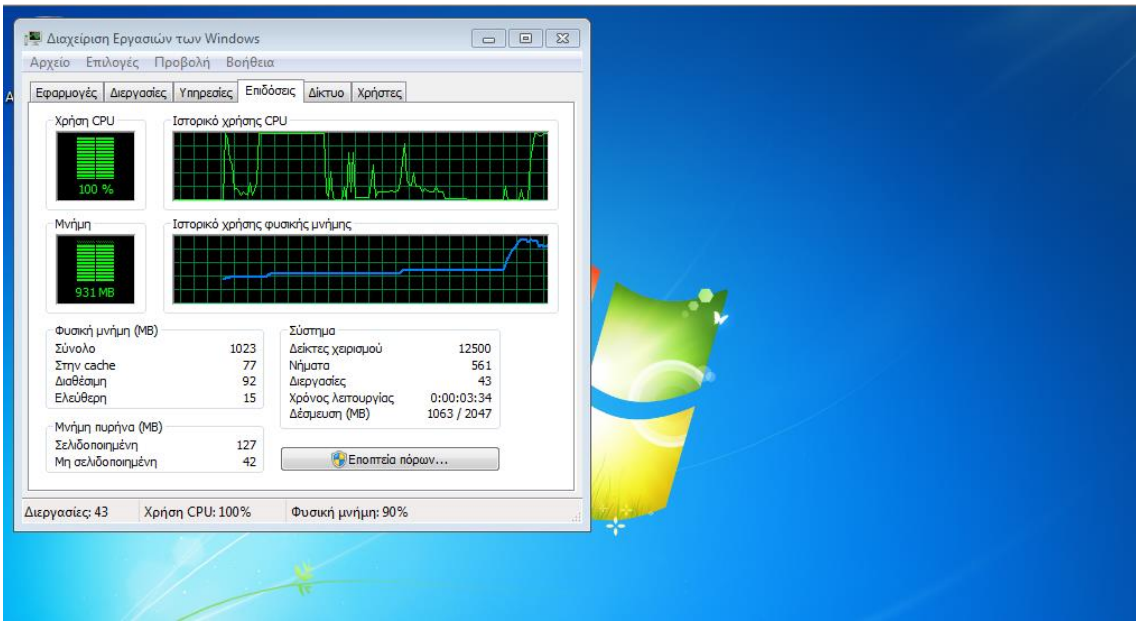

Figure 9 Flood router impact on a Windows 7 Operating System

- Targets-ipv6-multicast-echo

Sends an ICMPv6 echo request packet to the all-nodes link-local multicast address (ff02::1).

nmap -6 --script=targets-ipv6-multicast-echo.nse --script-args 'newtargets,interface=vboxnet0'
Starting Nmap 6.46 (http://nmap.org) at 2014-05-31 17:46 EEST
Pre-scan script results:
| targets-ipv6-multicast-echo:
| IP: fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a MAC: 08:00:27:d1:d1:7a IFACE: vboxnet0
| IP: fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa MAC: 08:00:27:74:dd:aa IFACE: vboxnet0
|_ IP: fdf3:f0c0:2567:7fe4:481e:249a:1d4f:77ec MAC: 08:00:27:6a:ca:6a IFACE: vboxnet0
... etc

Only Linux and OpenBSD (!) systems respond (not Windows and FreeBSD). Not very effective technique on its own.

- Targets-ipv6-multicast-invalid-dst

It sends an ICMPv6 packet with an invalid extension header to the all-nodes link-local multicast address (ff02::1) to discover (some) available hosts on the LAN. This works because some hosts will respond to this probe with an ICMPv6 Parameter Problem packet.

nmap -6 --script=targets-ipv6-multicast-invalid-dst.nse --script-args 'newtargets,interface=vboxnet0' -sP
Starting Nmap 6.46 (http://nmap.org) at 2014-05-31 17:48 EEST
Pre-scan script results:
| targets-ipv6-multicast-invalid-dst:
| IP: fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa MAC: 08:00:27:74:dd:aa IFACE: vboxnet0
| IP: fdf3:f0c0:2567:7fe4:481e:249a:1d4f:77ec MAC: 08:00:27:6a:ca:6a IFACE: vboxnet0
| IP: fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a MAC: 08:00:27:d1:d1:7a IFACE: vboxnet0
|_ IP: fdf3:f0c0:2567:7fe4:c099:d19b:fbd4:aadb MAC: 08:00:27:8e:96:84 IFACE: vboxnet0

Four systems responded (Fedora, Centos, OpenBSD and Windows 8.1). Slightly more effective technique than targets-ipv6-multicast-echo. Similar results to alive6 of the thc-ipv6 attack toolkit.

An example of the sent packets is displayed in the Wireshark output shown below:

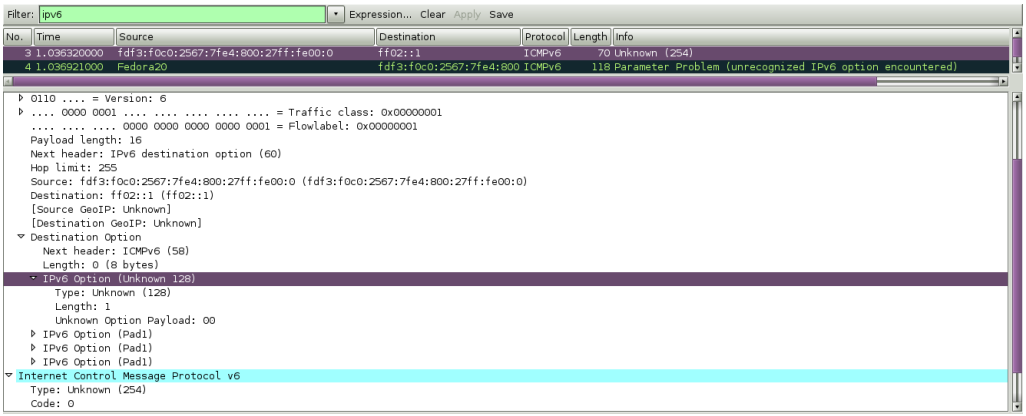

Figure 10 Wireshark Output targets-ipv6-multicast-invalid-dst

- Targets-ipv6-multicast-mld

It attempts to discover available IPv6 hosts on the LAN by sending an MLD (multicast listener discovery) query to the link-local multicast address (ff02::1) and listening for any responses.

nmap -6 --script=targets-ipv6-multicast-mld.nse --script-args 'interface=vboxnet0'

An example output for the sent packet is displayed below:

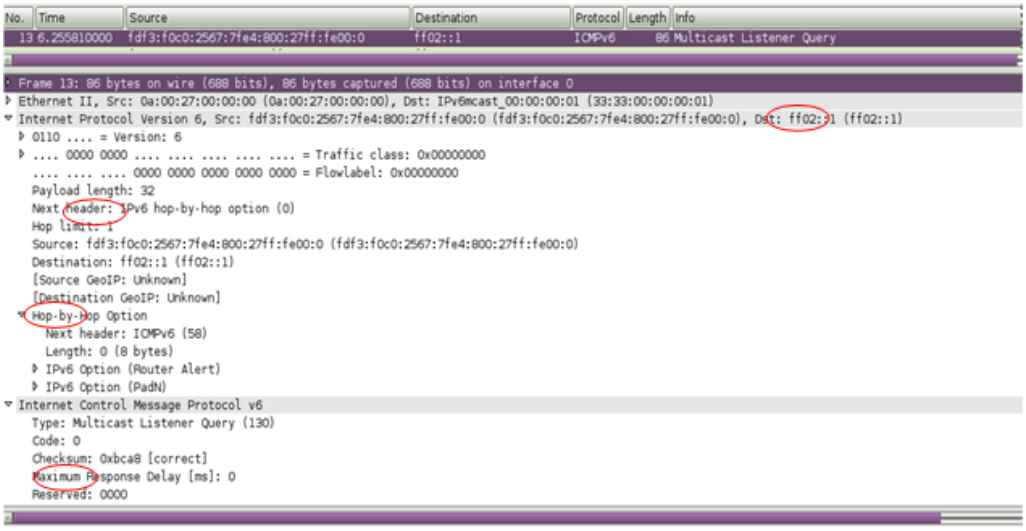

Figure 11 Targets-ipv6-multicast-mld

Unfortunately, it doesn't seem to work effectively.

- Targets-ipv6-multicast-slaac

This script attempts to perform IPv6 host discovery by triggering stateless address auto-configuration (SLAAC). It doesn't seem to work, though.

- Resolveall

Resolves hostnames and adds every address (IPv4 or IPv6, depending on Nmap mode) to Nmap's target list.

Command:

nmap -6 --script=resolveall --script-args=newtargets,resolveall.hosts={www.ernw.de}
Starting Nmap 6.47 (http://nmap.org) at 2014-09-02 23:22 EEST
Pre-scan script results:
| resolveall:
| Host 'www.ernw.de' resolves to:
| 2003:60:4010:1090::11
|_ Successfully added 1 new targets
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 8.12 seconds

9.1.1 Conclusion

Several NSE scripts either support IPv6 or they are IPv6-specific. Some of them do not appear to work properly. From the rest, the most interesting/effective ones are the following:

- IPv6-ra-flood (quite effective even against the latest OS).
- Targets-ipv6-multicast-invalid-dst (this produces similar results to alive6 of the thc-ipv6 attacking toolkit).
- Targets-ipv6-multicast-echo
- IPv6-node-info
- Resolveall

9.2 Ettercap

Ettercap[22] is a comprehensive suite for man in the middle attacks. The tested version was 0.8.0.

- IPv6 support was added at version 0.7.5.
- DNS spoofing for IPv6 addresses plugin was added in version 0.8.0.

NOTE:

IPv6 is not supported out-of-the-box in the rpm package. In the source code, you must enable it: (Edit build/CmakeCache.txt and change ENABLE_IPV6:BOOL=OFF to ENABLE_IPV6:BOOL=ON).

Test:

/usr/local/bin/ettercap -i vboxnet0 -T //fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa/ //fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a/
ettercap 0.8.0 copyright 2001-2013 Ettercap Development Team
Listening on:
vboxnet0 -> 0A:00:27:00:00:00
192.168.56.1/255.255.255.0
fe80::800:27ff:fe00:0/64
fdf3:f0c0:2567:7fe4:800:27ff:fe00:0/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to UID 65534 GID 65534...
plugin ec_sslstrip.so cannot be loaded...
32 plugins
42 protocol dissectors
57 ports monitored
16074 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
** |==========>| 100.00 %*
Scanning for merged targets (2 hosts)...
** |==========>| 100.00 %*
0 hosts added to the hosts list...
Starting Unified sniffing...
Text only Interface activated...

Not sure how to launch an IPv6 MITM attack using ettercap. We didn't find any related MITM module in the man pages. We pinged the hosts using IPv6 and we did not get anything in real time (just IPv4). But when we press l (list hosts), IPv6 hosts are listed:

```
Hosts list:

1)      fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa  08:00:27:74:DD:AA
2)      fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a  08:00:27:D1:D1:7A
3)      fe80::a00:27ff:fe74:ddaa        08:00:27:74:DD:AA
```

Figure 12 IPv6 Hosts List Ettercap

IPv6-related Filters:

- DNS spoofing for IPv6 addresses
- Support for IPv4 and IPv6 Tunnels

[22] *http://ettercap.github.io/ettercap/*

9.2.1 Conclusion

Although ettercap supports IPv6 addresses, critical modules (like MITM attacks) do not seem to be implemented yet.

9.3 Cain & Abel

Cain & Abel[23], a password recovery tool for Windows was tested with version 4.9.56 on Windows 8.1.

Ping/ping6 traffic was sent to Cain while its sniffer was running, but just the IPv4 address of the sender was captured.

9.3.1 Conclusion

IPv6 is not supported.

9.4 Net-snmp

Net-SNMP[24] is a suite of applications used to implement SNMPv1, SNMPv2c and SNMPv3 using both IPv4 and IPv6.Tested version was 5.7.3.pre5.

When you compile it, you need to enable IPv6 support (not enabled by default).

Example:

snmpwalk -Os -c public -v 1 fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa system

The SNMP operation was confirmed with Wireshark.

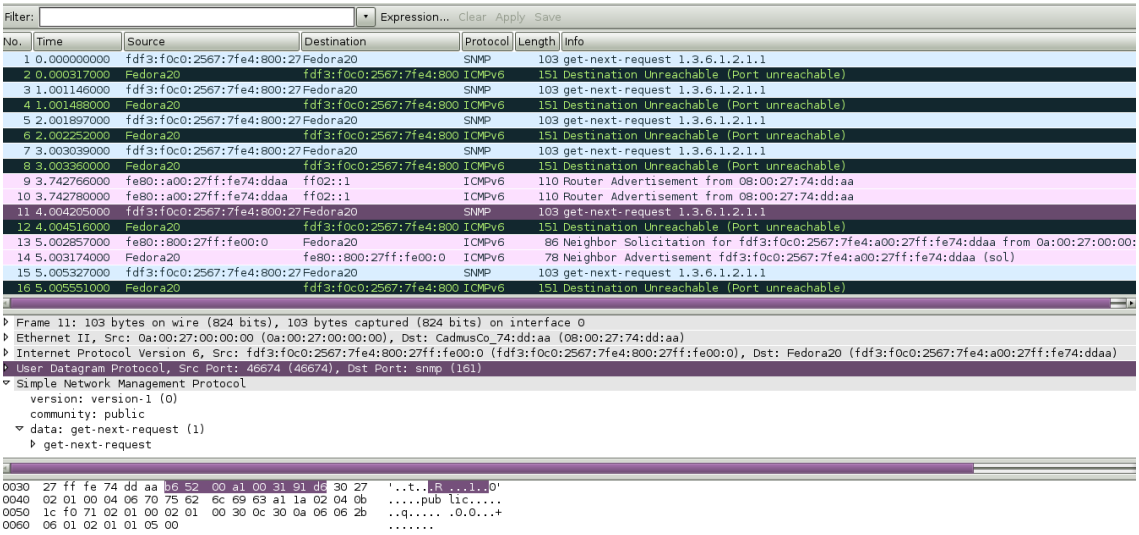

Figure 13 Wireshark Output SNMP Walk

9.4.1 Conclusion

IPv6 is supported by Net-snmp, but you must compile it with this option enabled (it is not by default).

[23] *http://www.oxid.it/cain.html*
[24] *http://www.net-snmp.org/*

10 VULNERABILTY SCANNERS

10.1 Nessus

The market defining vulnerability scanning solution was tested version with Nessus Home, Engine: 5.2.7.

Web UI:

2.3.8 (master #98), Plugins

Last Updated:

August 28, 2014

How to Define your IPv6 Targets:

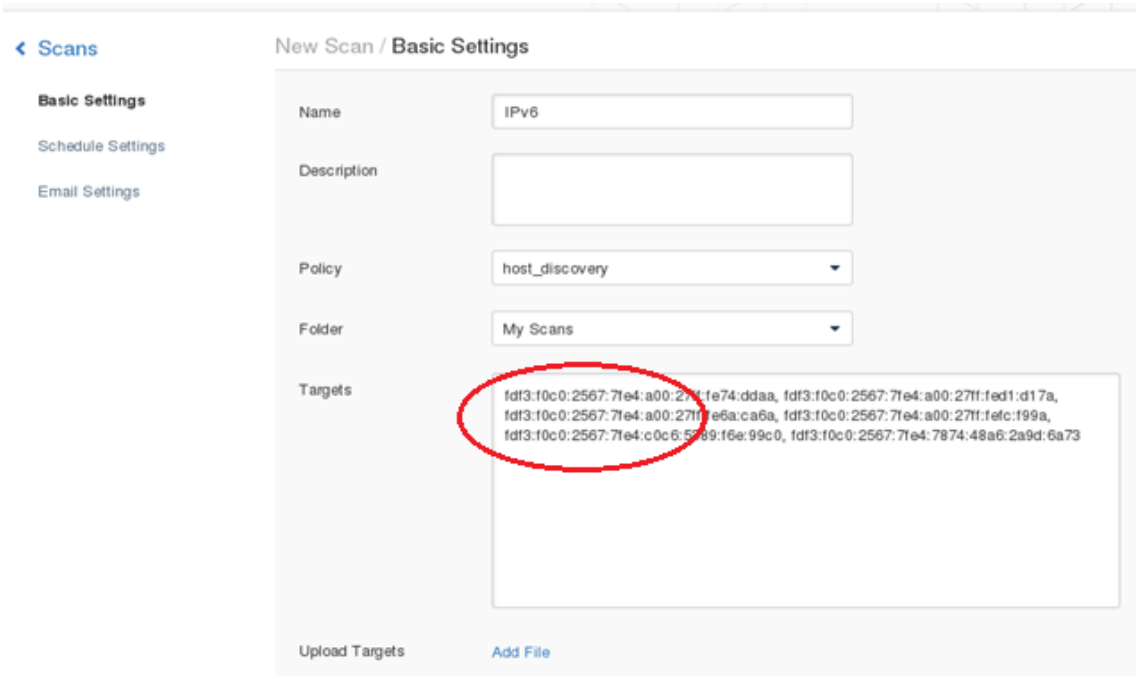

Figure 14 Nessus IPv6 Targets

Let's see, using a Wireshark output, the methods by Nessus for IPv6 discovery:

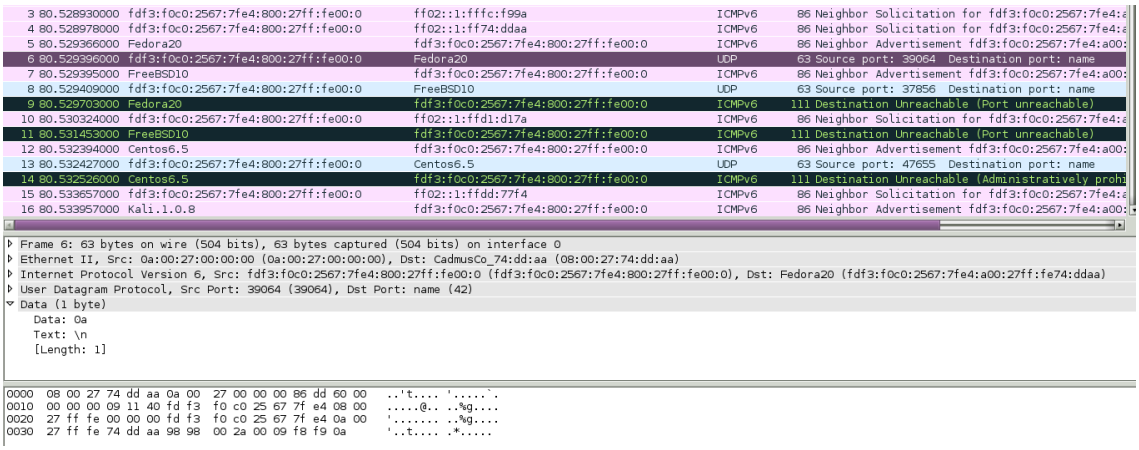

Figure 15 Wireshark Output ICMPv6 Echo Request

As we can see, ICMPv6 Echo Request plus UDP Port Scanning at port 42 are used! Moreover, port scanning is also launched by the IPv6 host discovery.

Results:

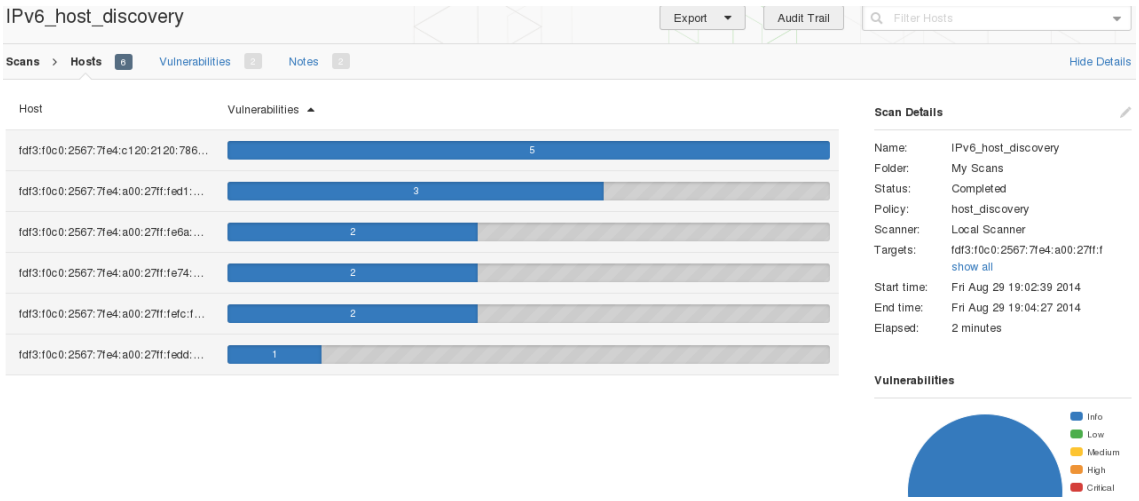

Figure 16 Nessus IPv6 Host Discovery

As we can see, all targets were identified.

Some more detailed results are displayed below:

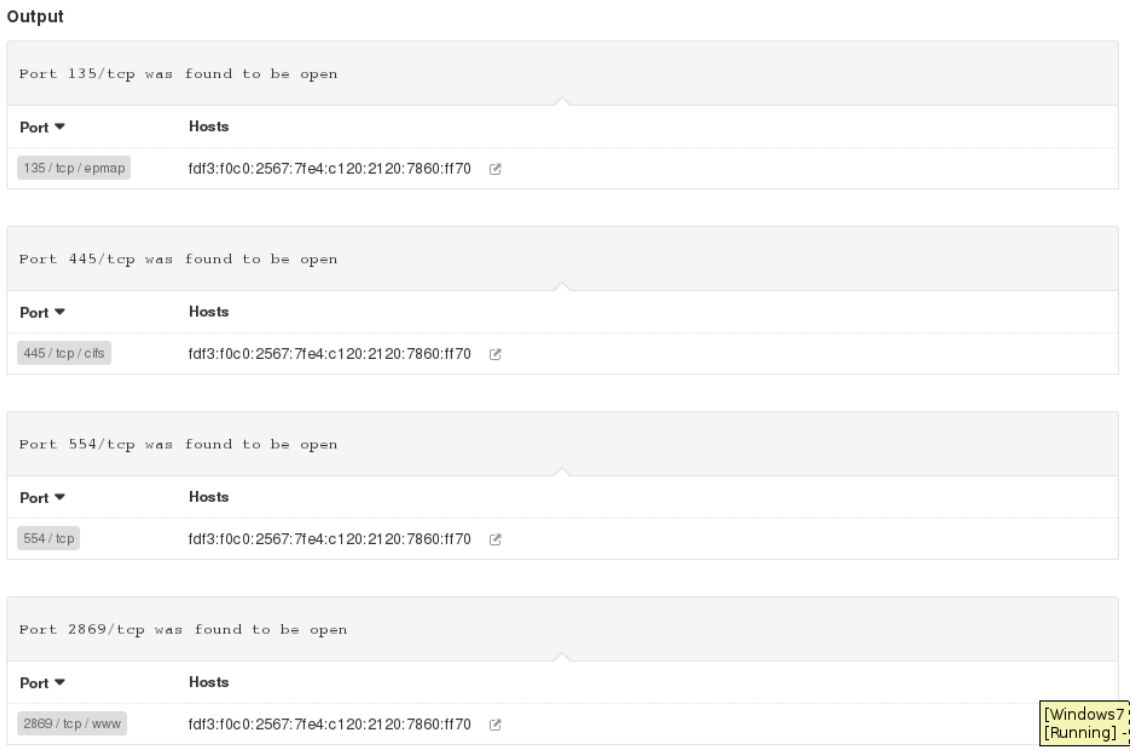

Figure 17 Detailed Nessus Output IPv6 Host Discovery

Other ways to define your IPv6 targets:

Link local addresses should be defined as targets like this: fe80::800:27ff:fe00:0%eth1.

When a /64 network is defined (e.g. fdf3:f0c0:2567:7fe4::/64)...

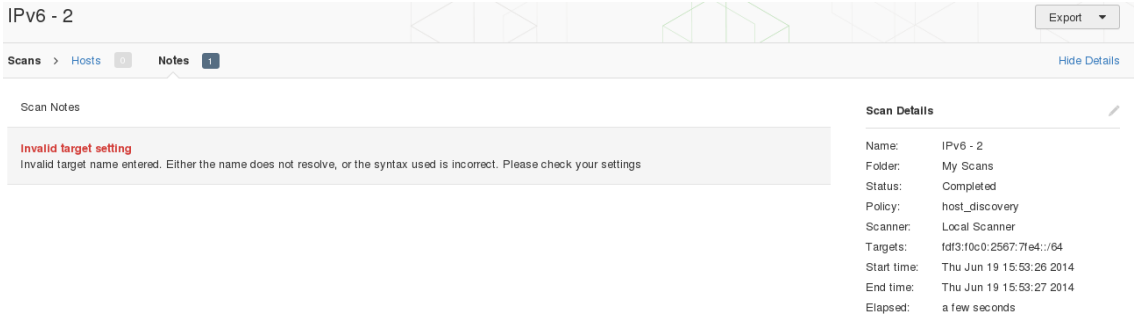

Figure 18 Nessus Invalid Target Settings

...we get an error message ("Invalid Target setting").

Similar errors we get when:

- A /64 network is defined as fdf3:f0c0:2567:7fe4::0/64
- Ranges of addresses are defined, e.g. fdf3:f0c0:2567:7fe4:a00:27ff:fe74:dd00 – fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddff or fdf3:f0c0:2567:7fe4:a00:27ff:fe74:dd00-ddff

IPv6 Plugins

Let's examine if IPv6-related plugins are supported in Nessus:

After selecting a policy and then "Plugins" from the left pane.

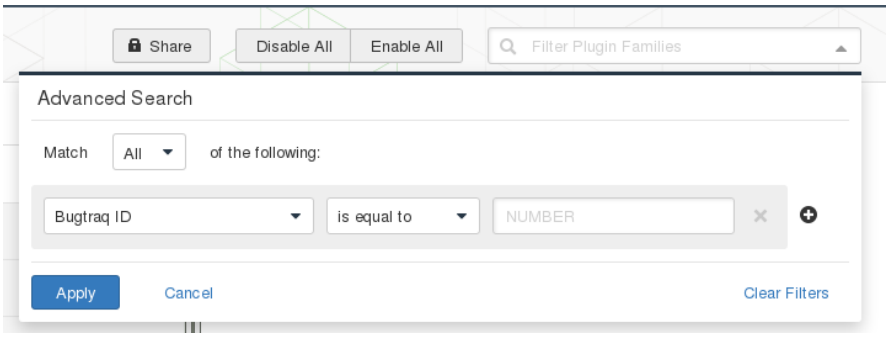

Figure 19 Advanced Search Nessus

We get a plenty of IPv6-related plugins.

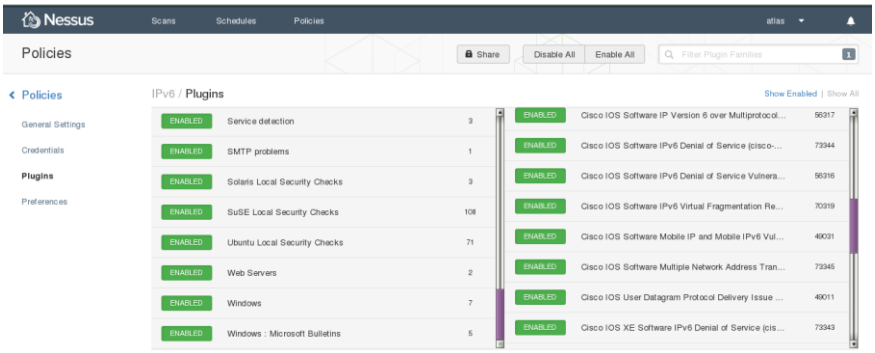

Figure 20 Nessus Plugins

And also some nice recommendations.

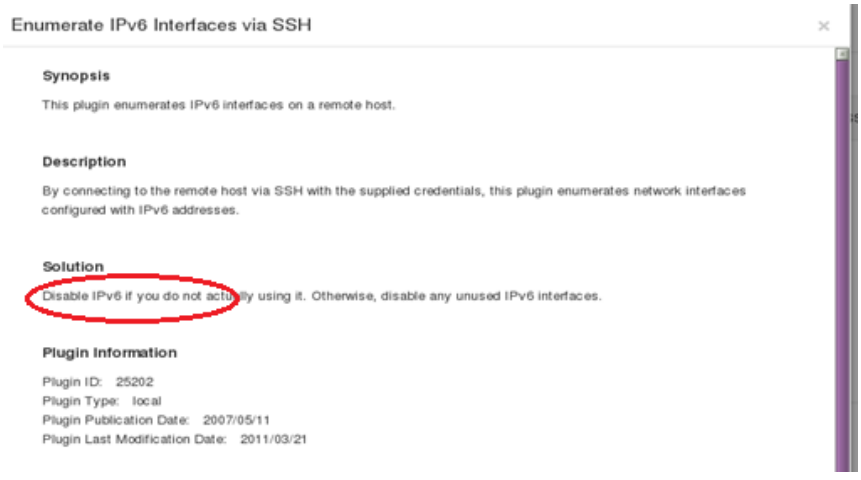

Figure 21 Nessus Enumeration IPv6 Interfaces via SSH

Note:

Nessus under Windows:

*"Nessus works under *nix platforms with IPv6, but the windows stack doesn't support the features needed for nessus to work." [1]*

10.1.1 Conclusion

Nessus supports IPv6 addresses as targets, but NOT using IPv6 prefixes or IPv6 ranges. The IPv6 host discovery module does not use many methods to discovery IPv6 hosts (e.g. IPv6 datagrams with erroneous parameters or extension headers, etc.). However, it incorporates several IPv6-related vulnerability discovery plugins. To sum-up, Nessus can be used against IPv6 networks but it is recommended that during the discovery phase more specialized tools, like the thc-ipv6 attack toolkit, should also be used.

11 WEB PENETRATION TESTING

11.1 BurpSuite

Burp Suite[25] is an integrated platform for performing security testing of web applications. The tested version was 1.6, free edition.

Tests:

As a target, an IPv6-only (very basic) website was used.

Results:

By configuring the web browser to use 127.0.0.1:8080, burpsuite fetches and spiders IPv6 targets:

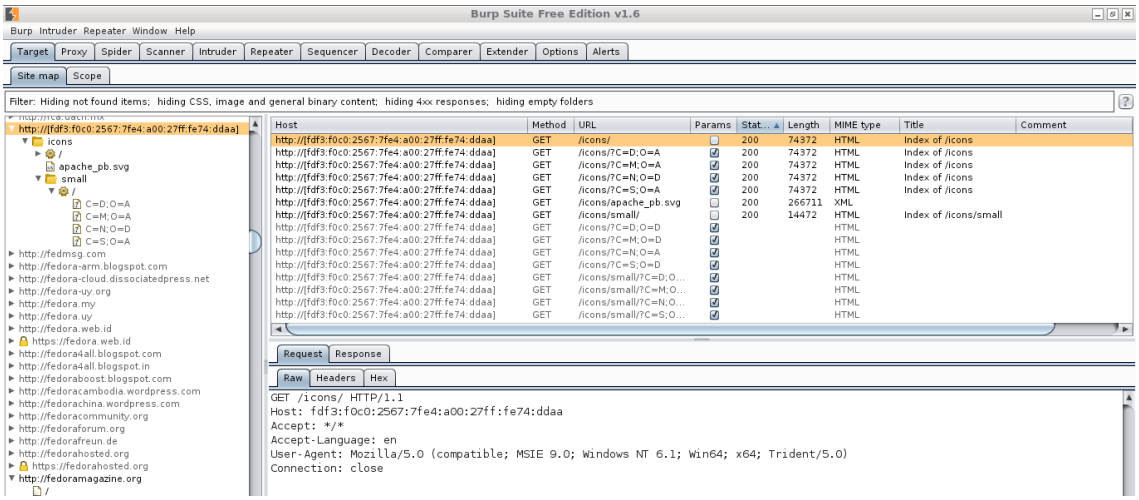

Figure 22 Burp Suite IPv6 Targets

When you use IPv6 local host (::1 or [::1]) at the proxy settings of the web browser, you get a "server not found" error message at your web browser. No big deal to use IPv4 localhost address for this purpose though!

11.1.1 Conclusion

Burpsuite can spider IPv6 targets smoothly!

11.2 ZAPROXY

The OWASP Zed Attack Proxy (ZAP)[26] is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. Tested version was 2.3.1.

[25] *http://portswigger.net/index.html*
[26] *https://code.google.com/p/zaproxy/*

Results:

IPv6 addresses cannot be attacked directly:

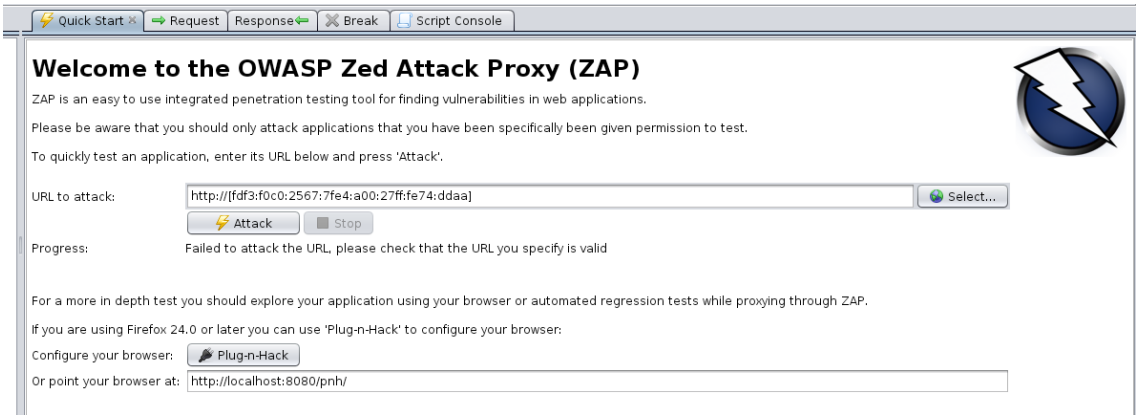

Figure 23 Zaproxy IPv6 URL To Attack

When used as a proxy, still an IPv6 address cannot be fetched:

Figure 24 Zaproxy Invalid Address Proxy Error

11.2.1 Conclusion

IPv6 is not supported by ZAP.

11.3 Nikto

Nikto[27] is an Open Source (*GPL*) web server scanner which performs comprehensive tests against web servers for multiple items. Tested version was 2.1.5.

[27] https://www.cirt.net/Nikto2

Test:

[aatlas@localhost nikto-2.1.5]$ perl nikto.pl -host http://[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]
- Nikto v2.1.5

+ ERROR: Cannot resolve hostname '[fdf3'
+ 0 host(s) tested

11.3.1 Conclusion

IPv6 is not supported by Nikto.

11.4 Skipfish

Skipfish[28] is an active web application security reconnaissance tool, tested with version 2.10b.

There is no optional parameter to define IPv6 addresses (like -6).

Test:

./skipfish -o output_dir2 http://[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]
skipfish web application scanner - version 2.10b
[!] WARNING: Wordlist '/dev/null' contained no valid entries.
[-] PROGRAM ABORT : Scan target 'http://[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]' is not a valid absolute URL.
Stop location : main(), src/skipfish.c:736

11.4.1 Conclusion

IPv6 is not supported by skipfish.

11.5 Sqlmap

Sqlmap[29] is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. Tested version was sqlmap/1.0-dev-nongit-20140823.

Sqlmap supports IPv6 out-of-the-box.

Example:

$ python sqlmap.py -u http://[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]
...<snipped for brevity> ...
[*] starting at 14:28:35
[14:28:35] [INFO] testing connection to the target URL
[14:28:35] [WARNING] the web server responded with an HTTP error code (403) which could interfere with the results of the tests
[14:28:35] [INFO] testing if the target URL is stable. This can take a couple of seconds
[14:28:37] [INFO] target URL is stable
[14:28:37] [CRITICAL] no parameter(s) found for testing in the provided data (e.g. GET parameter 'id' in 'www.site.com/index.php?id=1')
[14:28:37] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 2 times

[28] https://code.google.com/p/skipfish

[29] sqlmap/1.0-dev-nongit-20140823

[*] shutting down at 14:28:37

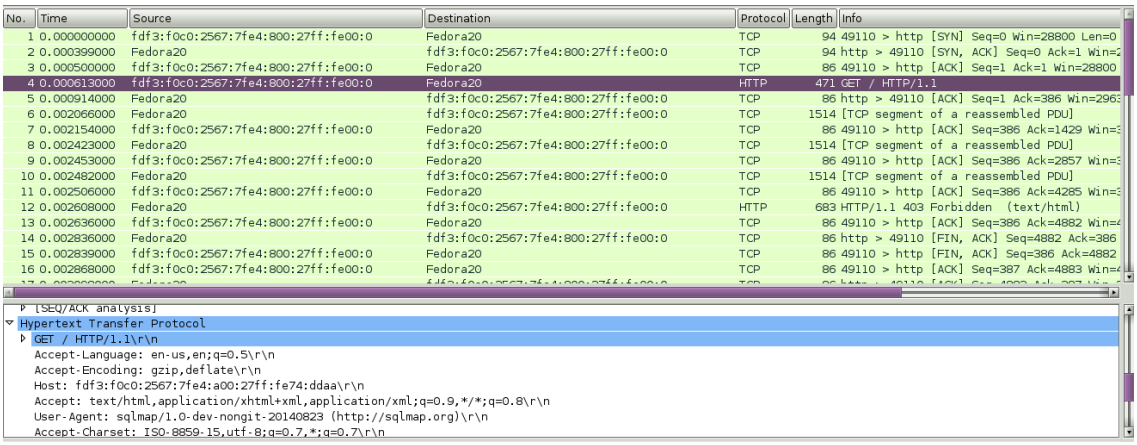

Figure 25 Wireshark Output SqlMap

11.5.1 Conclusion

IPv6 is supported by sqlmap.

11.6 sqlninja

Sqlninja[30] is a SQL server injecting and takeover tool. The version tested was the latest dev version download in 23rd August 2014 via svn.

The following sqlninja configurations were tested:

<...snipped for brevity...>
--httprequest_start--
GET http://[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]/checkid.asp?id=1;__SQL2INJECT__ HTTP/1.0
Host: [fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]
<...snipped for brevity...>
lhost = [fdf3:f0c0:2567:7fe4:800:27ff:fe00:0]
<...snipped for brevity...>

Results:

./sqlninja -m t
Sqlninja rel. 0.2.ff-svn <http://sqlninja.sf.net>
(C) 2006-2014 icesurfer & nico
[+] Parsing sqlninja.conf...
[-] host not defined in sqlninja.conf

Now, trying the following:

<...snipped for brevity...>
--httprequest_start--
GET http://[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]/checkid.asp?id=1;__SQL2INJECT__ HTTP/1.0
Host: fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
<...snipped for brevity...>
 # Local host: your IP address (for backscan and revshell modes)

[30] http://sqlninja.sourceforge.net/

lhost = [fdf3:f0c0:2567:7fe4:800:27ff:fe00:0]
<...snipped for brevity...>

We get again:

$ ./sqlninja -m t
Sqlninja rel. 0.2.ff-svn <http://sqlninja.sf.net>
(C) 2006-2014 icesurfer & nico
[+] Parsing sqlninja.conf...
[-] host not defined in sqlninja.conf

Same results if:

<...snipped for brevity...>
--httprequest_start--
GET http://[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]/checkid.asp?id=1;__SQL2INJECT__ HTTP/1.0
Host: [fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]
<...snipped for brevity...>
 # Local host: your IP address (for backscan and revshell modes)
lhost = fdf3:f0c0:2567:7fe4:800:27ff:fe00:0]
<...snipped for brevity...>

However, if you change the GET request like:

<...snipped for brevity...>
--httprequest_start--
GET http://fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa/checkid.asp?id=1;__SQL2INJECT__ HTTP/1.0
<...snipped for brevity...>

You get:

$ ./sqlninja -m t
Sqlninja rel. 0.2.ff-svn <http://sqlninja.sf.net>
(C) 2006-2014 icesurfer & nico
[+] Parsing sqlninja.conf...
[+] Loading extraction module: lib/getdata_time.pl
[+] Port 80. Assuming cleartext
[+] Target is: fdf3:80
[+] Checking that server is responding...
Error: could not create socket to fdf3:8

A note about perl and IPv6 support, according to: http://www.perl.org/about/whitepapers/perl-ipv6.html :

How to use both IPv4 and IPv6 networks from Perl:
To enable IPv6 in Perl, replace any use of IO::Socket::INET with IO::Socket::IP and you will be able to use both IPv4 and IPv6. Perl 5.14 has the full set of IPv6 functions as part of its core Socket module.

The Perl community is currently in the process of converting older modules to use this. If you use a module which has not been converted please report it to the module author.

The IO::Socket::IP class provides a general-purpose socket that can provide TCP connections or UDP packets using either IPv4 or IPv6. It is an API-compatible replacement for its IPv4-only predecessor, IO::Socket::INET.

Still not able to use IPv6 with sqlninja, although the aforementioned perl module is installed:

*# yum list installed perl*Socket**
Installed Packages

perl-IO-Socket-IP.noarch	*0.30-2.fc20*	*@updates*
perl-IO-Socket-SSL.noarch	*1.955-2.fc20*	*@updates*
perl-Socket.x86_64	*1:2.014-1.fc20*	*@updates*

My perl version is:

$perl --version
This is perl 5, version 18, subversion 2 (v5.18.2) built for x86_64-linux-thread-multi

11.6.1 Conclusion

Seems that sqlninja does not support IPv6.

11.7 w3af

w3af[31] is a Web Application Attack and Audit Framework. Tested version was 1.6.0.4, 19th August revision.

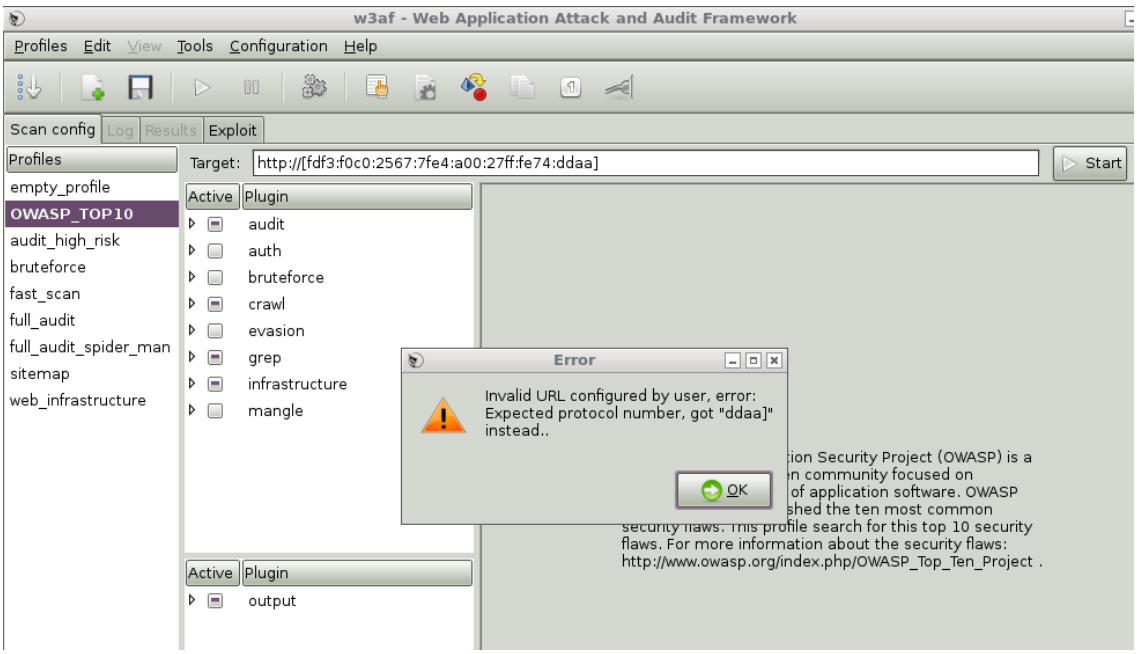

Figure 26 w3af Invalid URL Error

11.7.1 Conclusion

w3af does not support IPv6. This was actually confirmed implicitly when an e-mail was sent to the w3af mailing list.

11.8 Arachni

Arachni[32] is an Open Source, feature-full, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the security of web applications. Arachni has been tested with Kali on version 0.4.6.

[31] *http://w3af.org/*
[32] *http://www.arachni-scanner.com/*

Tests:

$arachni_web

Default Port: 9292

Default credentials

Administrator account:

E-mail: admin@admin.admin

Password: administrator

Regular user account

E-mail: user@user.user

Password: regular_user

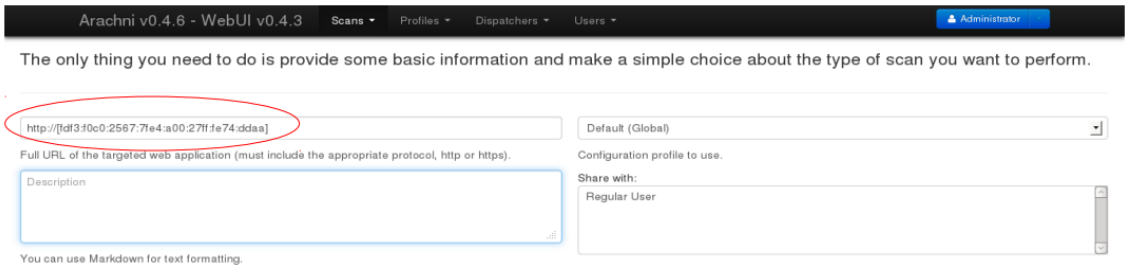

Figure 27 ArachniIPv6 URL

Results:

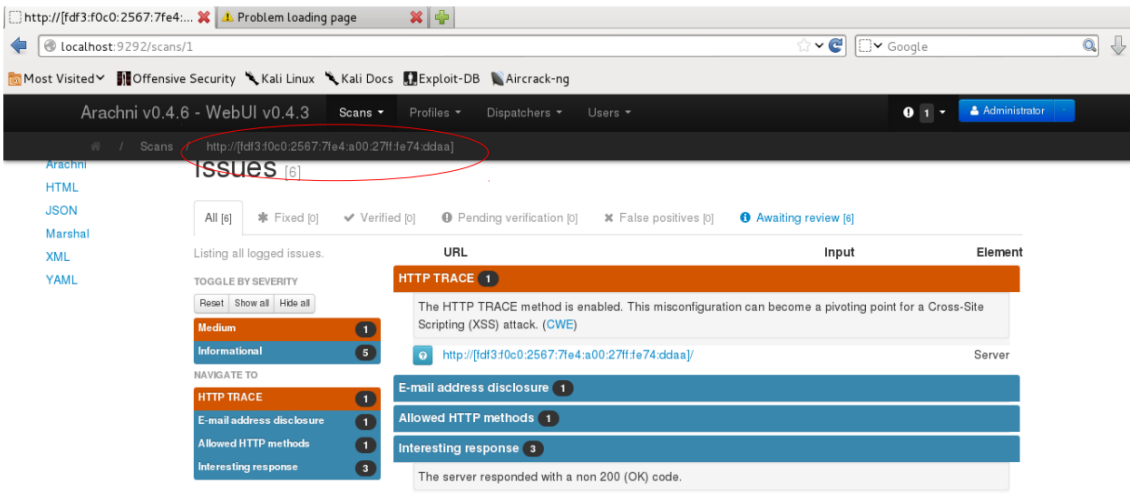

Figure 28 Arachni Issues

11.8.1 Conclusion

Arachni supports IPv6.

12 EXPLOITATION FRAMEWORKS

12.1 Metasploit

Metasploit[33], the famous penetration testing software was used with version 4.9.3-1 [core:4.9 api:1.0] and version 4.10.0-2014082003 [core:4.10.0.pre.2014082003 api:1.0.0].

Metasploit IPv6-related modules are given in Metasploit Modules. As we can see, several ipv6-related payloads are supported (tcp, http/https, perl, php, reverse or bind, etc.). Moreover, the following auxiliary modules are supported:

auxiliary/gather/dns_info	*normal DNS Basic Information Enumeration*
auxiliary/gather/dns_srv_enum	*normal DNS Common Service Record Enumeration*
auxiliary/scanner/discovery/ipv6_multicast_ping	*normal IPv6 Link Local/Node Local Ping Discovery*
auxiliary/scanner/discovery/ipv6_neighbor	*normal IPv6 Local Neighbor Discovery*
auxiliary/scanner/discovery/ipv6_neighbor_router_advertisement	*normal IPv6 Local Neighbor Discovery Using Router Advertisement*

Now, let's examine if any module (exploit, auxiliary, etc.) can be used against a target using IPv6 addresses:

Example 1:

- SSH users enumeration

```
msf > use auxiliary/scanner/ssh/ssh_enumusers
msf auxiliary(ssh_enumusers) > set RHOSTS fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
RHOSTS => fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
msf auxiliary(ssh_enumusers) > set USER_FILE /tmp/users.txt
USER_FILE => /tmp/users.txt
msf auxiliary(ssh_enumusers) > exploit
[*] fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 - SSH - Checking for false positives
[*]fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 - SSH - Starting scan
[!] fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 - SSH - User 'root' not found
[!] fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 - SSH - User 'atlas' not found
[!] fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa:22 - SSH - User 'test' not found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

The connections where verified by using Wireshark.

Example 2:

- IPv6_multicast_ping

Description:

It sends an ICMPv6 ping request to all default multicast addresses, and wait to see who responds.

```
msf > use auxiliary/scanner/discovery/ipv6_multicast_ping
msf auxiliary(ipv6_multicast_ping) > set interface vboxnet0
interface => vboxnet0
msf auxiliary(ipv6_multicast_ping) > exploit
[*] Sending multicast pings...
[*] Listening for responses...
[*]   |*| fe80::a00:27ff:fe74:ddaa => 08:00:27:74:dd:aa
[*]   |*| fe80::a00:27ff:fed1:d17a => 08:00:27:d1:d1:7a
[*] Auxiliary module execution completed
```

A Wireshark output if the captured packets is displayed below:

[33] *http://www.metasploit.com/*

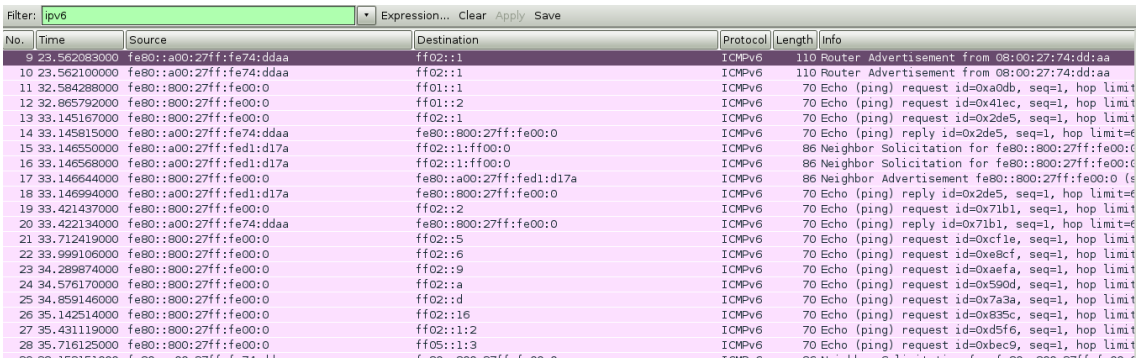

Figure 29 Wireshark Output Multicast Ping

Example 3:

- IPv6_neighbor

Description:

Enumerate local IPv6 hosts which respond to Neighbor Solicitations with a link-local address.

Basic options:

Name	Current Setting	Required	Description
INTERFACE		no	The name of the interface
PCAPFILE		no	The name of the PCAP capture file to process
RHOSTS		yes	The target address range or CIDR identifier
SHOST		no	Source IP Address
SMAC		no	Source MAC Address
THREADS	1	yes	The number of concurrent threads
TIMEOUT	500	yes	The number of seconds to wait for new data

msf auxiliary(ipv6_neighbor) > set RHOSTS fdf3:f0c0:2567:7fe4::/64

RHOSTS => fdf3:f0c0:2567:7fe4::/64

msf auxiliary(ipv6_neighbor) > exploit

[-] Auxiliary failed: Msf::OptionValidateError The following options failed to validate: RHOSTS.

msf auxiliary(ipv6_neighbor) > set RHOSTS fdf3:f0c0:2567:7fe4::0/64

RHOSTS => fdf3:f0c0:2567:7fe4::0/64

msf auxiliary(ipv6_neighbor) > exploit

[-] Auxiliary failed: Msf::OptionValidateError The following options failed to validate: RHOSTS.

msf auxiliary(ipv6_neighbor) > set RHOSTS fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa

RHOSTS => fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa

msf auxiliary(ipv6_neighbor) > exploit

[*] Discovering IPv4 nodes via ARP...

[-] Auxiliary failed: ArgumentError str is not a valid IPV4 address

[-] Call stack:

[-] /opt/metasploit/apps/pro/vendor/bundle/ruby/1.9.1/gems/packetfu-1.1.9/lib/packetfu/protos/ip/header.rb:48:in `read_quad'

[-] /opt/metasploit/apps/pro/vendor/bundle/ruby/1.9.1/gems/packetfu-1.1.9/lib/packetfu/protos/arp/header.rb:141:in `arp_daddr_ip='

[-] /opt/metasploit/apps/pro/vendor/bundle/ruby/1.9.1/gems/packetfu-1.1.9/lib/packetfu/protos/arp/mixin.rb:31:in `arp_daddr_ip='

[-] /opt/metasploit/apps/pro/msf3/modules/auxiliary/scanner/discovery/ipv6_neighbor.rb:175:in `buildprobe'

[-] /opt/metasploit/apps/pro/msf3/modules/auxiliary/scanner/discovery/ipv6_neighbor.rb:67:in `block in run_batch'

[-] /opt/metasploit/apps/pro/msf3/modules/auxiliary/scanner/discovery/ipv6_neighbor.rb:65:in `each'

[-] opt/metasploit/apps/pro/msf3/modules/auxiliary/scanner/discovery/ipv6_neighbor.rb:65:in `run_batch'

[-] /opt/metasploit/apps/pro/msf3/lib/msf/core/auxiliary/scanner.rb:174:in `block in run'

[-] /opt/metasploit/apps/pro/msf3/lib/msf/core/thread_manager.rb:100:in `call'

[-] /opt/metasploit/apps/pro/msf3/lib/msf/core/thread_manager.rb:100:in `block in spawn'

[*] Auxiliary module execution completed

So, it seems that we cannot use IPv6 scopes, although it is advertise that we can do.

Example 4:

- IPv6_neighbor_router_advertisement

Description:

- Send a spoofed router advertisement with high priority to force hosts to start the IPv6 address auto-config.
- Monitor for IPv6 host advertisements, and try to guess the link-local address by concatenating the prefix, and the host portion of the IPv6 address.
- Use NDP host solicitation to determine if the IP address is valid.

```
msf auxiliary(ipv6_neighbor) > use auxiliary/scanner/discovery/ipv6_neighbor_router_advertisement
set interface vboxnet0
interface => vboxnet0
msf auxiliary(ipv6_neighbor_router_advertisement) > explot
[-] Unknown command: explot.
msf auxiliary(ipv6_neighbor_router_advertisement) > exploit

[*] Sending router advertisement...
[*] Listening for neighbor solicitation...
[*]    |*| 2001:1234:dead:beef:a00:27ff:fe74:ddaa
[*]    |*| 2001:1234:dead:beef:a00:27ff:fed1:d17a
[*] Attempting to solicit link-local addresses...
[*]    |*| fe80::a00:27ff:fe74:ddaa -> 08:00:27:74:dd:aa
[*]    |*| fe80::a00:27ff:fed1:d17a -> 08:00:27:d1:d1:7a
[*] Auxiliary module execution completed
```

A sample Wireshark output of the aforementioned attack:

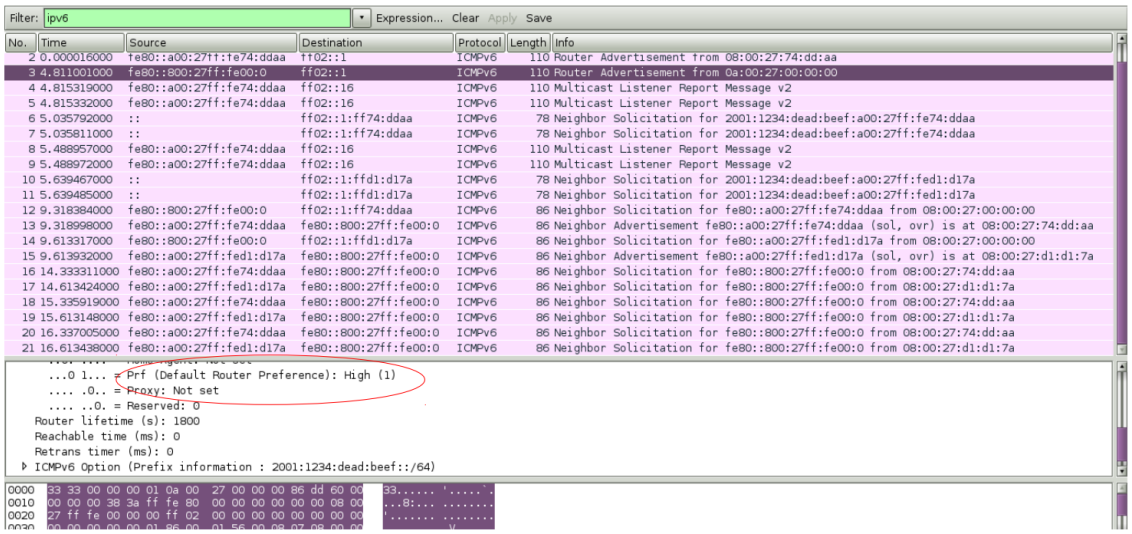

Figure 30 Wireshark Output IPv6 Neighbor Router Advertisement

It seems that this module works smoothly.

12.1.1 Conclusion

Metasploit can be used with IPv6 (against IPv6 targets, using IPv6-related payloads etc.) but, for the time being, it does not include many IPv6-specific auxiliary or exploit modules. Hence, it is recommended that during the discovery-phase other IPv6-specific tools, like the thc-ipv6 attack toolkit should be used. However, when you identify your IPv6 targets, you can use your favorite exploits against them via an IPv6 connection. To this end, several IPv6-specific payloads are supported (see Metasploit Modules).

13 When our Favorite Hacking Tool Does not Support IPv6

13.1 Fast and Easy

Probably one of the best solutions is to use socat[34]. A command line based utility that establishes two bidirectional byte streams and transfers data between them. Because the streams can be constructed from a large set of different types of data sinks and sources (see *address types*), and because lots of *address options* may be applied to the streams, socat can be used for many different purposes.

socat TCP-LISTEN:8080,reuseaddr,fork TCP6:[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]:80

Now, let's leverage Nikto (which does not support IPv6) against our target:

```
perl nikto.pl -host 127.0.0.1 -port 8080
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          127.0.0.1
+ Target Hostname:    localhost
+ Target Port:        8080
+ Start Time:         2014-09-01 16:07:56 (GMT3)
---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Fedora)
... <snipped for brevity> ...
+ End Time:           2014-09-01 16:08:19 (GMT3) (23 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

As we can see, Nikto runs and finished smoothly the scan of our IPv6 target via socat. If you need some advanced IPv6 usage, then you should use the IPv6-to-IPv4 Proxy of *Chiron* :)

[34] *http://www.dest-unreach.org/socat/doc/socat.html*

13.2 Exploiting IPv6 Features with your IPv4 Tools

Socat and other similar tools do not "exploit" the features and the capabilities of the IPv6 protocols, such as the IPv6 Extension Headers and/or fragmentation. Chiron proxy that comes bundled with Chiron[35] operates like a proxy between the IPv4 and the IPv6 protocol. It is not a common proxy like a web proxy, because it operates at layer 3. It accepts packets at a specific IPv4 address, extract the layer header and its payload, and sends them to a "target" using IPv6 but adding optionally one or more IPv6 Extension headers. So, chiron proxy is not useful only when IPv6 is not supported by your favorite ethical hacking tool, but, moreover, with tools that support IPv6 natively but you want to use some IPv6 features like the Extension Headers.

To use the tool, you must define, apart from the interface, at least the following parameters too:

- IPv4_sender, the IPv4 address of the software that sends the packet.
- IPv4_receiver, the IPv4 address where the proxy listens to

Of course, you must also define your IPv6 destination, as well as other generic parameters you may wish to include. The way that chiron proxy operates is displayed in the figure below:

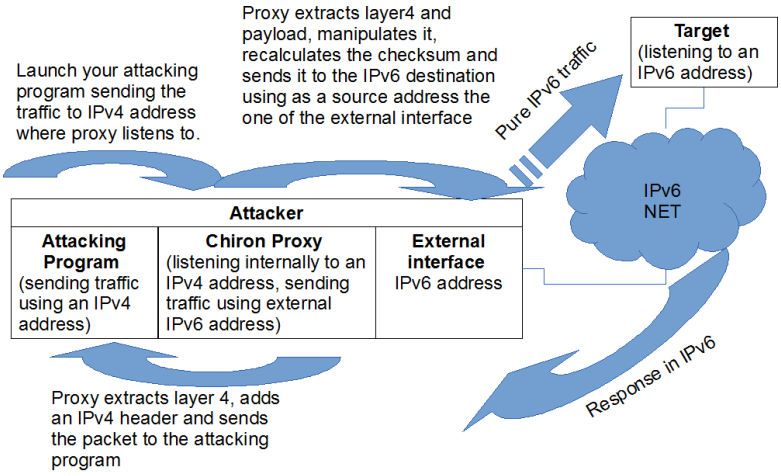

Figure 31 Chiron Workflow

The framework does not use the OS stack but its own library. When you send packets using the framework (e.g. a TCP SYN packet) and the other replies (SYN ACK in our example), your OS, which does not know anything about this, it will RESET (RST) the connection. To this end, you must temporarily configure your host firewall to drop such outgoing RST packets to the specific IPv6 destination.

For the time being, you have to do it on your own. In an updated version it will be configured automatically for you (at least for ip(6)tables and pf).

Example:

- You need to launch nikto against an IPv6-enabled web server.
- Your target's IPv6 address is fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa
- Your machine's IPv6 address is fdf3:f0c0:2567:7fe4:800:27ff:fe00:0

[35] *http://www.secfu.net/tools-scripts*

Step 1: Configure your firewall

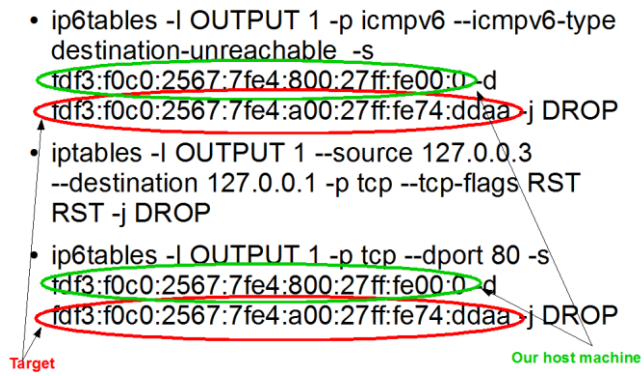

Figure 32 IP6 Tables Firewall Configuration

Step 2: Launch the Proxy:

./chiron_proxy.py vboxnet0 127.0.0.1 127.0.0.3 -d fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa -threads 10

Step 3: Run your program

perl nikto.pl -h http://127.0.0.3

It will take some extra time in comparison with direct communication, due to extra manipulation. It is the price that you have to "pay" using this tool. More information about Chiron can be found and the detailed tutorial that comes bundled with the source code at http://www.secfu.net/tools-scripts/

14 Conclusion

To summarize, we may draw the following conclusion:

- Information Gathering: Robtex, Shodan provide useful info.
- DNSRecon does the job for DNS.
- Dradis is fine for information gathering.
- Traceroute/traceroute6 work but not for advanced stuff (no Ext headers).
- Nmap - quite good support. Not ranges yet.
- Useful scripts - not something extraordinary, not everything works.
- Fingerprinting: p0f seems better than nmap - amap is always useful for fingerprinting services.
- Brute forcing - hydra is the only choice.
- Remote shells: Ncat works.
- Packet crafting: Scapy is your (only) friend.
- Nessus: Many IPv6-related plugins, not ranges or subnets.
- Web hacking: Burpsuite, sqlmap and Arachni work!
- For MITM, Ettercap supports IPv6 but not that useful yet.
- Net-snmp also supports IPv6.
- Metasploit: supports several IPv6 payloads, it does not include many IPv6-specific auxiliary or exploit modules.
- For the rest: Socat could be your friend. Chiron might even be better ;-)

15 APPENDIX

15.1 List of the Tested Tools

Tool	Remarks
Information Gathering / Collaboration	
Robtex	Robtex can be used for IPv6 reconnaissance purposes. It provides both IPv4 and IPv6 information for the targets.
Shodan	Some information can be obtained regarding IPv6, but this should be further examined and analyzed. Certainly, digging further is required from the analyst's perspective.
Maltego	As far as IPv6 is concerned, it seems that Maltego is not an option.
Dradis	It can be used for information collaboration in the IPv6 era.
Reconnaissance	
Fierce	IPv6 does not seem to be supported by fierce.
DNSrecon	IPv6 is supported and related info is provided. A useful tool for the IPv6 era.
Tcptraceroute	Although IPv6 is advertised, it doesn't seem to work.
Traceroute6	IPv6 is supported. No advanced features though (i.e. support of IPv6 Extension headers).
Firewalk	It does not support IPv6.
Network (port) Scanning	

Unicornscan	It does not support IPv6
Nmap	Quite good support of port scanning in IPv6 using nmap. The most significant drawback is the fact that it does not support a range of IPv6 addresses, as well as a comma-separated list of addresses. The latest can be handled by using an input file.
Fingerprinting	
Nmap	Although IPv6 fingerprinting is supported under IPv6, it is not that effective yet.
xprobe2	It does not support IPv6.
p0f	It recognizes IPv6 traffic. It seems to recognize Linux (as Linux 2.2.x-3.x) and Windows hosts (as "Windows 7 or 8"), but not BSD ones. More testing on this field is required though using normal traffic, but, definitely, IPv6 is fingerprinting is supported. The only question is how effective it can be.
Amap	There shouldn't be any problem by using amap/amap6 with IPv6 when you use as an input just a single address. Its detection performance does not depend on layer-3 and hence, it should be the same as using IPv4. However, its creator, Marc Heuse, recommend amap just for UDP IPv6 scam only. Otherwise, considered outdated... Moreover, when you try to read the addresses/ports from an nmap machine readable output file (produced using nmap -oM), this is not performed properly and the service fingerprint fails.
Brute-forcing	
Hydra	It partially supports IPv6. You can define a single IPv6 target using -6, but you cannot define a list of targets in a file using -M. No options for adding IPv6 Extension headers or other IPv6-related capabilities (e.g. for evading purposes).
Medusa	It does not support IPv6.
Ncrack	Although it is claimed to be supported, at least experimentally, IPv6 does not seem to work.
Packet Crafting	

Hping	It does not support IPv6.
Nping	It offers very limited IPv6 functionality. For arbitrary IPv6 packet crafting, use either Scapy or Chiron :-)
Scapy	Very good support of IPv6, not all the latest IPv6 Extension headers or protocols though (e.g. MLDv2).
Remote Shells	
Ncat	It works without problem using IPv6. It also supports some handy features, like ssl encryption, even using over IPv6.
LAN / MITM Attacks & other	
Nmap NSE scripts	Several NSE scripts either support IPv6 or they are IPv6-specific ones. Some of them do not appear to work properly. From the rest, the most interesting/effective ones are the following: ■ IPv6-ra-flood (quite effective even against the latest OS). ■ Targets-ipv6-multicast-invalid-dst (this produces similar results to alive6 of the thc-ipv6 attacking toolkit). ■ Targets-ipv6-multicast-echo ■ IPv6-node-info ■ Resolveall
Ettercap	Although ettercap supports IPv6 addresses, critical modules (like MITM attacks) do not seem to be implemented yet.
Cain & Abel	It does not support IPv6.
Net-snmp	IPv6 is supported by Net-snmp, but you must compile it with this option enabled (it is not by default).
Vulnerability Scanning	
Nessus	It supports IPv6 addresses as targets, but NOT using IPv6 prefixes or IPv6 ranges. The IPv6 host discovery module does not use many methods to discovery IPv6 hosts (e.g. IPv6 datagrams with erroneous parameters or extension headers, etc.). However, it incorporates several IPv6-related vulnerability discovery plugins. To sum-up, Nessus can be used against IPv6 networks but it is recommended that during the discovery phase more specialized tools, like the thc-ipv6 attack

	toolkit, should be used.
Web Penetration Testing	
Nikto	It does not support IPv6.
Skipfish	It does not support IPv6.
Zaproxy	It does not support IPv6.
Burpsuite	Burpsuite can be used as a web proxy and can spider IPv6 targets smoothly!
Arachni	It supports IPv6.
Sqlmap	It supports IPv6.
Sqlninja	It does not support IPv6.
W3af	It does not support IPv6.
Exploitation Frameworks	
Metasploit	It can be used with IPv6 (against IPv6 targets, using IPv6-related payloads etc.) but, for the time being, it does not include many IPv6-specific auxiliary or exploit modules. Hence, it is recommended that during the discovery-phase other IPv6-specific tools, like the thc-ipv6 attack toolkit should be used. However, when you identify your IPv6 targets, you can use your favorite exploits against them via an IPv6 connection. To this end, several IPv6-specific payloads are supported (see Appendix 15.2: Metasploit modules).

Table 1 List of Tested Tools

15.2 Metasploit Modules

msf > search IPv6
[!] Database not connected or cache not built, using slow search

Matching Modules
==========

Name	Disclosure Date	Rank	Description
auxiliary/gather/dns_info		normal	DNS Basic Information Enumeration
auxiliary/gather/dns_srv_enum		normal	DNS Common Service Record Enumeration
auxiliary/scanner/discovery/ipv6_multicast_ping		normal	IPv6 Link Local/Node Local Ping Discovery
auxiliary/scanner/discovery/ipv6_neighbor		normal	IPv6 Local Neighbor Discovery
auxiliary/scanner/discovery/ipv6_neighbor_router_advertisement		normal	IPv6 Local Neighbor Discovery Using Router Advertisement
payload/bsd/x86/shell/bind_ipv6_tcp		normal	BSD Command Shell, Bind TCP Stager (IPv6)
payload/bsd/x86/shell/reverse_ipv6_tcp		normal	BSD Command Shell, Reverse TCP Stager (IPv6)
payload/bsd/x86/shell_bind_tcp_ipv6		normal	BSD Command Shell, Bind TCP Inline (IPv6)
payload/bsd/x86/shell_reverse_tcp_ipv6		normal	BSD Command Shell, Reverse TCP Inline (IPv6)
payload/cmd/unix/bind_netcat_gaping_ipv6		normal	Unix Command Shell, Bind TCP (via netcat -e) IPv6
payload/cmd/unix/bind_perl_ipv6		normal	Unix Command Shell, Bind TCP (via perl) IPv6
payload/cmd/unix/bind_ruby_ipv6		normal	Unix Command Shell, Bind TCP (via Ruby) IPv6
payload/cmd/windows/bind_perl_ipv6		normal	Windows Command Shell, Bind TCP (via perl) IPv6
payload/linux/x86/meterpreter/bind_ipv6_tcp		normal	Linux Meterpreter, Bind TCP Stager (IPv6)
payload/linux/x86/meterpreter/reverse_ipv6_tcp		normal	Linux Meterpreter, Reverse TCP Stager (IPv6)
payload/linux/x86/shell/bind_ipv6_tcp		normal	Linux Command Shell, Bind TCP Stager (IPv6)
payload/linux/x86/shell/reverse_ipv6_tcp		normal	Linux Command Shell, Reverse TCP Stager (IPv6)
payload/linux/x86/shell_bind_ipv6_tcp		normal	Linux Command Shell, Bind TCP Inline (IPv6)
payload/php/bind_perl_ipv6		normal	PHP Command Shell, Bind TCP (via perl) IPv6
payload/php/bind_php_ipv6		normal	PHP Command Shell, Bind TCP (via php) IPv6
payload/php/meterpreter/bind_tcp_ipv6		normal	PHP Meterpreter, Bind TCP Stager IPv6
payload/ruby/shell_bind_tcp_ipv6		normal	Ruby Command Shell, Bind TCP IPv6
payload/windows/dllinject/bind_ipv6_tcp		normal	Reflective DLL Injection, Bind TCP Stager (IPv6)
payload/windows/dllinject/reverse_ipv6_tcp		normal	Reflective DLL Injection, Reverse TCP Stager (IPv6)
payload/windows/meterpreter/bind_ipv6_tcp		normal	Windows Meterpreter (Reflective Injection), Bind TCP Stager (IPv6)
payload/windows/meterpreter/reverse_ipv6_tcp		normal	Windows Meterpreter (Reflective Injection), Reverse TCP Stager (IPv6)
payload/windows/patchupdllinject/bind_ipv6_tcp		normal	Windows Inject DLL, Bind TCP Stager (IPv6)
payload/windows/patchupdllinject/reverse_ipv6_tcp		normal	Windows Inject DLL, Reverse TCP Stager (IPv6)
payload/windows/patchupmeterpreter/bind_ipv6_tcp		normal	Windows Meterpreter (skape/jt Injection), Bind TCP Stager (IPv6)
payload/windows/patchupmeterpreter/reverse_ipv6_tcp		normal	Windows Meterpreter (skape/jt Injection), Reverse TCP Stager (IPv6)
payload/windows/shell/bind_ipv6_tcp		normal	Windows Command Shell, Bind TCP Stager (IPv6)
payload/windows/shell/reverse_ipv6_tcp		normal	Windows Command Shell, Reverse TCP Stager (IPv6)
payload/windows/upexec/bind_ipv6_tcp		normal	Windows Upload/Execute, Bind TCP Stager (IPv6)
payload/windows/upexec/reverse_ipv6_tcp		normal	Windows Upload/Execute, Reverse TCP Stager (IPv6)
payload/windows/vncinject/bind_ipv6_tcp		normal	VNC Server (Reflective Injection), Bind TCP Stager (IPv6)
payload/windows/vncinject/reverse_ipv6_tcp		normal	VNC Server (Reflective Injection), Reverse TCP Stager (IPv6)
post/multi/gather/resolve_hosts		normal	Multi Gather Resolve Hosts
post/windows/manage/portproxy		normal	Windows Manage Set Port Forwarding With PortProxy

15.3 References

[1] https://isc.sans.edu/forums/diary/Are+your+tools+ready+for+IPv6+part+2/11416

15.4 Disclaimer

All products, company names, brand names, trademarks and logos are the property of their respective owners.