Penetration Testing Tools that (do not) Support IPv6
# 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  TRACEROUTING - 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  PDF ......................................................................................................................... 20
5.3.1 Conclusion ........................................................................................................... 20
5.4  AMP ......................................................................................................................... 20
5.4.1 Conclusion ........................................................................................................... 21
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Brute-Forcing</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.1 HYDRA</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.1.1 Conclusion</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>6.2 MEDUSA</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>6.2.1 Conclusion</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>6.3 NCRACK</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>6.3.1 Conclusion</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>Packet Crafting</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>7.1 HPING</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>7.1.1 Conclusion</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>7.2 NPING</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>7.2.1 Conclusion</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>7.3 SCAPY</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>7.3.1 Conclusion</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>Remote Shells</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>8.1 Ncat</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>8.1.1 Conclusion</td>
<td>27</td>
</tr>
<tr>
<td>9</td>
<td>LAN / MITM Attacks &amp; Other</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>9.1 NSE Scripts</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>9.1.1 Conclusion</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>9.2 ETTERCAP</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>9.2.1 Conclusion</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>9.3 CAIN &amp; ABEL</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>9.3.1 Conclusion</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>9.4 NET-SNMP</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>9.4.1 Conclusion</td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>Vulnerability Scanners</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>10.1 Nessus</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>10.1.1 Conclusion</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>Web Penetration Testing</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>11.1 BURP SUITE</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>11.1.1 Conclusion</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>11.2 ZAPROXY</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>11.2.1 Conclusion</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>11.3 NIKTO</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>11.3.1 Conclusion</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>11.4 SKIPFISH</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>11.4.1 Conclusion</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>11.5 SQLMAP</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>11.5.1 Conclusion</td>
<td>44</td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Figure 1 Robtex Output &quot;ernw.de&quot;</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Figure 2 IPv6 Search Results Shodan</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Figure 3 Maltego Target <a href="http://www.ernw.de">www.ernw.de</a></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Figure 4 Dradis Nmap Output</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Figure 5 Ncat Unencrypted Communication – Example 1</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Figure 6 Ncat Unencrypted Communication – Example 2</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Figure 7 RFC 4620</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Figure 8 Wireshark Output nmap scan</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Figure 9 Flood router impact on a Windows 7 Operating System</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Figure 10 Wireshark Output targets-ipv6-multicast-invalid-dst</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Figure 11 Targets-ipv6-multicast-mld</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Figure 12 IPv6 Hosts List Ettercap</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>Figure 13 Wireshark Output SNMP Walk</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Figure 14 Nessus IPv6 Targets</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Figure 15 Wireshark Output ICMPv6 Echo Request</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>Figure 16 Nessus IPv6 Host Discovery</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>Figure 17 Detailed Nessus Output IPv6 Host Discovery</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>Figure 18 Nessus Invalid Target Settings</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>Figure 19 Advanced Search Nessus</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Figure 20 Nessus Plugins</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Figure 21 Nessus Enumeration IPv6 Interfaces via SSH</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Figure 22 Burp Suite IPv6 Targets</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Figure 23 Zaproy IPv6 URL To Attack</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Figure 24 Zaproy Invalid Address Proxy Error</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Figure 25 Wireshark Output SqlMap</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Figure 26 w3af Invalid URL Error</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>Figure 27 ArachniIPv6 URL</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>Figure 28 Arachni Issues</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>Figure 29 Wireshark Output Multicast Ping</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>Figure 30 Wireshark Output IPv6 Neighbor Router Advertisement</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Figure 31 Chiron Workflow</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>Figure 32 IP6 Tables Firewall Configuration</td>
<td>54</td>
</tr>
</tbody>
</table>
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-IPV6¹
- SI6 Networks’ IPv6 Toolkit²
- Chiron³

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):

<table>
<thead>
<tr>
<th>Operating System</th>
<th>IPv6 Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora 20</td>
<td>fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa</td>
</tr>
<tr>
<td>Centos 6.5</td>
<td>fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a</td>
</tr>
<tr>
<td>OpenBSD 5.5</td>
<td>fdf3:f0c0:2567:7fe4:a00:27ff:fe6a:ca6a</td>
</tr>
<tr>
<td>FreeBSD 10</td>
<td>fdf3:f0c0:2567:7fe4:a00:27ff:fefc:f99a</td>
</tr>
<tr>
<td>Windows 7.1</td>
<td>fdf3:f0c0:2567:7fe4:c06:5389:f6e:99c0</td>
</tr>
</tbody>
</table>

¹ https://www.thc.org/thc-ipv6/
² http://www.si6networks.com/tools/ipv6toolkit/
³ http://www.secfu.net/tools-scripts/
Fedora 20 host also served as an IPv6 router by using radvd4.

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/rdvd/
2 INFORMATION GATHERING

2.1 Robtex

Let’s start by trying to check if Robtex\(^2\) 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).

![Figure 1 Robtex Output "ernw.de"](image)

**2.1.1 Conclusion**

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

---

\(^2\) https://www.robtex.com
2.2 Shodan

Now, let’s continue with Shodan\(^4\) for searching about IPv6-related findings. An example output is displayed below:

![Shodan IPv6 Search Results](image)

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.

\(^4\) 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:

![Maltego Output](image)

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.

![Dradis Output](image)

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  

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:
s2.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:

[*] Reverse Look-up of a Range
[*] 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.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 tracering.

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 / Traceroote

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-ltn-kln-15L500.forthnet.gr (2a02:2148:77:50:2::5) 36.209 ms 39.093 ms 39.939 ms
3 2001:4860:1:1::4d9:0:1 (2001:4860:1:1::4d9:0:1) 68.665 ms 50.945 ms 51.564 ms
5 2a02:2148:2:6::22 (2a02:2148:2:6::22) 56.300 ms 61.197 ms 61.938 ms
6 2001:4860:0:1::617 (2001:4860:0:1::617) 53.609 ms 50.496 ms 51.813 ms
7 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-ltn-kln-12L500.forthnet.gr (2a02:2148:77:50:2::17) 40.596 ms 40.582 ms 41.146 ms
3 te0-1-0-distr-kln-01.forthnet.gr (2a02:2148:2:65::22) 42.024 ms 42.799 ms 43.429 ms
4 BE-2.core-kln-2.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
7 2001:4860:0:1::617 (2001:4860:0:1::617) 50.795 ms 46.540 ms 51.746 ms
8 sof01-s01-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\(^2\) 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::1

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.

\(^2\) 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 -l 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 \(\text{license}\) 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/](http://www.unicornscan.org/)

\(^9\) [http://nmap.org/](http://nmap.org/)
**Conclusion**

Regarding network scanning:

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

---

10The 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 SP2 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 fd03:f0c0:2567:7fe4:a00:27ff:fe74:ddaa

Xprobe2 v0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, meder@o0o.nu
5.2.1 Conclusion
IPv6 is not supported by Xprobe2.

5.3 p0f
p0f\(^1\), 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 -fp p0f.fp -r nmap-6-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:fo:0:2567:7fe4: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
Total amount of tasks to perform in plain connect mode: 23
Waiting for timeout on 23 connections ...
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.

\(^1\) [http://lcamtuf.coredump.cx/p0f3](http://lcamtuf.coredump.cx/p0f3)
\(^2\) [https://www.thc.org/thc-amap](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 
[::fddf::f0c0:2567:7f4:a00:27ff:fe74:ddaa]:22/tcp, disabling port [EUNKNI]).

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\(^2\) 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.

[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

```
```

1 of 1 target successfully completed, 1 valid password found


- 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.

[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use `-t 4`

[DATA] max 6 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!
```


\(^2\) Website: https://www.thc.org/thc-hydra
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\textsuperscript{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:fe4:0002:7ff:fe74:ddaa -u root -p pass -M ssh
```

Medusa v2.1.1 [http://www.foofus.net] (C) JoMo-Kun / Foofas Networks <jmk@foofus.net>

**NOTICE:** ssh.mod: failed to connect, port 22 was not open on fdf3:f0c0:2567:fe4:0002:7ff:fe74:ddaa

6.2.1 Conclusion

IPv6 is not supported by medusa.

6.3 Ncrack

Ncrack\textsuperscript{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:7501:4819:2000:2107:ff: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.

\textsuperscript{14} Website: http://foofus.net/goons/jmk/medusa/medusa.html

\textsuperscript{15} http://nmap.org/ncrack/
7 Packet Crafting

7.1 Hping

Hping\(^6\) assembles and sends custom ICMP, UDP, or TCP packets and then displays any replies. The tested version was 20051105-20.

```
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\(^7\) 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.

\(^6\) [http://sectools.org/tool/hping/](http://sectools.org/tool/hping/)

\(^7\) [http://nmap.org/nping/](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\textsuperscript{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].

\textsuperscript{18} http://www.secdev.org/projects/scapy/
8 REMOTE SHELLS

8.1 Ncat

Ncat\(^1\) 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 fdf3:00c0:2567:7fe4:800:27ff:fe00:0`

Default port: `31237`

It provides you a typical non-interactive shell.

Transferring a file:

$ `ncat -l -6 > test.txt`

$ `ncat -6 fdf3:00c0:2567:7fe4:800:27ff: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`).

\(^1\) http://nmap.org/ncat/
Another example output of Ncat encrypted communication is displayed below:

![Ncat encrypted communication example](image)

**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)\(^\text{20}\) 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).

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].

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.

There are three useful types of queries:

qtype=2: Node Name
qtype=3: Node Addresses
qtype=4: IPv4 Addresses

Command example:

nmap -6 fdf3:f0c0:2567:7f0e:a00:27ff:fe1: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].
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 8 Wireshark Output nmap scan**

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>255.255.255.255/32</td>
<td>127.0.0.1</td>
<td>ICMPv6</td>
<td>28</td>
<td>unreachable</td>
</tr>
<tr>
<td>255.255.255.255/32</td>
<td>127.0.0.1</td>
<td>ICMPv6</td>
<td>28</td>
<td>unreachable</td>
</tr>
<tr>
<td>255.255.255.255/32</td>
<td>127.0.0.1</td>
<td>ICMPv6</td>
<td>28</td>
<td>unreachable</td>
</tr>
<tr>
<td>255.255.255.255/32</td>
<td>127.0.0.1</td>
<td>ICMPv6</td>
<td>28</td>
<td>unreachable</td>
</tr>
<tr>
<td>255.255.255.255/32</td>
<td>127.0.0.1</td>
<td>ICMPv6</td>
<td>28</td>
<td>unreachable</td>
</tr>
</tbody>
</table>

**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'  
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  

Four systems responded (Fedora, CentOS, FreeBSD 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:

![Packet Example]

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 //fddf3:f0c0:2567:7fe4:a00:27ff:fe74:dda// //fddf3:f0c0:2567:7fe4:a00:27ff:fe00:0d17a/ 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
fddf3: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 ee_sslstrip.so cannot be loaded...
32 plugins
42 protocol dissectors
57 ports monitored
16074 mac vendor fingerprint
766 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 a 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) fddf3:f0c0:2567:7fe4:a00:27ff:fe74:dda 08:00:27:74:DD::AA
2) fddf3:f0c0:2567:7fe4:a00:27ff:fe00:0d17a 08:00:27:D1:D1:7A
3) fe80::a00:27ff:fe74:dda 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/](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 fdff:0c0:2567:7e4: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](http://www.oxid.it/cain.html)

\(^{24}\) [http://www.net-snmp.org/](http://www.net-snmp.org/)
10 **VULNERABILITY 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](image-url)
Let's see, using a Wireshark output, the methods by Nessus for IPv6 discovery:

![Figure 15 Wireshark Output ICMPv6 Echo Request](image)

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:**

**IPv6_host_discovery**

<table>
<thead>
<tr>
<th>Score</th>
<th>Host</th>
<th>Vulnerabilities</th>
<th>Notes</th>
<th>Hide Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>192.168.2.256:1900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>192.168.2.256:1900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>192.168.2.256:1900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>192.168.2.256:1900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>192.168.2.256:1900</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 16 Nessus IPv6 Host Discovery](image)

As we can see, all targets were identified.
Some more detailed results are displayed below:

**Output**

<table>
<thead>
<tr>
<th>Port 135/tcp was found to be open</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
</tr>
<tr>
<td>135/tcp/arp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port 445/tcp was found to be open</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port 554/tcp was found to be open</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
</tr>
<tr>
<td>554/tcp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port 2809/tcp was found to be open</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
</tr>
<tr>
<td>2809/tcp</td>
</tr>
</tbody>
</table>

*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. fdff: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 fdff:f0c0:2567:7fe4::/64
- Ranges of addresses are defined, e.g. fdff:f0c0:2567:7fe4:a00:27ff:fe74:dd00 – fdff:f0c0:2567:7fe4:a00:27ff:fe74:ddff or fdff: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.

We get a plenty of IPv6-related plugins.

And also some nice recommendations.

Enumerate IPv6 Interfaces via SSH

**Synopsis**
This plugin enumerates IPv6 interfaces on a remote host.

**Description**
By connecting to the remote host via SSH with the supplied credentials, this plugin enumerates network interfaces configured with IPv6 addresses.

**Solution**
Enable IPv6 if you are not already using it. Otherwise, disable any unused IPv6 interfaces.

**Plugin Information**
- Plugin ID: 32202
- Plugin Type: local
- Plugin Published Date: 2027/09/11
- Plugin Last Modification Date: 2021/09/11
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](image)

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 Zaproy IPv6 URL To Attack

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

The address isn't valid

The URL is not valid and cannot be loaded.

- Web addresses are usually written like http://www.example.com/
- Make sure that you're using forward slashes (i.e. /).

Figure 24 Zaproy 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\textsuperscript{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\textsuperscript{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] [INFO] connection success
[14:28:35] [INFO] URL: http://[fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa]
[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] [INFO] testing if the target URL is stable
[14:28:37] [INFO] target URL is stable
[14:28:38] [WARNING] no parameter(s) found for testing in the provided data (e.g. GET parameter 'id' in 'www.site/index.php?id=1')
[14:28:37] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 2 times

\textsuperscript{28} https://code.google.com/p/skipfish

\textsuperscript{29} sqlmap/1.0-dev-nongit-20140823
11.5.1 Conclusion
IPv6 is supported by sqlmap.

11.6 sqlninja
Sqlninja\(^3\) is a SQL server injecting and takeover tool. The version tested was the latest dev version download in 23\(^{rd}\) August 2014 via svn.

The following sqlninja configurations were tested:

```
<...snipped for brevity...>

HTTP request 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...>
```

Results:
```
./sqlninja -mt
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...>

HTTP request 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)

\(^3\) http://sqlninja.sourceforge.net/
We get again:

```bash
$ ./sqlninja
```

```
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:

```bash
<...snipped for brevity...>
```

```bash
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)
```bash
lhost = [fdf3:f0c0:2567:7fe4:800:27ff:fe00:0]
```

However, if you change the GET request like:

```bash
<...snipped for brevity...>
```

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

You get:

```bash
$ ./sqlninja
```

```
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: ffd3:80
[+] Checking that server is responding...
```

```bash
Error: could not create socket to ffd3:8
```

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:

```bash
# 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:

```bash
$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, 19\(^{th}\) August revision.

![w3af Interface](image)

*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/](http://w3af.org/)

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\(^{23}\), 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:d7a => 08:00:27:d1:d7:a
[*] Auxiliary module execution completed
```

A Wireshark output if the captured packets is displayed below:

\(^{23}\) 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:

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

```
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/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
msf auxiliary(ipv6_neighbor_router_advertisement) > exploit
[-] Unknown command: exploit.
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...
[*] 08:00:27:74:dd:aa
[*] 08:00:27:d1:d1:7a
[*] Auxiliary module execution completed
```
A sample Wireshark output of the aforementioned attack:

![Wireshark Output](image)

**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 tool, 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. 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, reuseaddrs, fork TCP6:[ff3::f0c0:2567:7f:fe:74:70]:80
```

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

```
perl nikto.pl -host 127.0.0.1 -port 8080

+ 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 :)
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\(^{25}\) 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\(_{\text{sender}}\), the IPv4 address of the software that sends the packet.
- IPv4\(_{\text{receiver}}\), 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](http://www.secfu.net/tools-scripts)

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

\(^{25}\) [http://www.secfu.net/tools-scripts](http://www.secfu.net/tools-scripts)
Step 1: Configure your firewall

```
• ip6tables -I OUTPUT 1 -p icmpv6 --icmpv6-type
destination-unreachable -s
  fdf3:f0c0:2567:7fe4:800:27ff:fe00:d
• ip6tables -I OUTPUT 1 --source 127.0.0.3
  --destination 127.0.0.1 -p tcp --tcp-flags RST
  RST -j DROP
• ip6tables -I OUTPUT 1 -p tcp --dport 80 -s
  fdf3:f0c0:2567:7fe4:800:27ff:fe00:d
  -j DROP
```

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 ;-}
# Appendix

## 15.1 List of the Tested Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Information Gathering / Collaboration</strong></td>
<td></td>
</tr>
<tr>
<td>Robtex</td>
<td>Robtex can be used for IPv6 reconnaissance purposes. It provides both IPv4 and IPv6 information for the targets.</td>
</tr>
<tr>
<td>Shodan</td>
<td>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.</td>
</tr>
<tr>
<td>Maltego</td>
<td>As far as IPv6 is concerned, it seems that Maltego is not an option.</td>
</tr>
<tr>
<td>Dradis</td>
<td>It can be used for information collaboration in the IPv6 era.</td>
</tr>
<tr>
<td><strong>Reconnaissance</strong></td>
<td></td>
</tr>
<tr>
<td>Fierce</td>
<td>IPv6 does not seem to be supported by fierce.</td>
</tr>
<tr>
<td>DNSrecon</td>
<td>IPv6 is supported and related info is provided. A useful tool for the IPv6 era.</td>
</tr>
<tr>
<td>Tcptraceroute</td>
<td>Although IPv6 is advertised, it doesn’t seem to work.</td>
</tr>
<tr>
<td>Traceroute6</td>
<td>IPv6 is supported. No advanced features though [i.e. support of IPv6 Extension headers].</td>
</tr>
<tr>
<td>Firewalk</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td><strong>Network (port) Scanning</strong></td>
<td></td>
</tr>
<tr>
<td>Tool</td>
<td>IPv6 Support</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Unicornscan</td>
<td>It does not support IPv6</td>
</tr>
<tr>
<td>Nmap</td>
<td>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.</td>
</tr>
<tr>
<td>Fingerprinting</td>
<td></td>
</tr>
<tr>
<td>Nmap</td>
<td>Although IPv6 fingerprinting is supported under IPv6, it is not that effective yet.</td>
</tr>
<tr>
<td>xprobe2</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td>p0f</td>
<td>It recognizes IPv6 traffic. It seems to recognize Linux (as Linux 2.2.x-3.x) and Windows hosts (as &quot;Windows 7 or 8&quot;), 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.</td>
</tr>
<tr>
<td>Amap</td>
<td>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.</td>
</tr>
<tr>
<td>Brute-forcing</td>
<td></td>
</tr>
<tr>
<td>Hydra</td>
<td>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).</td>
</tr>
<tr>
<td>Medusa</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td>Ncrack</td>
<td>Although it is claimed to be supported, at least experimentally, IPv6 does not seem to work.</td>
</tr>
<tr>
<td>Packet Crafting</td>
<td></td>
</tr>
<tr>
<td>Tool</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Hping</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td>Nping</td>
<td>It offers very limited IPv6 functionality. For arbitrary IPv6 packet crafting, use either Scapy or Chiron.</td>
</tr>
<tr>
<td>Scapy</td>
<td>Very good support of IPv6, not all the latest IPv6 Extension headers or protocols though (e.g. MLDv2).</td>
</tr>
<tr>
<td>Remote Shells</td>
<td></td>
</tr>
<tr>
<td>Ncat</td>
<td>It works without problem using IPv6. It also supports some handy features, like ssl encryption, even using over IPv6.</td>
</tr>
<tr>
<td>LAN / MITM Attacks &amp; other</td>
<td></td>
</tr>
<tr>
<td>Nmap NSE scripts</td>
<td>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:</td>
</tr>
<tr>
<td>Ettercap</td>
<td>Although ettercap supports IPv6 addresses, critical modules (like MITM attacks) do not seem to be implemented yet.</td>
</tr>
<tr>
<td>Cain &amp; Abel</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td>Net-snmp</td>
<td>IPv6 is supported by Net-snmp, but you must compile it with this option enabled (it is not by default).</td>
</tr>
<tr>
<td>Vulnerability Scanning</td>
<td></td>
</tr>
</tbody>
</table>
| 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
Web Penetration Testing

<table>
<thead>
<tr>
<th>Tool</th>
<th>Support IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nikto</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td>Skipfish</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td>Zaproxy</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td>Burpsuite</td>
<td>Burpsuite can be used as a web proxy and can spider IPv6 targets smoothly!</td>
</tr>
<tr>
<td>Arachni</td>
<td>It supports IPv6.</td>
</tr>
<tr>
<td>Sqlmap</td>
<td>It supports IPv6.</td>
</tr>
<tr>
<td>Sqlninja</td>
<td>It does not support IPv6.</td>
</tr>
<tr>
<td>W3af</td>
<td>It does not support IPv6.</td>
</tr>
</tbody>
</table>

Exploitation Frameworks

<table>
<thead>
<tr>
<th>Tool</th>
<th>Support IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metasploit</td>
<td>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].</td>
</tr>
</tbody>
</table>

Table 1 List of Tested Tools
15.2 Metasploit Modules

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

Matching Modules

<table>
<thead>
<tr>
<th>Name</th>
<th>Disclosure Date</th>
<th>Rank</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>auxiliary/gather/dns_info</code></td>
<td>normal</td>
<td>DNS Basic Information Enumeration</td>
<td></td>
</tr>
<tr>
<td><code>auxiliary/gather/dns_srv_enum</code></td>
<td>normal</td>
<td>DNS Common Service Record Enumeration</td>
<td></td>
</tr>
<tr>
<td><code>auxiliary/scanner/discovery/ipv6_multicast_ping</code></td>
<td>normal</td>
<td>IPv6 Link Local/Node Local Ping Discovery</td>
<td></td>
</tr>
<tr>
<td><code>auxiliary/scanner/discovery/ipv6_neighbor</code></td>
<td>normal</td>
<td>IPv6 Local Neighbor Discovery</td>
<td></td>
</tr>
<tr>
<td><code>auxiliary/scanner/discovery/ipv6_neighbor_router_advertisement</code></td>
<td>normal</td>
<td>IPv6 Local Neighbor Discovery Using Router Advertisement</td>
<td></td>
</tr>
<tr>
<td><code>payload/bbsd86/shell/bind_ipv6_tcp</code></td>
<td>normal</td>
<td>BSD Command Shell, Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/bbsd86/shell/reverse_ipv6_tcp</code></td>
<td>normal</td>
<td>BSD Command Shell, Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/bbsd86/shell_bind_tcp_ipv6</code></td>
<td>normal</td>
<td>BSD Command Shell, Bind TCP Inline (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/unix/bind_netcat_gaping_ipv6</code></td>
<td>normal</td>
<td>Unix Command Shell, Bind TCP (via netcat -e) IPv6</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/unix/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Unix Command Shell, Bind TCP (via perl) IPv6</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/unix/bind_ruby_ipv6</code></td>
<td>normal</td>
<td>Unix Command Shell, Bind TCP (via Ruby) IPv6</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Command Shell, Bind TCP (via perl) IPv6</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Linux Meterpreter, Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Linux Meterpreter, Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Linux Command Shell, Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Linux Command Shell, Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Linux Command Shell, Bind TCP Inline (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>PHP Command Shell, Bind TCP (via perl) IPv6</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>PHP Command Shell, Bind TCP (via php) IPv6</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>PHP Meterpreter, Bind TCP Stager IPv6</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Ruby Command Shell, Bind TCP IPv6</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Reflective DLL Injection, Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Reflective DLL Injection, Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Meterpreter (Reflective Injection), Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Meterpreter (Reflective Injection), Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Inject DLL, Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Inject DLL, Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Meterpreter (skape/jt Injection), Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Meterpreter (skape/jt Injection), Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Command Shell, Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Command Shell, Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Upload/Execute, Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>VNC Server (Reflective Injection), Bind TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>VNC Server (Reflective Injection), Reverse TCP Stager (IPv6)</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Multi Gather Resolve Hosts</td>
<td></td>
</tr>
<tr>
<td><code>payload/cmd/windows/bind_perl_ipv6</code></td>
<td>normal</td>
<td>Windows Manage Set Port Forwarding With PortProxy</td>
<td></td>
</tr>
</tbody>
</table>
15.3 References

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