IPv6 Neighbor Discovery Protocol based OS fingerprinting

Frédéric Beck, Olivier Festor, Isabelle Chrisment

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

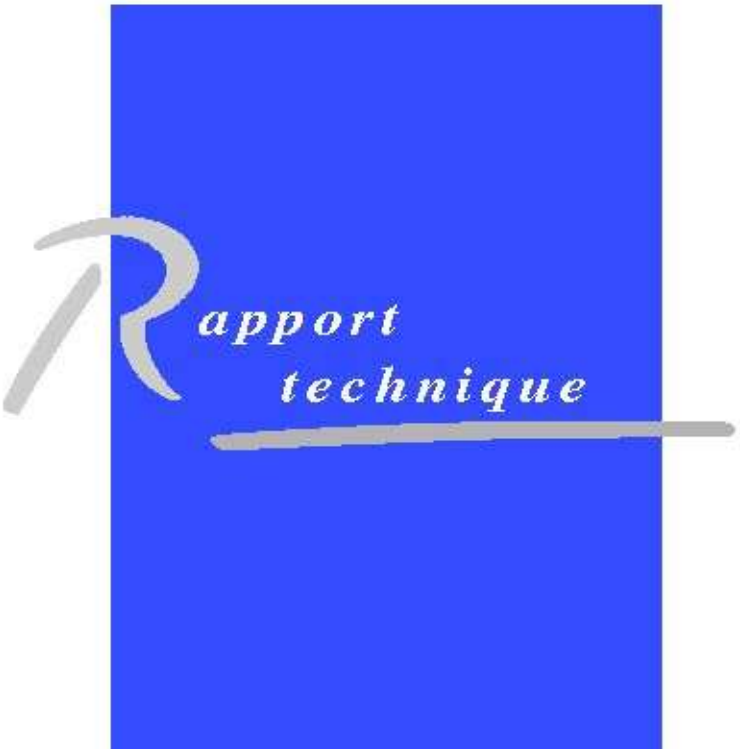
IPv6 Neighbor Discovery Protocol based OS fingerprinting

Frédéric Beck, Olivier Festor and Isabelle Chrisment

N° 0345

March 2007

Thème COM



ISSN 0249-0803

IPv6 Neighbor Discovery Protocol based OS fingerprinting

Frédéric Beck, Olivier Festor and Isabelle Chrisment

Thème COM — Systèmes communicants
Projet MADYNES

Rapport technique n° 0345 — March 2007 — 24 pages

Abstract: Many different parameters impact the management of a device (nature, IP address...), and the Operating System (OS) it runs is one of them. With IPv6 being deployed, many new device, all different, will be able to obtain an IP address and acces the network. Knowing the OS these devices are running is very important in order to manage them efficiently. This is performed with OS Fingerprinting. In this report, we present a study we performed on the subject. We present our methodology and the results observed, leading to the definition of fingerprints for some IPv6 stacks and thus Operating Systems.

Key-words: IPv6, management, monitoring, network

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

OS Fingerprinting basé sur le protocol de découverte de voisins d'IPv6

Résumé : De nombreux paramètres différents impactent le management des systèmes (nature, adresse IP...), et le système d'exploitation (OS) qu'ils utilisent est l'un d'eux. Avec le déploiement d'IPv6 qui a commencé, beaucoup de nouveaux systèmes, tous différents, seront capables d'obtenir une adresse IP et d'accéder au réseau. Connaître l'OS de ces systèmes est donc très important pour pouvoir les gérer efficacement. Cette opération est possible grâce à l'OS Fingerprinting. Dans ce rapport, nous présentons une étude menée sur le sujet. Nous décrivons la méthodolgie utilisée et les résultats collectés, menant à la définition des empreintes de plusieurs piles IPv6 et donc d'OS.

Mots-clés : IPv6, management, supervision, réseau

Contents

List of Figures

1 Introduction

Knowing only the IP address of a host is mandatory but not sufficient to perform all the monitoring and management operations on a system. First of all, knowing the nature of a device is very important, as a printer, a router or a computer are not managed in the same way. Moreover, once the nature of the device itself has been identified, we need more information.

As the nature impacts the way of managing a device, the Operating System (OS) it runs also does. The OS can determine the way of accessing the device or its data, the default services it runs, data ordering... In addition, getting a more precise vision about the version of the OS can also help to detect potential vulnerabilities.

Determining a system's OS is done thanks to a mechanism called OS Fingerprinting. For IPv4, many tools perform this operation, but not all these tools do support IPv6 yet. One of IPv6 building blocks is the Neighbor Discovery Protocol [2], and is used by IPv6 nodes to interact. The idea of this study is to use this protocol to perform OS Fingerprinting on IPv6, and extend a monitoring tool we already developed for this protocol, called NDPMon [1].

In this report, we will firstly present and explain what exactly is OS Fingerprinting. Then, we will introduce our

2 OS Fingerprinting

OS fingerprinting is a process of determining the operating system used by the remote target [1].

2.1 Reasons for OS detection

Some benefits of discovering the underlying OS and device types on a network are obvious, as we already introduced them. This section lists and summarizes some reasons discovering this extra information.

- Determining vulnerability of target hosts: as security holes are usually patched and solved, determining the OS version of a device coupled with a database of security alert can help determining which are the exploits the system is vulnerable to;
- Tailoring exploits: knowing which exploit the target is vulnerable to is not enough, as the same shell codes or instructions are not sent to the same OS (you don't send the same shell code on a Linux than to a Windows box for example);
- Network inventory and support: useful for IT budgeting and ensuring that all company equipment is accounted for, and that you have the required support for all OS on your network;

[1] http://ndmon.sf.net

- Detecting unauthorized and dangerous devices: a regular scanning can help detecting infected or unwanted devices on the network and prevent damages;
- Social engineering.

There are two types of OS Fingerprinting: Active OS fingerprinting and Passive OS fingerprinting

2.1.1 Active OS Fingerprinting

Active fingerprinting is aggressive in nature. An active fingerprinting tool transmits to and receives from the targeted device. It can be located anywhere in the network and with the active fingerprinting method you can learn more information about the target than passive OS fingerprinting. The downside to this method is that the can be identified by an Intrusion Detection System (IDS) on the network.

Active Os Fingerprinting is usually done via TCP Stack Querying, by sending ICMP, TCP or SNMP solicitations to the host, and analyzing the response or their absence.

Another possibility is banner grabbing. It is done by trying to initializing a connection with a remote target, and analyzing the welcome banner. The usual target services are FTP, TELNET and HTTP.

Finally, Port Probing consists in a service scan identifies the services running on a list of open ports. By comparing the result to a knowledge database of standard services/opened ports on OS, we can guess the Os of the targeted host.

2.1.2 Passive OS Fingerprinting

Passive fingerprinting is undetectable by an IDS on the network. A passive fingerprinter (a person or an application) does not send any data across the network because of this nature it is undetectable. The downside to passive fingerprinting is the fact that the fingerprinter must be on the same hub as the other servers and clients in order to capture any packets on the wire. The captured packets or their sequence are then compared to the fingerprints in the internal database, and the Os is determined thanks to a matching algorithm.

2.2 Protect yourself against Fingerprinting

In order to protect against Fingerprinting, the first step is to block all unnecessary outgoing ICMP traffic especially unusual ones like address mask and timestamp also block any ICMP echo replies and watch for excessive TCP SYN packets. Another necessary task, is to set in an appropriate way the welcome banner of usual targeted services

2.3 Fingerprinting tools and IPv6 support

The most well known tool for Active Fingerprinting is Nmap [2]. It is used to evaluate the security of computers, and to discover services or servers on a computer network. Nmap includes host discovery, port scanning, version and OS detection. It may be the most complete tool for fingerprinting, and support IPv6. Unfortunately, the OS detection module does not work yet for IPv6.

p0f [3] is the most well known tool for Passive Fingerprinting. It uses TCP connections it can see to determine the OS of the nodes it can see. It also detects the presence of firewalls, the usage of NAT, the existence of a load balancer setup, the distance to the remote system and its uptime, other guy's network hookup (DSL, OC3, avian carriers) and his ISP. It works well for IPv4 but does not support IPv6 yet.

The best (and only ?) tool performing IPv6 OS Fingerprinting is SinFP [4]. It performs TCP active fingerprinting to determine the remote OS. It works fine, but the fingerprints are not very accurate, and some of the tests could be considered as attacks by IDS.

3 Methodology

In this chapter, we will present more precisely our study, by motivating it, and presenting the experimentations we performed.

3.1 Objective

As IPv6 OS fingerprinting is not widely supported, we wanted to work on the subject and propose such a tool. But as SinFP already performs active IPv6 fingerprinting by using TCP connections, there was no need to do the same operations. We could have extended the fingerprints, or add IPv6 support in Nmap or p0f. As we already made a study on the Neighbor Discovery Protocol (NDP) and developed a tool monitoring this protocol, the idea was to develop a module performing OS fingerprinting based on the NDP. As we wanted the tool to be undetectable and to not disturb the network, we firstly thought about passive fingerprinting, before trying active fingerprinting as shown in the following sections.

We decided to perform our study on several different OS. We chose FreeBSD 6.2, Mac OS X panther and tiger, Windows XP SP2 and Vista RC1, and finally GNU/Linux with various kernel versions. The testbed we used is shown in figure 1.

But this study was not only motivated by the will of fingerprinting OS, we also wanted to take a closer look at IPv6 stack implementations and how these behave.

[2] http://insecure.org/nmap/
[3] http://lcamtuf.coredump.cx/p0f.shtml
[4] http://www.gomor.org/sinfp

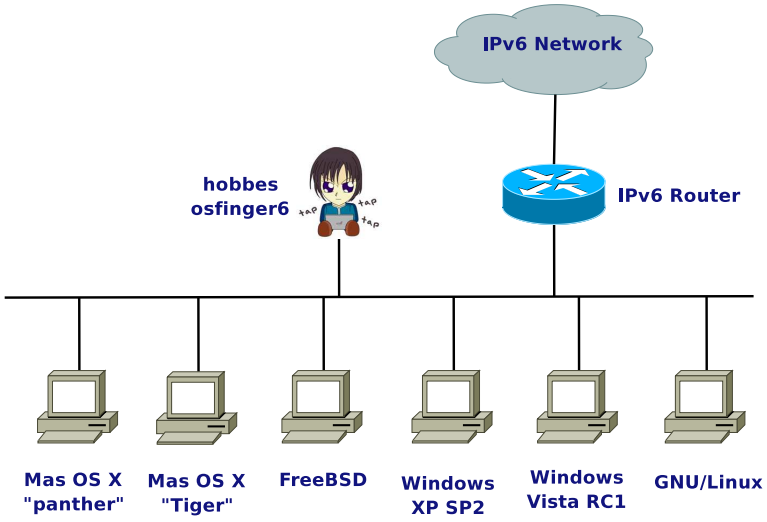


Figure 1: Testbed

3.2 Passive OS Fingerprinting

The first step of our study was to try Passive OS Fingerprinting. To do so, we decided to capture NDP packets sent by the different OS when used normally. We capture then packets sent at boot time, when an interface goes down and up again, and when the station is used normally, for WEB browsing for example.

As the NDP is only composed of 5 ICMPv6 messages (Neighbor Solicitation/Advertisement - NS/NA, Router solicitation/Advertisement - RS/RA and Redirect), and these messages have a simple semantic, we could not extract any strong factor we could use to differentiate the OS.

We decided then, to use longer sequences of packets, including MLD reports sent by the nodes. We confirmed that the only OS supporting MLDv2 natively are Windows Vista and GNU/Linux whereas the other ones use MLDv1. But besides this conclusion, once again, we could not extract any outstanding differences.

We concluded that NDP based Passive IPv6 OS Fingerprinting was not feasible, and decided to go for Active Fingerprinting.

3.3 Active OS Fingerprinting

The first step for this part of the study was focused on end-hosts. The procedure we used was to define a set of NS messages and send them to the targeted host and analyze the responses.

3.3.1 NS messages sent

We decided to target both link local and global addresses on targeted hosts, by using both link local and global addresses as source of the NS. We defined accordingly a set of valid NS messages. In order to get as many data as possible, and analyze more deeply the different stacks, we also defined invalid NS, using reserved addresses (all-nodes...), unawaited options...

Table 1 shows the NS messages we defined and used for this experiment.

NS messages include in their definition 3 other options which can be modified:

- IPv6 Hop Limit: default value is 255 in order to ensure that the packet has not been routed;
- ICMPv6 code: default value is 0;
- ICMPv6 reserved field: by default all bits set to 0, this field should be ignored.

We performed the tests on the targeted nodes by using the default values for all fields, and then by modifying one at the time, and finally all the possible combinations. For Windows XP and Vista who are using *Global Temporary* and *global* IPv6 addresses, the tests are performed on both of them. All the addresses assigned on an interface are thus tested.

Moreover, besides these valid options, we tried also to add in the NS, some other NDP options exist, but are not meant to be used with NS messages. These options are Mobility options, Prefix Informations and MTU, and are supposed to be set in NA messages. We decided to add these options in the NS, and examine the responses or absence of responses to these triggers.

Finally, during the tests, other prefixes appeared on the network, namely a 6to4 prefix (*2002::/16*) and a site local prefix (*fec0::/16*). All the tests previously presented have been performed on all addresses when these prefixes were advertised.

Number	Source Address	Destination Address	Target	Link Layer Option
1	any	all-nodes	all-nodes	No
2	any	all-nodes	all-nodes	Yes
3	any	Sol Multi	LLA	No
4	any	Sol Multi	LLA	Yes
5	any	all-nodes	LLA	No
6	any	all-nodes	LLA	Yes
7	any	all-routers	LLA	No
8	any	all-routers	LLA	Yes
9	any	LLA	LLA	No

continued on next page

continued from previous page

Number	Source Address	Destination Address	Target	Link Layer Option
10	any	LLA	LLA	Yes
11	any	Global	LLA	No
12	any	Global	LLA	Yes
13	any	Sol Multi	Global	No
14	any	Sol Multi	Global	Yes
15	any	all-nodes	Global	No
16	any	all-nodes	Global	Yes
17	any	all-routers	Global	No
18	any	all-routers	Global	Yes
19	any	LLA	Global	No
20	any	LLA	Global	Yes
21	any	Global	Global	No
22	any	Global	Global	Yes
23	any	Sol Multi	Sol Multi	No
24	any	Sol Multi	Sol Multi	Yes
25	any	all-nodes	Sol Multi	No
26	any	all-nodes	Sol Multi	Yes
27	any	all-routers	Sol Multi	No
28	any	all-routers	Sol Multi	Yes
29	any	LLA	Sol Multi	No
30	any	LLA	Sol Multi	Yes
31	any	Global	Sol Multi	No
32	any	Global	Sol Multi	Yes
33	LLA	all-nodes	all-nodes	No
34	LLA	all-nodes	all-nodes	Yes
35	LLA	Sol Multi	LLA	No
36	LLA	Sol Multi	LLA	Yes
37	LLA	all-nodes	LLA	No
38	LLA	all-nodes	LLA	Yes
39	LLA	all-routers	LLA	No
40	LLA	all-routers	LLA	Yes
41	LLA	LLA	LLA	No
42	LLA	LLA	LLA	Yes
43	LLA	Global	LLA	No
44	LLA	Global	LLA	Yes
45	LLA	Sol Multi	Global	No
46	LLA	Sol Multi	Global	Yes

continued on next page

continued from previous page

Number	Source Address	Destination Address	Target	Link Layer Option
47	LLA	all-nodes	Global	No
48	LLA	all-nodes	Global	Yes
49	LLA	all-routers	Global	No
50	LLA	all-routers	Global	Yes
51	LLA	LLA	Global	No
52	LLA	LLA	Global	Yes
53	LLA	Global	Global	No
54	LLA	Global	Global	Yes
55	LLA	Sol Multi	Sol Multi	No
56	LLA	Sol Multi	Sol Multi	Yes
57	LLA	all-nodes	Sol Multi	No
58	LLA	all-nodes	Sol Multi	Yes
59	LLA	all-routers	Sol Multi	No
60	LLA	all-routers	Sol Multi	Yes
61	LLA	LLA	Sol Multi	No
62	LLA	LLA	Sol Multi	Yes
63	LLA	Global	Sol Multi	No
64	LLA	Global	Sol Multi	Yes
65	Global	all-nodes	all-nodes	No
66	Global	all-nodes	all-nodes	Yes
67	Global	Sol Multi	LLA	No
68	Global	Sol Multi	LLA	Yes
69	Global	all-nodes	LLA	No
70	Global	all-nodes	LLA	Yes
71	Global	all-routers	LLA	No
72	Global	all-routers	LLA	Yes
73	Global	LLA	LLA	No
74	Global	LLA	LLA	Yes
75	Global	Global	LLA	No
76	Global	Global	LLA	Yes
77	Global	Sol Multi	Global	No
78	Global	Sol Multi	Global	Yes
79	Global	all-nodes	Global	No
80	Global	all-nodes	Global	Yes
81	Global	all-routers	Global	No
82	Global	all-routers	Global	Yes
83	Global	LLA	Global	No

continued on next page

continued from previous page

Number	Source Address	Destination Address	Target	Link Layer Option
84	Global	LLA	Global	Yes
85	Global	Global	Global	No
86	Global	Global	Global	Yes
87	Global	Sol Multi	Sol Multi	No
88	Global	Sol Multi	Sol Multi	Yes
89	Global	all-nodes	Sol Multi	No
90	Global	all-nodes	Sol Multi	Yes
91	Global	all-routers	Sol Multi	No
92	Global	all-routers	Sol Multi	Yes
93	Global	LLA	Sol Multi	No
94	Global	LLA	Sol Multi	Yes
95	Global	Global	Sol Multi	No
96	Global	Global	Sol Multi	Yes
97	all-nodes	Sol Multi	LLA	No
98	all-nodes	Sol Multi	LLA	Yes
99	all-nodes	all-nodes	LLA	No
100	all-nodes	all-nodes	LLA	Yes
101	all-nodes	all-routers	LLA	No
102	all-nodes	all-routers	LLA	Yes
103	all-nodes	LLA	LLA	No
104	all-nodes	LLA	LLA	Yes
105	all-nodes	Global	LLA	No
106	all-nodes	Global	LLA	Yes
107	all-nodes	Sol Multi	Global	No
108	all-nodes	Sol Multi	Global	Yes
109	all-nodes	all-nodes	Global	No
110	all-nodes	all-nodes	Global	Yes
111	all-nodes	all-routers	Global	No
112	all-nodes	all-routers	Global	Yes
113	all-nodes	LLA	Global	No
114	all-nodes	LLA	Global	Yes
115	all-nodes	Global	Global	No
116	all-nodes	Global	Global	Yes
117	all-nodes	Sol Multi	Sol Multi	No
118	all-nodes	Sol Multi	Sol Multi	Yes
119	all-nodes	all-nodes	Sol Multi	No
120	all-nodes	all-nodes	Sol Multi	Yes

continued on next page

continued from previous page				
Number	Source Address	Destination Address	Target	Link Layer Option
121	all-nodes	all-routers	Sol Multi	No
122	all-nodes	all-routers	Sol Multi	Yes
123	all-nodes	LLA	Sol Multi	No
124	all-nodes	LLA	Sol Multi	Yes
125	all-nodes	Global	Sol Multi	No
126	all-nodes	Global	Sol Multi	Yes
127	Sol Multi	Sol Multi	LLA	No
128	Sol Multi	Sol Multi	LLA	Yes
129	Sol Multi	all-nodes	LLA	No
130	Sol Multi	all-nodes	LLA	Yes
131	Sol Multi	all-routers	LLA	No
132	Sol Multi	all-routers	LLA	Yes
133	Sol Multi	LLA	LLA	No
134	Sol Multi	LLA	LLA	Yes
135	Sol Multi	Global	LLA	No
136	Sol Multi	Global	LLA	Yes
137	Sol Multi	Sol Multi	Global	No
138	Sol Multi	Sol Multi	Global	Yes
139	Sol Multi	all-nodes	Global	No
140	Sol Multi	all-nodes	Global	Yes
141	Sol Multi	all-routers	Global	No
142	Sol Multi	all-routers	Global	Yes
143	Sol Multi	LLA	Global	No
144	Sol Multi	LLA	Global	Yes
145	Sol Multi	Global	Global	No
146	Sol Multi	Global	Global	Yes
147	Sol Multi	Sol Multi	Sol Multi	No
148	Sol Multi	Sol Multi	Sol Multi	Yes
149	Sol Multi	all-nodes	Sol Multi	No
150	Sol Multi	all-nodes	Sol Multi	Yes
151	Sol Multi	all-routers	Sol Multi	No
152	Sol Multi	all-routers	Sol Multi	Yes
153	Sol Multi	LLA	Sol Multi	No
154	Sol Multi	LLA	Sol Multi	Yes
155	Sol Multi	Global	Sol Multi	No
156	Sol Multi	Global	Sol Multi	Yes

Table 1: Forged NS messages

3.3.2 osfinger6

In order to make the tests, we developed a small tool, called osfinger6, by using the Python language. We used a python library, Scapy6 [5], to generate all NDP messages used. The tool architecture is shown in figure 2.

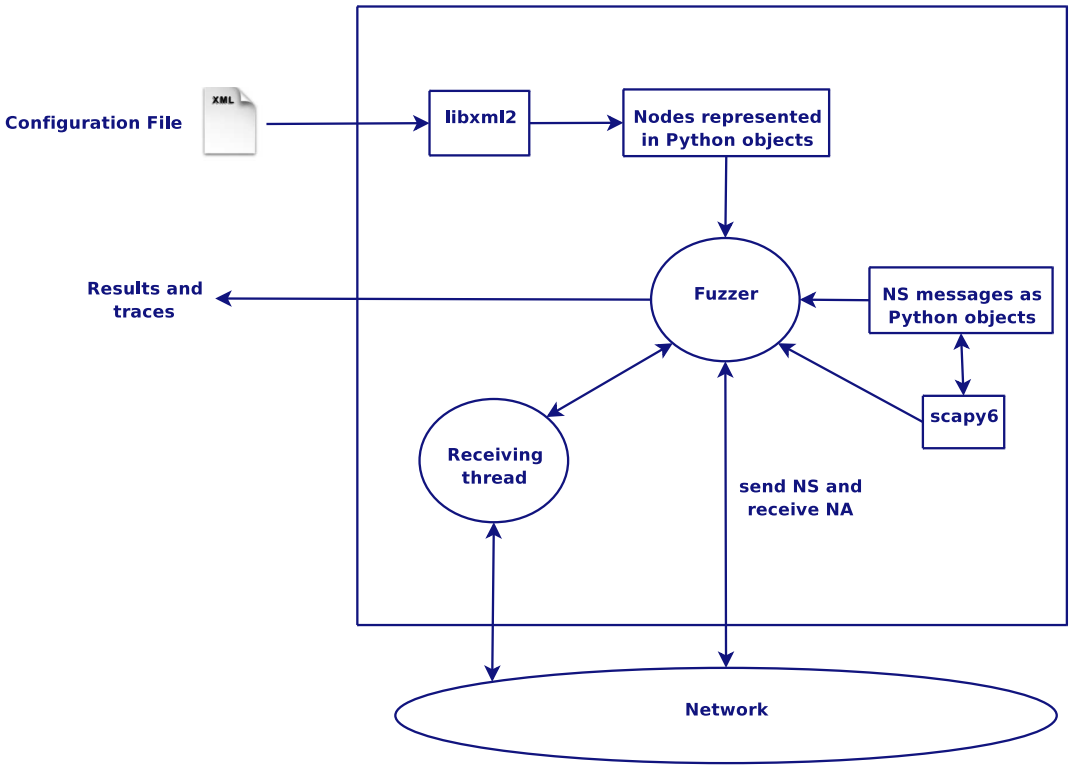


Figure 2: osfinger6 Architecture

In entry, osfinger6 takes an XML file containing a description of the tests to perform. Such a description is composed of the source node of the NS, the targeted node, and the tests to perform (NS, RS...). By parsing this file, Python objects describing the tests are created. Depending on the requested tests, the corresponding messages to send are created thanks to the scapy6 library and stored in python objects.

The fuzzer module is in charge of sending these messages and receiving the responses. As some of these responses are sent to the *all-nodes* IPv6 address, they are not captured by the function used. The tool uses then a second thread, capturing the responses sent to this

[5] http://namabiiru.hongo.wide.ad.jp/scapy6/

address. A timer has been put to ensure the threads synchronization. By default it is set to 5 seconds, which makes the tests quite long. A smaller timer should work as well, as the responses, when we get one, are almost instantaneous.

Finally, when all the tests are done, the results are written in two text files. The first one is a summary of the responses and the key fields in this response, as follows:

```
3;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=0;O=0;res=0;lladdr=1
13;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=0;O=0;res=0;lladdr=1
35;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
36;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
37;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
38;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
[...]
```

The second file contains the source message and the response received, in case we need more informations than the summary:

```
###########################################
For the following request 54:
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= 0
  nh= ICMPv6
  hlim= 255
  src= fe80::213:72ff:fe35:745
  dst= 2001:660:4501:1:213:72ff:fe14:c458
###[ ICMPv6 Neighbor Discovery - Neighbor Solicitation ]###
     type= Neighbor Solicitation
     code= 0
     cksum= 0x0
     R= 0
     S= 0
     O= 0
     res= 0x0
     tgt= 2001:660:4501:1:213:72ff:fe14:c458
###[ ICMPv6 Neighbor Discovery Option - Source Link-Layer Address ]###
        type= 1
        len= 1
        lladdr= 00:13:72:35:07:45


We got the answer:
###[ IPv6 ]###
  version= 6L
```

```
  tc= 0L
  fl= 0L
  plen= 24
  nh= ICMPv6
  hlim= 255
  src= 2001:660:4501:1:213:72ff:fe14:c458
  dst= fe80::213:72ff:fe35:745
###[ ICMPv6 Neighbor Discovery - Neighbor Advertisement ]###
     type= Neighbor Advertisement
     code= 0
     cksum= 0x78d7
     R= 0L
     S= 1L
     O= 0L
     res= 0x0L
     tgt= 2001:660:4501:1:213:72ff:fe14:c458
##############################################

[...]
```

After performing all the test, we could extract the fingerprints and the differences between the OS from these two output files.

4 Fingerprints

In this chapter, we present for each OS the fingerprint we determined, and the specificities of each of them.

For all OS, we got the same result with standard values for the fields *hop limit*, *code* and *reserved* (255;0;0) and when we changed the *reserved* value by putting a random value. These results are jointly presented in the category *standard*.

In the same way, when randomly setting the fields *hop limit* and/or *code*, and combining it with random *reserved* values, the results are identical. We thus categorized them with the label *code*.

4.1 GNU/Linux

Here follows the Linux fingerprint for the standard tests:

```
3;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=0;O=0;res=0;lladdr=1
13;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=0;O=0;res=0;lladdr=1
35;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
36;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
37;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
38;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
41;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
```

```
42;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
43;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
44;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
45;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
46;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
47;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
48;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
51;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
52;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
53;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
54;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
67;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
68;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
69;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
70;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
73;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
74;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
75;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
76;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
77;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
78;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
79;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
80;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
83;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
84;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
85;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
86;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
97;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
98;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
99;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
100;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
103;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
104;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
105;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
106;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
107;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
108;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
109;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
110;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
113;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
114;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
115;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
116;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
127;src=lla;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
128;src=lla;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
133;src=lla;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
```

```
134;src=lla;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
135;src=lla;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
136;src=lla;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
137;src=ip6;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
138;src=ip6;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
143;src=ip6;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
144;src=ip6;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
145;src=ip6;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
146;src=ip6;dst=unknown;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
```

For the code set of tests, we got the fingerprint:

```
41;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
42;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
85;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
86;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
```

These fingerprints are our main references for the comparisons.

4.2 Windows XP SP2 and Vista

For Windows XP SP2 and Vista RC1, we got the same results, whatever the test performed was.

For the standard tests, we got:

```
3;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=0;O=0;res=0;lladdr=1
13;src=ip6;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=0;O=0;res=0;lladdr=1
35;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
36;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
37;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
38;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
41;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
42;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
43;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
44;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
45;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
46;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
47;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
48;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
51;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
52;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
53;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
54;src=ip6;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
67;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
68;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
69;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
```

```
70;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
73;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
74;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
75;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
76;src=lla;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
77;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
78;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
79;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
80;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
83;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
84;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
85;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
86;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
```

For the code tests:

```
41;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
42;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
85;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
86;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
```

4.3 Mac OS X panther and tiger

As it was the case for Windows, we got the same results for both versions of Mac OS X. For the standard tests:

```
3;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=0;O=0;res=0;lladdr=1
13;src=lla;dst=allnodes;hlim=255;tgt=NS;code=0;R=0;S=0;O=0;res=0;lladdr=1
35;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
36;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
37;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
38;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
41;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
42;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
43;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
44;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
45;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
46;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
47;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
48;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
51;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
52;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
53;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
54;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
67;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
68;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
```

```
69;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
70;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
73;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
74;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
75;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
76;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
77;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
78;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
79;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
80;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
83;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
84;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=0
85;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
86;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
```

And for the code tests:

```
41;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
42;src=lla;dst=lla;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
85;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
86;src=ip6;dst=ip6;hlim=255;tgt=NS;code=0;R=0;S=1;O=1;res=0;lladdr=1
```

4.4 FreeBSD

FreeBSD gave us the same results than Mac OS X. It is not quite surprising, as Mac OS X is based on BSD.

4.5 Message NS 44

While doing the tests, we noticed that we had some random responses from the OS on NS 44. We decided thus to test only this message but many times (around 500 times for each target). Here are the results:

GNU/Linux always responses for standard tests, and an average response percentage of 10% for the code tests

Windows XP SP2 We have the same results for the global and the global temporary addresses:

- for the standard tests, the response rate is 100%;
- for the code tests, we have an average response rate from 1 to 4%.

Windows Vista Contrary to XP SP2, this time we have differences between the two global addresses:

- the global address has a response rate of 100% for the standard tests, and an average of 5% for the code ones;
- the global temporary address has a response rate of 100% for the standard tests as well, but 0% for the code tests.

Mas OS X panther and tiger the standard tests give a response rate of 100%, but there are no answers for the code tests.

FreeBSD for the standard tests, the response rate is 100%, whereas it is from 1 to 4% for the code tests.

5 Decision Tree

By basing us on the results obtained and presented in chapter 4, we extracted some specificities for each OS (see section 5.1 and we defined a first version of a decision tree to determine which OS a node is running (section 5.2).

5.1 Specificities

In this section we present the specificities of each OS:

GNU/Linux :

- is the only OS responding for standard NS 97-99, 103-110, 113-116, 127-128, 133-138 and 143-146;
- uses MLDv2 for multicast group reporting.

Windows XP SP2 :

- the Link Layer Address option is set for all NA, especially 43-44, 51-54, 73-76 and 83-84;
- the global temporary address has a response rate of 1 to 4% for NS 44 during the code tests;
- uses MLDv1 for multicast group reporting.

Windows Vista :

- the Link Layer Address option is set for all NA, especially 43-44, 51-54, 73-76 and 83-84;
- the global temporary address has a response rate of 0% for NS 44 during the code tests;

- the interface identifier in the IPv6 Link Local and Global addresses is not the EUI-64;
- the internal firewall supports IPv6 and denies the ping by default;
- uses MLDv2 for multicast group reporting.

Mas OS X :

- uses the Link Local Address as source for responses to NS 13, 45-48 and 51-54;
- uses the Global Address as source for responses to NS 67-70 and 73-76;
- has a response rate of 0% for NS 44 during the code tests;
- uses MLDv1 for multicast group reporting.

FreeBSD :

- uses the Link Local Address as source for responses to NS 13, 45-48 and 51-54;
- uses the Global Address as source for responses to NS 67-70 and 73-76;
- has a response rate from 1 to 4% for the code tests of NS 44;
- uses MLDv1 for multicast group reporting.

5.2 Decision Tree

Thanks to the observation made in section 5.1, we defined the decision tree presented in figure 3.

6 Unusable Remarks

Some of the tests we performed did not give any interesting result, or the nature of the tests and the conditions to fill to execute them were too restrictive. These tests were thus not integrated in the decision tree, and we simply summarize them in this chapter.

6.1 Additionnal NDP Options

We added the NDP MTU option in the NS sent to the targeted hosts, and analyzed the responses we got, by taking as reference the same tests without this option set. It appeared that in all cases, we have the same results, if the MTU option is present or not. To validate the assumption that adding padding to the NS does not change the results, we have to perform the same tests with the other NDP options, and random padding.

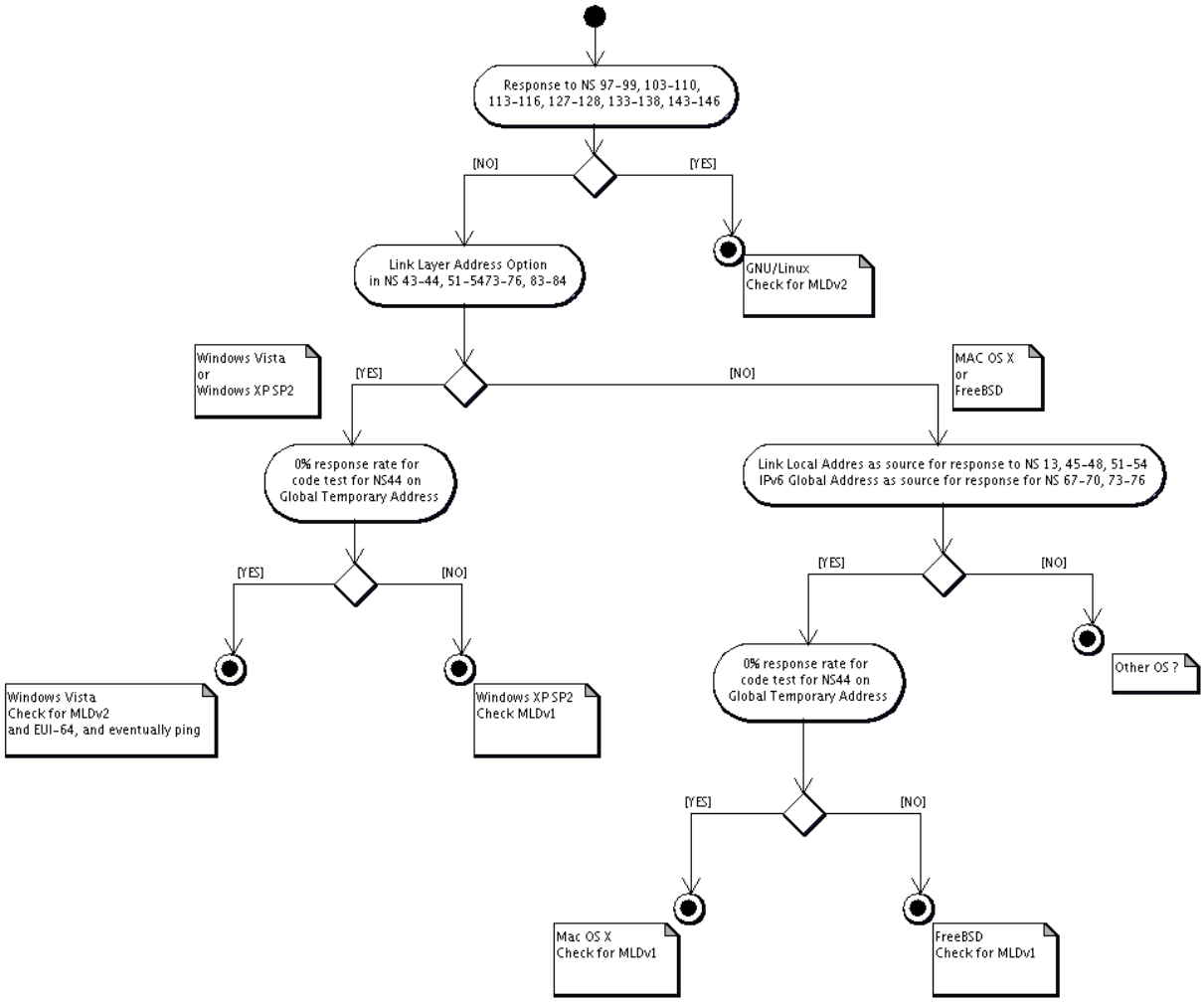


Figure 3: NDP based Active OS Fingerprinting Decision Tree

6.2 Specific Prefixes

When specific prefixes are advertised, namely a 6to4 or site-local prefixes, are advertised on the network, some new differences appear in the OS behavior:

GNU/Linux :

- the Link Layer Address option is not set for NS 85-86 for the standard tests;
- there are no NA in response to NS 67 and 69 for the standard tests;
- for the code tests, only NA in response to NS 41-42 (and not 85-86).

Windows Vista :

- for the code tests, only NA in response to NS 41-42 (and not 85-86).

Mas OS X :

- the Link Layer Address option is not set for NS 85-86 for the standard tests;
- for the code tests, only NA in response to NS 41-42 (and not 85-86).

FreeBSD :

- the Link Layer Address option is not set for NS 85-86 for the standard tests;
- for the code tests, only NA in response to NS 41-42 (and not 85-86).

As the presence of these prefixes on the network was due to tests or misconfiguration from the network, we could not perform the tests on Windows XP SP2 or other versions of the GNU/Linux kernel (the only one tested is the 2.6.18). It would be interesting to set up a testbed with these addresses and perform the same bunch of tests.

The advertising of this kind of prefixes is not very common in real IPv6 networks. Moreover, these prefixes are not something our tool could announce, perform the tests, and then remove. It would have a high impact on the network, which is not the objective of this study. For these reasons, we decided to not take into account these results.

7 Conclusion and Future Work

During this study, we made some very interesting discoveries about the IPv6 stack behaviors of the different OS we tested. Some of these remarks are surprising (Linux responding to NS 97-99, 103-110, 113-116, 127-128, 133-138 and 143-146), others not really (response rate of 0% for NS 44 test in Mac OS X), but all of them show that the stack implementation still have some differences concerning the implementation of the NDP, which is one of the building blocks of IPv6, and a quite simple protocol.

Thanks to these remarks, and by using other known differences, we were able to define a first decision tree. However, at the moment, even if we are able to differentiate the different OS, we can not distinguish precisely the OS versions. It is one point on which we will keep working. To do so, we will need to test also more OS versions (all 2.6 and 2.4 Linux kernels), and more OS (OpenBSD...).

Another point is the tests duration. As we have a timer for the threads synchronization, we lose a lot of time, and the tests are too long to be performed on an operational network. Thus, the tool osfinger6 has to be modified to avoid using a second thread, maybe by testing a newer version of scapy6 or by extending it.

Finally, we will extend the tool to make the same fingerprinting operations on routers, by using both NS and RS messages.

References

[1] Remote OS Detection via TCP/IP Fingerprinting (2nd Generation) - http://insecure.org/nmap/osdetect/.

[2] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461 (Draft Standard), December 1998.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803