

## Kali Linux 渗透测试实战 2.2 操作系统指纹识别

识别目标主机的操作系统，首先，可以帮助我们进一步探测操作系统级别的漏洞从而可以从这一级别进行渗透测试。其次，操作系统和建筑在本系统之上的应用一般 是成套出现的，例如 LAMP 或者 LNMP。操作系统的版本也有助于我们准确定位服务程序或者软件的版本，比如 windows server 2003 搭载的 IIS 为 6.0，windows server 2008 R2 搭载的是 IIS7.5。操作系统指纹识别技术多种多样，这里我简要介绍我所知道的几种常用技术，不会具体深入到细节中，若您感兴趣可自己查阅资料。

### 目录

#### 2.2 操作系统指纹识别

##### 2.2.1 Banner 抓取

##### 2.2.2 TCP 和 ICMP 常规指纹识别技术

##### TCP 数据报格式

##### ICMP 首部格式

##### TTL 与 TCP 窗口大小

##### FIN 探测

##### BOGUS flag 探测

##### TCP ISN 抽样

##### IPID 抽样

##### TCP Timestamp

##### ACK 值

##### ICMP 错误信息

DHCP

2.2.3 数据包重传延时技术

2.2.4 使用 Nmap 进行操作系统探测..

一般性探测

指定网络扫描类型.

设置扫描条件

推测结果.

2.2.5 使用 Xprobe2 进行操作系统探测

2.2.6 使用 p0f 进行操作系统探测

2.2.7 使用 miranda 进行操作系统探测

小结

## 1 . 操作系统指纹识别

识别目标主机的操作系统，首先，可以帮助我们进一步探测操作系统级别的漏洞从而可以从这一级别进行渗透测试。其次，操作系统和建筑在本系统之上的应用一般是成套出现的，例如 LAMP 或者 LNMP。操作系统的版本也有助于我们准确定位服务程序或者软件的版本，比如 windows server 2003 搭载的 IIS 为 6.0，windows server 2008 R2 搭载的是 IIS7.5。

操作系统指纹识别技术多种多样，这里我简要介绍我所知道的几种常用技术，不会具体深入到细节中，若您感兴趣可自己查阅资料。

### 1.1.Banner 抓取

Banner 抓取是最基础、最简单的指纹识别技术，而且在不需要其他专门的工具的情况下就可以做。操作简单，通常获取的信息也相对准确。

严格的讲，banner 抓取是应用程序指纹识别而不是操作系统指纹识别。Banner 信息并不是操作系统本身的行为，是由应用程序自动返回的，比如 apathe、exchange。而且很多时候并不会直接返回操作系统信息，幸运的话，可能会看到服务程序本身的版本信息，并以此进行推断。

凡事皆有利弊，越是简单的方法越容易被防御，这种方法奏效的成功率也越来越低了。

先来看一个直接 Banner 抓取的例子。

```
root@kali-xuanhun: /# telnet 121.101.1.1 80
Trying 121.101.1.1...
Connected to 121.101.1.1.
Escape character is '^]'.
get/ http/1.1
HTTP/1.1 400 Bad Request
Content-Type: text/html; charset=us-ascii
Server: Microsoft-HTTPAPI/2.0
Date: Wed, 25 Dec 2013 10:08:17 GMT
Connection: close
Content-Length: 326
```

在上图中，直接 telnet 80 端口，在返回的服务器 banner 信息中，看到“Server: Microsoft-HTTPAPI/2.0”的字样。

在 IIS 中使用 ISAPI 扩展后，经常会看到这样的 Banner。下表可以帮助我们识别操作系统：

<u>Server Header Value</u>	<u>Windows</u>
	<u>Server Version</u>
Microsoft-HTTPAPI/2.0	Windows 2003 Sp2, Windows 7, Windows 2008, Windows 2008 R2
Microsoft-HTTPAPI/1.0	Windows 2003

如果没有 ISAPI 拦截，我们可能会看到如下图的信息，



Content-Length: 1179

Connection: close

Content-Type: text/html

有经验的管理员都会修改 banner 或者禁止输出 banner 信息，比如下面的测试：



```
root@kali-xuanhun:~# telnet nostromo.joeh.org 80
Trying 86.59.36.167...
Connected to nostromo.joeh.org.
Escape character is '^]'.
get / http/1.0
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

我们可以看到图中目标站点并未输出任何 banner 信息。

除了 web 服务器程序，很多 ftp、smtp 服务也会返回 banner 信息。在 2.3 节《服务程序指纹识别》一节中，还会讲解 Banner 抓取，本节就简要介绍到这里。

## 1.2. TCP 和 ICMP 常规指纹识别技术

正常而言，操作系统对 TCP/IP 的实现，都是严格遵从 RFC 文档的，因为必须遵从相同的协议才能实现网络通信。但是在具体实现上还是有略微的差别，这些差别是在协议规范之内所允许的，大多数操作系统指纹识别工具都是基于这些细小的差别进行探测分析的。

如果您不熟悉 TCP/IP 协议，那么可以查询资料或者跳过这一部分，不影响对工具的使用。

为了节省篇幅，不影响实践学习，我只是简单列出使用的技术，并未深入。

TCP 数据报格式



## tcp 头

中间的标志位 ( flags ) 就是用于协议的一些机制的实现的比特位大家可以看到有 6 比特，它们依次如下：

URG、ACK、PSH、RST、SYN、FIN。

URG 表示紧急指针字段有效；

ACK 置位表示确认号字段有效；

PSH 表示当前报文需要请求推 ( push ) 操作；

RST 置位表示复位 TCP 连接；

SYN 用于建立 TCP 连接时同步序号；

FIN 用于释放 TCP 连接时标识发送方比特流结束。

源端口 ( Sequence Number ) 和目的端口：各为 16 比特，用于表示应用层的连接。源端口表示产生数据包的应用层进程，而目的端口则表示数据包所要到达的目的进程。

序列号：为 32 比特，表示数据流中的字节数。序列号为首字节在整个数据流中的位置。初始序列号随机产生，并在连接建立阶段予以同步。

确认号 表示序号为确认号减去 1 的数据包及其以前的所有数据包已经正确接收，也就是说他相当于下一个准备接收的字节的序号。

头部信息：4 比特，用于指示数据起始位置。由于 TCP 包头中可选项的长度可变，因此整个包头的长度不固定。如果没有附加字段，则 TCP 数据包基本长度为 20 字节。

窗口 :16 位 ,表示源端主机在请求接收端等待确认之前需要接收的字节数。它用于流量控制，窗口大小根据网络拥塞情况和资源可用性进行增减。

校验位：16 位。用于检查 TCP 数据包头和数据的一致性。

紧急指针：16 位。当 URG 码有效时只向紧急数据字节。

可选项：存在时表示 TCP 包头后还有另外的 4 字节数据。TCP 常用的选项为最大数据包（并非整个 TCP 报文）MSS。每一个 TCP 段都包含一个固定的 20 字节的段头。TCP 段头由 20 字节固定头和一些可选项组成。实际数据部分最多可以有 65495 ( 65535 - 20 - 20 = 65495 ) 字节。

#### ICMP 首部格式

4	8	12	16	20	24	28	32
Type		Code		Checksum			
Data							

#### icmp 首部

对于上图中的 Data 部分，不同的 ICMP 类型，会拆分成不同的格式，这里就不一一介绍了。

#### TTL 与 TCP 窗口大小

下表是几个典型的操作系统的 TTL 和 TCP 窗口的大小数值。

Operating System	Time To Live	TCP
------------------	--------------	-----

			Window	Size
Linux (Kernel	2.4 and 2.6)	64	5840	
Google Linux		64	5720	
FreeBSD		64	65535	
Windows XP		128	65535	
Windows Vista	and 7 (Server	128	8192	
2008)				
iOS 12.4	(Cisco Routers)	255	4128	

产生上表中数据差别的主要原因在于 RFC 文档对于 TTL 和滑动窗口大小并没有明确的规定。另外需要注意的是，TTL 即时在同一系统下，也总是变化的，因为路由设备会修改它的值。

基于 TTL 与 TCP 窗口大小的操作系统探测需要监听网络，抓取数据包进行分析，这种方法通常被称之为被动分析。

### FIN 探测

在 RFC793 中规定 FIN 数据包被接收后，主机不发送响应信息。但是很多系统由于之前的固有实现，可能会发送一个 RESET 响应。比如 MS Windows, BSDI, CISCO, HP/UX, MVS, 和 IRIX。

### BOGUS flag 探测

发送一个带有未定义 FLAG 的 TCP SYN 数据包，不同的操作系统会有不同的响应。比如 Linux 2.0.35 之前的系统会在响应包中报告未定义的 FLAG。

### TCP ISN 抽样



TCP 连接的初始序列号 ( ISN ) , 是一个随机值 , 但是不同的操作系统的随机方式不一样 , 还有的操作系统每次的 ISN 都是相同的。针对 ISN 做多次抽样然后比对规律可以识别操作系统类型。

#### IPID 抽样

IP 标识是用来分组数据包分片的标志位 , 和 ISN 一样 , 不同的操作系统初始化和增长该标识值的方式也不一样。

#### TCP Timestamp

有的操作系统不支持该特性 , 有的操作系统以不同的更新频率来更新时间戳 , 还有的操作系统返回 0。

#### ACK 值

在不同场景下 , 不同的请求 , 操作系统对 ACK 的值处理方式也不一样。比如对一个关闭的端口发送数据包 , 有的操作系统 ACK+1 , 有的系统则不变。

#### ICMP 错误信息

ICMP 错误信息是操作系统指纹识别的最重要手段之一 , 因为 ICMP 本身具有多个类型 , 而错误信息又是每个操作系统在小范围内可以自定义的。

#### DHCP

DHCP 本身在 RFC 历史上经历了 1541、2131、2132、4361、4388、4578 多个版本 , 使得应用 DHCP 进行操作系统识别成为可能。

### 1.3 .数据包重传延时技术

之所以把数据包延时重传技术单独拿出来 , 是因为相对于上面说的技术 , 它属于新技术 , 目前大多数系统都没有针对该方法做有效的防御。但是基于该技术的工具也不是很成熟 , 这里希望引起读者的重视或者激发你对该技术的热情。

对于在 2.2.2 节中介绍的技术，很大程度上受到网络环境、防火墙、入侵检测系统的影响。那么数据包重传延时技术能解决这些问题吗？

由于数据包丢失，或者网络阻塞，TCP 数据包重传属于正常情况。为了识别重复的数据包，TCP 协议使用相同的 ISN 和 ACK 来确定接收的数据包。

包重传的延时由重传定时器决定，但是确定一种合适的延时算法比较困难，这是源于以下原因：

确认信号的延迟在实际网络环境中是可变的；

传输的分段或确认信号可能丢失，使得估计往返时间有误。

TCP 采用了自适应的重传算法，以适应互连网络中时延的变化。该算法的基本思想是通过最近的时延变化来不断修正原有的时延样本，RFC 中并没有明确具体如何执行。

由于不同操作系统会选择采用自己的重传延迟算法，这就造成了通过分析各系统重发包的延迟来判断其操作系统类型的可能性，如果各操作系统的重传延迟相互存在各异性，那么就很容易将它们彼此区分开来。

由于此种技术，采用标准的 TCP 数据包，一般情况下可以有效的躲过防火墙和入侵检测系统。但是目前基于此种技术的工具还很少。

#### **1.4. 使用 Nmap 进行操作系统探测**

一般性探测

使用 Nmap 进行操作系统识别最简单的方法为使用 -O 参数，如下是我对内网扫描的几个数据。

```
nmap -O 192.168.1.1/24
```

上面的命令表示对 192.168.1.1 所在网段的 C 类 255 个 ip 进行操作系统版本探测。

对 192.168.1.1 的扫描结果 ( 1.1 是 Tp-link 路由器 ) :

MAC Address: A8:15:4D:85:4A:30 (Tp-link Technologies Co.)

Device type: general purpose

Running: Linux 2.6.X

OS CPE: cpe:/o:linux:linux\_kernel:2.6

OS details: Linux 2.6.23 - 2.6.38

Network Distance: 1 hop

对 192.168.1.101 的扫描结果 ( 实际为 android 系统手机 ) :

MAC Address: 18:DC:56:F0:65:E0 (Yulong Computer

Telecommunication Scientific(shenzhen)Co.)

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

TCP/IP fingerprint:

OS:SCAN(V=6.40%E=4%D=12/27%OT=7800%CT=1%CU=39712%P  
V=Y%DS=1%DC=D%G=Y%M=18DC5

OS:6%TM=52BD035E%P=x86\_64-unknown-linux-gnu)SEQ(SP=100  
%GCD=1%ISR=109%TI=Z%C

OS:I=Z%II=I%TS=7)OPS(O1=M5B4ST11NW6%O2=M5B4ST11NW6  
%O3=M5B4NNT11NW6%O4=M5B4S

OS:T11NW6%O5=M5B4ST11NW6%O6=M5B4ST11)WIN(W1=7120  
%W2=7120%W3=7120%W4=7120%W5

OS:=7120%W6=7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M5B4  
NNSNW6%CC=Y%Q=)T1(R=Y%DF=Y%

OS:T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y  
%DF=Y%T=40%W=0%S=A%A=Z%F=

OS:R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%  
F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T

OS:=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%  
T=40%W=0%S=Z%A=S+%F=AR%O=%RD=

OS:0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=  
G%RIPCK=G%RUCK=G%RUD=G)IE(

OS:R=Y%DFI=N%T=40%CD=S)

从上面的结果可以看出，nmap 对 android 系统识别率不高。

对 192.168.1.102 的结果如下（实际系统为 windows 7 sp1）：

Device type: general purpose

Running: Microsoft Windows 7

OS CPE: cpe:/o:microsoft:windows\_7::-

cpe:/o:microsoft:windows\_7::sp1

OS details: Microsoft Windows 7 SP0 - SP1

对 192.168.1.106 的探测结果如下（实际系统为 ios 5.0）：

MAC Address: CC:78:5F:82:98:68 (Apple)

Device type: media device|phone

Running: Apple iOS 4.X|5.X|6.X

OS CPE: cpe:/o:apple:iphone\_os:4 cpe:/a:apple:apple\_tv:4

cpe:/o:apple:iphone\_os:5 cpe:/o:apple:iphone\_os:6

OS details: Apple Mac OS X 10.8.0 - 10.8.3 (Mountain Lion) or iOS  
4.4.2 - 6.1.3 (Darwin 11.0.0 - 12.3.0)

对 192.168.1.106 的探测结果如下( 实际为苹果一体机、windows7 sp1 ) :

MAC Address: 7C:C3:A1:A7:EF:8E (Apple)

Too many fingerprints match this host to give specific OS details

对 192.168.1.119 的探测结果如下 ( 实际为 windows server 2008

r2,vmware 虚拟机 ) :

MAC Address: 00:0C:29:AA:75:3D (VMware)

Device type: general purpose

Running: Microsoft Windows 7|2008

OS CPE: cpe:/o:microsoft:windows\_7::-

cpe:/o:microsoft:windows\_7::sp1

cpe:/o:microsoft:windows\_server\_2008::sp1 cpe:/o:microsoft:windows\_8

OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008  
SP1, or Windows 8

Network Distance: 1 hop

对 192.168.1.128 探测结果如下 ( centOS 6.4 , VMware 虚拟机 ) :

MAC Address: 00:0C:29:FE:DD:13 (VMware)

Device type: general purpose

Running: Linux 3.X

OS CPE: cpe:/o:linux:linux\_kernel:3

OS details: Linux 3.0 - 3.9

指定网络扫描类型

nmap 支持以下扫描类型：

- sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon

- sN/sF/sX: TCP Null, FIN, and Xmas

- scanflags <flags>: Customize TCP scan flags

- sI <zombie host[:probeport]>: Idlescan

- sO: IP protocol scan

- b <ftp relay host>: FTP bounce scan

比如要使用 TCP SYN 扫描，可以使用如下的命令：

```
nmap -sS -O 192.168.1.1/24
```

设置扫描条件

采用--osscan-limit 这个选项，Nmap 只对满足“具有打开和关闭的端口”条件的主机进行操作系统检测，这样可以节约时间，特别在使用-P0 扫描多个主机时。这个选项仅在使用 -O 或-A 进行操作系统检测时起作用。

如：

```
nmap -sS -O --osscan-limit 192.168.1.119/24
```

推测结果

从上面的扫描示例，我们也能看出，Nmap 默认会对无法精确匹配的结果进行推测的。按官方文档的说法，使用--osscan-guess; --fuzzy 选项，会使推测结果更有效，实际测试没有任何区别。

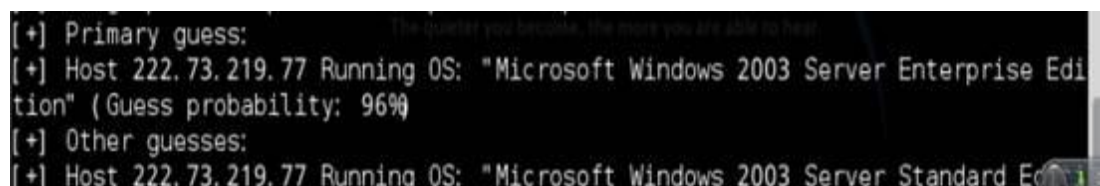
### 1.5. 使用 Xprobe2 进行操作系统探测

Xprobe2 是一款使用 ICMP 消息进行操作系统探测的软件，探测结果可以和 Nmap 互为参照。但是该软件目前公开版本为 2005 年的版本，对老的操作系统探测结果较为准确，新系统则无能为力了。

下面命令为 xprobe2 简单用法：

```
xprobe2 -v www.iprezi.cn
```

结果如下：



```
[+] Primary guess:
[+] Host 222.73.219.77 Running OS: "Microsoft Windows 2003 Server Enterprise Edition" (Guess probability: 96%)
[+] Other guesses:
[+] Host 222.73.219.77 Running OS: "Microsoft Windows 2003 Server Standard Edition"
```

### 1.6.使用 p0f 进行操作系统探测

p0f 是一款被动探测工具，通过分析网络数据包来判断操作系统类型。目前最新版本为 3.06b。同时 p0f 在网络分析方面功能强大，可以用它来分析 NAT、负载均衡、应用代理等。

p0f 的命令参数很简单，基本说明如下：

-f fname 指定指纹数据库 (p0f.fp) 路径，不指定则使用默认数据库。

-i iface 指定监听的网卡。

-L 监听所有可用网络。

-r fname 读取由抓包工具抓到的网络数据包文件。

-o fname 附加之前监听的 log 文件 ,只有同一网卡的 log 文件才可以附加合并到本次监听中来。

-d 以后台进程方式运行 p0f ;

-u user 以指定用户身份运行程序 , 工作目录会切换到当前用户根目录下 ;

-p 设置 -i 参数指定的网卡为混杂模式 ;

-S num 设置 API 并发数 , 默认为 20 , 上限为 100 ;

-m c,h 设置最大网络连接数和同时追踪的主机数 (默认值: c = 1,000, h = 10,000).

-t c,h 设置连接超时时间

下面使用如下命令进行测试 :

p0f -i eth0 -p

上面命令的含义为监听网卡 eth0 , 并开启混杂模式。这样会监听到每一个网络连接 , 部分结果摘录如下 :

```
-[ 192.168.1.108/13860 -> 119.188.46.24/80 (syn+ack) ]-  
server   = 119.188.46.24/80  
os       = Linux 2.6.x  
dist     = 8  
params   = none  
raw_sig  = 4: 56+8: 0: 1440: mss*4, 9: mss, nop, nop, sok, nop, ws; df: 0
```

p0f 监听结果 1

```
-[ 192.168.1.108/13860 -> 119.188.46.24/80 (syn) ]-  
client   = 192.168.1.108/13860  
os       = Windows 7 or 8  
dist     = 0  
params   = fuzzy  
raw_sig  = 4: 64+0: 0: 1260: 8192, 2: mss, nop, ws, nop, nop, sok: df, id+: 0
```

p0f 监听结果 2



在 p0f 监听结果 2 图中，检测的结果我 windows7 或 8，对比下 nmap 的结果为 windows7，实际该机器系统为 windows7 sp1。

```
Device type: general purpose
Running: Microsoft Windows 7
OS CPE: cpe:/o:microsoft:windows_7:- cpe:/o:microsoft:windows_7::sp1
OS details: Microsoft Windows 7 SP0 / SP1
Network Distance: 1 hop
```

nmap 检测结果

```
- [ 192.168.1.108/13858 -> 121.101.223.244/80 (http request) ] -
client    = 192.168.1.108/13858
app       = MSIE 8 or newer
lang      = Chinese
params    = dishonest
raw_sig   = 1: X-Requested-With=[XMLHttpRequest], Accept=[application/json, text/
javascript, */*; q=0.01], ?Referer, Accept-Language=[zh-CN], Accept-Encoding=[gzip,
deflate], User-Agent, Host, DNT=[1], Connection=[Keep-Alive], ?Cookie: Accept-Charset
Keep-Alive: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
```

p0f 监听结果 3

在 p0f 监听结果 3 图中，捕获的数据是浏览器发送的请求数据，我们可以看到浏览器请求信息中“Windows NT 6.1; WOW64; Trident/7.0; rv:11.0”的字样，从这段 UserAgent 中，可以看出发出请求的系统为 windows7 64 位 IE11。

## 1.7. 使用 miranda 进行操作系统探测

miranda 工具是一个通过 UPNP 功能来探测主机信息的工具，并不限于探测操作系统。下面我们通过一个实例，演示如何使用 miranda。

在终端输入如下命令：

```
miranda -v -i eth0
```

上面的命令是指定打开网卡 eth0，返回结果如下：

```
Verbose mode enabled!
Binding to interface eth0 ...
upnp> █
```

miranda 提示输入开启 upnp 的主机，现在我们不知道哪台主机开启了 upnp，输入命令“msearch”，会自动搜索 upnp 主机，

```
Verbose mode enabled!  
Binding to interface eth0 ...  
upnp> msearch
```

接着我们会看到扫描到的 upnp 主机：

```
upnp> msearch  
Entering discovery mode for 'upnp: rootdevice', Ctl+C to stop...  
*****  
SSDP notification message from 192.168.1.1:1900  
XML file is located at http://192.168.1.1:1900/igd.xml  
Device is running ipos/7.0 UPnP/1.0 TL-WR2041N/1.0  
*****  
*****  
SSDP notification message from 192.168.1.1:49152  
XML file is located at http://192.168.1.1:49152/wps_device.xml  
Device is running Unspecified, UPnP/1.0, Unspecified  
*****  
*****  
SSDP notification message from 192.168.1.108:2869  
XML file is located at http://192.168.1.108:2869/upnpghost/udhisapi.dll?  
uid:49dd5a5d-69c0-4a6b-8de7-2755ed48bb6e  
Device is running Microsoft-Windows-NT/5.1 UPnP/1.0 UPnP-Device-Host/1.  
*****
```

按 CTRL +C 终止扫描，输入 host list。

```
upnp> host get 0  
Requesting device and service info for 192.168.1.1:1900 (this could take a few seconds)...  
Device urn:schemas-upnp-org:device:WANDevice:1 does not have a presentationURL  
Device urn:schemas-upnp-org:device:WANConnectionDevice:1 does not have a presentationURL  
Host data enumeration complete!
```

可以看到搜集的主机列表，然后使用 host get [index]命令可以查看该主机的 upnp 设备列表。

```
upnp> host get 0  
Requesting device and service info for 192.168.1.1:1900 (this could take a few seconds)...  
Device urn:schemas-upnp-org:device:WANDevice:1 does not have a presentationURL  
Device urn:schemas-upnp-org:device:WANConnectionDevice:1 does not have a presentationURL  
Host data enumeration complete!
```

使用 host info [index]查看主机详细信息。

```
upnp> host info 0

xmlFile : http://192.168.1.1:1900/igd.xml
name : 192.168.1.1:1900
proto : http://
serverType : ipos/7.0 UPnP/1.0 TL-WR2041N/1.0
upnpServer : ipos/7.0 UPnP/1.0 TL-WR2041N/1.0
dataComplete : True
deviceList : {}
```

从上图信息可以看到，这是一台 TP-Link 路由器。同样的方法，查看一台 windows 7 主机。

```
upnp> host info 2

xmlFile : http://192.168.1.108:2869/upnpHost/udhisapi.dll?content=uuid:49dd5a5d-69c0-4a6b-8de7-2755ed48bb6e
name : 192.168.1.108:2869
proto : http://
serverType : None
upnpServer : Microsoft-Windows-NT/5.1 UPnP/1.0 UPnP-Device-Host/1.0
dataComplete : False
deviceList : {}
```

## 1.9.小结

本节大致罗列了操作系统识别的常用技术和典型工具。因为本书是实践性质的，所以没有对指纹识别技术做深入的讲解。

基于数据包延时重传技术的工具，笔者只知道 RING 和 Cron-OS，但是这两款工具没有集成到 Kali Linux 中，同时也很久没有更新，故没有做介绍。