

WEEK 8 workshop Exercises

OS Fingerprinting the Operating System

Operating system fingerprinting is the method of learning what operating system is running on a host. By analyzing certain protocol flags, options, and data in the packets a device sends onto the network, we can make relatively accurate guesses about the OS that sent those packets. By pinpointing the exact OS of a host, an attacker or penetration tester can launch a specific attack against a target machine. In a world of buffer overflows, knowing the exact version and architecture of an OS could be all the opportunity an attacker needs.

Defining the operating system of a remote system is conducted using two types of scans:

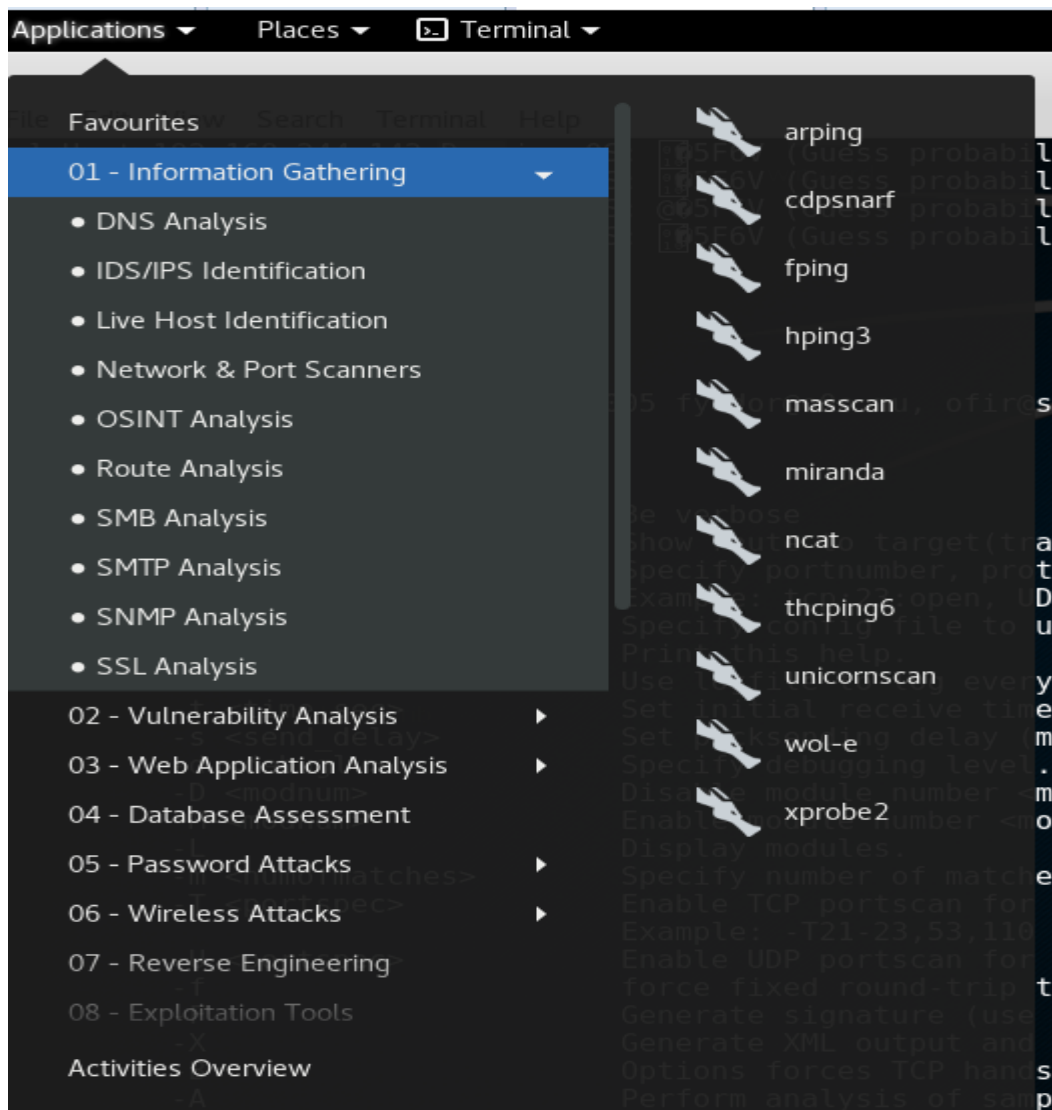
1. Active fingerprinting: The attacker sends normal and malformed packets to the target and records its response pattern, referred to as the fingerprint. By comparing the fingerprint to a local database, the operating system can be determined.
2. Passive fingerprinting: The attacker sniffs, or records and analyses the packet stream to determine the characteristics of the packets. Active fingerprinting is faster and more accurate than passive fingerprinting.

Activity1- OS Active Fingerprinting With Xprobe2 Tool

Xprobe2 is an active operating system fingerprinting tool with a diverse tactics to operating system fingerprinting. *xprobe2* depend on fuzzy signature matching, probabilistic guesses, multiple matches at the same time, and a signature database.

Xprobe2 fingerprints remote operating system by analyzing the replies from the target, so to get the most out of *xprobe2* you need to supply *xprobe2* with as much information as possible, it is important to supply at least one open TCP port and one closed UDP port. Open TCP port can either be provided in command line (-p), found through built-in port scanner (-T) or -B option can be used to cause *xprobe2* to try to blindly guess open TCP port. UDP port can be supplied via command line (-p) or through built-in port scanner (-U).

1. To open Xprobe2 Open Menu **Application** → **Kali Linux** → **Information gathering** → **Live Host Identification** → **xprobe2**



2. Open Terminal type xprobe2

Xprobe2 options

usage: xprobe2 [options] target

Options:

- v Be verbose
- r Show route to target(traceroute)
- p <proto:portnum:state> Specify portnumber, protocol and state.
Example: tcp:23:open, UDP:53:CLOSED
- c <configfile> Specify config file to use.
- h Print this help.
- o <fname> Use logfile to log everything.
- t <time_sec> Set initial receive timeout or roundtrip time.
- s <send_delay> Set pack sending delay (miliseconds).
- d <debuglv> Specify debugging level.
- D <modnum> Disable module number <modnum>.
- M <modnum> Enable module number <modnum>.
- L Display modules.
- m <numofmatches> Specify number of matches to print.
- T <portspec> Enable TCP portscan for specified port(s).
: -T21-23,53,110
- U <portspec> Enable UDP portscan for specified port(s).
- f force fixed round-trip time (-t opt).
- F Generate signature (use -o to save to a file).
- X Generate XML output and save it to logfile specified with -o.
- B Options forces TCP handshake module to try to guess open TCP port
- A Perform analysis of sample packets gathered during portscan in order to detect suspicious traffic (i.e. transparent proxies, firewalls/NIDSs resetting connections). Use with -T.

3. To enable port scanning module Xprobe2, which will scan TCP ports starting from 1 to 1024 on computer with IP address 192.168.X.X type command **xprobe2 -T 1-1024 192.168.244.142**. (note the IP address of the lab is different than the example IP address. When the student is exercising on the lab computer he/she must find the target IP address by using windows command **ipconfig**).
4. To see if the target computer is alive open Terminal and type **xprobe2 192.168. X.X**. (Replace X.X. the target IP address that you want scan).
5. To scan UDP port 53 open terminal and type **xprobe2 -U 53 192.168.X.X**. (During DNS resolution, DNS messages are sent from DNS clients to DNS servers or between DNS servers. Messages are sent over UDP and DNS servers bind to UDP port 53).

More information how to use Xprobe tool explore the link below

[https://man.cx/xprobe2\(1\)](https://man.cx/xprobe2(1))

Activity2 Passive Fingerprinting With P0F Tool

P0f is a tool that advances an array of sophisticated, purely passive traffic fingerprinting mechanisms to recognize the players behind any incidental TCP/IP communications (often as little as a single normal SYN) without interfering in any way. Version 3 is a complete rewrite of the original codebase, incorporating a considerable number of improvements to network-level fingerprinting, and suggesting the ability to reason about application-level payloads (Zalewski, n.d.).

1. Open P0F Menu **Application | Information gathering | P0F** or open Kali Linux Terminal and Then type **P0F**.

2. **P0f** option type kali terminal **P0F -h** command to display the options.

```
root@kali:~# p0f -h
```

```
--- p0f 3.07b by Michal Zalewski <lcamtuf@coredump.cx> ---
```

```
./p0f: invalid option -- 'h'
```

```
Usage: p0f [ ...options... ] [ 'filter rule' ]
```

Network interface options:

- i iface - listen on the specified network interface
- r file - read offline pcap data from a given file
- p - put the listening interface in promiscuous mode
- L - list all available interfaces

Operating mode and output settings:

- f file - read fingerprint database from 'file' (p0f.fp)
- o file - write information to the specified log file
- s name - answer to API queries at a named unix socket
- u user - switch to the specified unprivileged account and chroot
- d - fork into background (requires -o or -s)

Performance-related options:

- S limit - limit number of parallel API connections (20)
- t c,h - set connection / host cache age limits (30s,120m)
- m c,h - cap the number of active connections / hosts (1000,10000)

Optional filter expressions (man tcpdump) can be specified in the command line to prevent p0f from looking at incidental network traffic.

Problems? You can reach the author at <lcamtuf@coredump.cx>.

3. To find out available interfaces to listen the packets open new Kali Linux terminal and then type command **p0f -L** click enter. This command will list all the available network interface on the Kali Linux computer.

```
root@kali:~# p0f -L
--- p0f 3.07b by Michal Zalewski <lcamtuf@coredump.cx> ---

-- Available interfaces --

0: Name      : eth0
   Description: VMware Tools
   IP address : 192.168.244.131

1: Name      : any
   Description: Pseudo-device that captures on all interfaces
   IP address : (none)

2: Name      : lo
   Description: 
   IP address : 127.0.0.1

3: Name      : nflog
   Description: Linux netfilter log (NFLOG) interface
   IP address : (none)

4: Name      : nfqueue
   Description: Linux netfilter queue (NFQUEUE) interface
   IP address : (none)

5: Name      : usbmon1
   Description: USB bus number 1
   IP address : (none)

6: Name      : usbmon2
   Description: USB bus number 2
   IP address : (none)
```

4. Instead of starting fingerprinting process you should execute the following command to start the tool on selected interface with promiscuous mode (listening mode). **P0F -i eth0** < listen on the specified network interface > **-p** < put the listening interface in promiscuous mode> **p0f -i eth0 -p**. (to process this command you need to open the any internet explorer on the Kali Linux and create internet traffic from Kali Linux into target network).

```
root@kali:~# p0f -i eth0 -p
--- p0f 3.07b by Michal Zalewski <lcamtuf@coredump.cx> ---

[+] Closed 1 file descriptor.
[+] Loaded 320 signatures from 'p0f.fp'.
[+] Intercepting traffic on interface 'eth0'.
[+] Default packet filtering configured [+VLAN].
[+] Entered main event loop.
```

Executed tool is expecting to capture TCP/IP packets. When's the traffic is created into target computer the tool will display the captured data. Example: Simple https request has been created from Kali Linux host into home wireless router management console. The TCP/IP connection has been established by using three-way TCP handshake process. The result below indicates that captured is originated Linux **3.11 and newer** and remote host **Ethernet or modem**, IP address of the Remote host **54.70.252.70/443** and communication port 443.

p0f -i eth0 -p

```
--- p0f 3.07b by Michal Zalewski <lcamtuf@coredump.cx> ---
```

```
[+] Closed 1 file descriptor.  
[+] Loaded 320 signatures from 'p0f.fp'.  
[+] Intercepting traffic on interface 'eth0'.  
[+] Default packet filtering configured [+VLAN].  
[+] Entered main event loop.
```

```
.-[ 192.168.229.128/38498 -> 54.70.252.70/443 (syn) ]-  
|  
| client  = 192.168.229.128/38498  
| os      = Linux 3.11 and newer  
| dist    = 0  
| params  = none  
| raw_sig = 4:64+0:0:1460:mss*20,7:mss,sok,ts,nop,ws:df,id+:0  
|  
|-----
```

```
.-[ 192.168.229.128/38498 -> 54.70.252.70/443 (mtu) ]-  
|  
| client  = 192.168.229.128/38498  
| link    = Ethernet or modem  
| raw_mtu = 1500  
|  
|-----
```

```
.-[ 192.168.229.128/38498 -> 54.70.252.70/443 (syn+ack) ]-  
|  
| server  = 54.70.252.70/443  
| os      = ???  
| dist    = 0  
| params  = none  
| raw_sig = 4:128+0:0:1460:mss*44,0:mss::0  
|  
|-----
```

Any traffic created between the two computers will be captured and analyzed during process running.