Frustrating OS Fingerprinting with Morph

Kathy Wang
Syn Ack Labs
knwang@synacklabs.net
Areas Covered in Talk

- OS Fingerprinting History
- What is Morph?
- Morph dependencies
- Morph architecture
- Implementation considerations
- Future directions
- Acknowledgments

www.synacklabs.net
What is OS Fingerprinting?

- Banner information
- Manual reconnaissance
- Active fingerprinting
- Passive fingerprinting
- Timing analysis fingerprinting
OS Fingerprinting History

- QueSO by Apostels
- Nmap by Fyodor
- p0f by Michael Zalewski
- Xprobe/Xprobe2 by Ofir Arkin and Fyodor Yarochkin
- RING by Franck Veysset, et al
Why Defeat OS Fingerprinting?

• Most attacks begin with some form of reconnaissance

• Target host OS information is very important

• OS scanners are designed to exploit expected OS behavior

• OS “honesty” leads to its own demise

• Not entirely vendors’ faults
What is Morph?

- Morph is a process that allows user to select desired OS to emulate
  - Goal: Windows 2000 SP4, Linux 2.4.x.x, OpenBSD 3.3
- Will handle inbound and outbound packets and change TCP, UDP, ICMP and IP headers to reflect selected OS
- Morph is a tool that will currently compile on Linux, and is in development for OpenBSD, FreeBSD, NetBSD
- Not production quality yet
- BSD licensed
- Download at http://www.synacklabs.net/projects/morph

www.synacklabs.net
Morph Dependencies

- Morph is built on Packet Purgatory library
- Wedge between OS kernel and network interface running in userland
- Packet Purgatory is built on libpcap and libdnet libraries
  - libpcap and libdnet provides interfaces to the kernel
High-Level Morph Architecture

- Remote Host
- Packet Purgatory
- Host OS Kernel
- Morph
Morph Internal Architecture

Remote Host

Inbound Handler

State Table

Outbound Handler

Host OS Kernel
More About Packet Purgatory

- Route table maintains IP address to intercept messages to/from
- OS firewall prevents kernel from knowing about packets until done with tampering
- Not a kernel module
- BSD licensed
- http://www.synacklabs.net/projects/packetp
How Does Packet Purgatory Utilize libpcap and libdnet?

• Packet Purgatory has two modes
  • Proxy mode
  • Loopback-firewall mode
Proxy Mode

Remote Host

Inbound

libdnet raw IP write

libpcap sniffs

Proxy IP

Host OS

Kernel

Outbound

libdnet

Raw IP Write

libpcap sniffs

packet sent

libdnet raw IP write
OS scanners that Morph will fool

- QueSO
- Nmap
- Xprobe/Xprobe2
- p0f (in progress)
- RING/Snacktime (in progress)
Other Tools that Defeat OS Fingerprinting

• FPF
• LKM for Linux
• IP Personality
• Patch for Linux 2.4 kernel

• There are a couple of other tools
• None are highly portable
• Most will not emulate another OS
Current OS Fingerprinting Techniques

- Active fingerprinting
- Passive fingerprinting
- Timing analysis fingerprinting
- All of the above can be defeated with Morph
How does QueSO work?

- Utilizes active fingerprinting techniques
- Sends 7 different types of packets to open ports on target host
- All 7 packets sent modify TCP headers (e.g., different flags are set)
- OS fingerprint signatures are somewhat outdated (e.g., no Linux fingerprint beyond 2.1 kernel)
## Morph Response to QueSO

<table>
<thead>
<tr>
<th>QueSO Packet Types</th>
<th>Inbound</th>
<th>State Table</th>
<th>Outbound</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN</td>
<td>If port is open pass packet to OS, else write RST as a response</td>
<td>Add SYN connection</td>
<td>Rewrite packet to reflect emulated OS</td>
</tr>
<tr>
<td>SYN+ACK</td>
<td>Check state table to see if connection is a response</td>
<td>Will update table if packet is solicited</td>
<td>If packet is solicited, then write appropriate ACK reply</td>
</tr>
<tr>
<td>FIN</td>
<td>Pass packet to OS, or in cases of Windows-like behavior, reply</td>
<td>Don’t care</td>
<td>Rewrite packet to reflect desired OS</td>
</tr>
<tr>
<td>FIN+ACK</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>SYN+FIN</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>PSH</td>
<td>Pass packet to OS</td>
<td>Don’t care</td>
<td>Rewrite packet to reflect desired OS</td>
</tr>
<tr>
<td>SYN+XXX+YYY</td>
<td>Depending on emulated OS, respond on behalf of emulated OS</td>
<td>Possibly add SYN connection</td>
<td>May rewrite packet to reflect emulated OS</td>
</tr>
</tbody>
</table>
How does Xprobes2 work?

- Utilizes active fingerprinting techniques
- Xprobes2 sends 4 different types of ICMP packets to target host
- Information request packet is basically obsolete (W. Richard Stevens, TCP/IP Illustrated, Vol. 1)
- UDP packet is sent for ICMP unreachable
- Final packet is vanilla SYN
<table>
<thead>
<tr>
<th>Xprobe2 Packet Types</th>
<th>Inbound</th>
<th>State Table</th>
<th>Outbound</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP ECHO</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>ICMP Timestamp</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>ICMP Address Mask Request</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>ICMP Information Request</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>UDP -&gt; ICMP Unreachable (Includes UDP Port Unreachable Error Message)</td>
<td>If port probed is open, pass to OS. Otherwise, respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Rewrite appropriate reply according to emulated OS</td>
</tr>
<tr>
<td>TCP SYN (Includes TCP Header Information)</td>
<td>If port is open pass packet to OS, else write RST as a response</td>
<td>Add SYN connection</td>
<td>Rewrite packet to reflect emulated OS</td>
</tr>
</tbody>
</table>
How does Nmap work?

- Nmap sends 9 different types of packets to target host
- Needs both open and closed ports for accuracy
- Nmap is challenging to defeat
- Nmap uses many test cases
- Sends non-standard, non-documented packet types to pinpoint OS of target
## Morph Response to Nmap 3.50

<table>
<thead>
<tr>
<th>Nmap Packet Types</th>
<th>Open Port</th>
<th>Closed Port</th>
<th>Morph Handling Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inbound</td>
<td>State Table</td>
<td>Outbound</td>
</tr>
<tr>
<td>TCP Sequence Test</td>
<td>Pass packet to OS</td>
<td>Add SYN connection</td>
<td>Send response packet to reflect emulated OS</td>
</tr>
<tr>
<td>SYN with Options</td>
<td>Pass packet to OS</td>
<td>Add SYN connection</td>
<td>Send response packet to reflect emulated OS</td>
</tr>
<tr>
<td>NULL with Options</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>SYN-FIN-URG-PSH with Options</td>
<td>If OS accepts it, pass to OS. Otherwise, respond on behalf of emulated OS</td>
<td>Add connection</td>
<td>If applicable, send response to reflect emulated OS</td>
</tr>
<tr>
<td>ACK with Options</td>
<td>If connection exists, pass packet to OS. Otherwise, respond on behalf of emulated OS</td>
<td>If part of existing connection, add ACK connection</td>
<td>Send response packet to reflect emulated OS if part of existing connection</td>
</tr>
<tr>
<td>SYN with Options</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>ACK with Options</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>PSH-FIN-URG with Options</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>UDP Packet</td>
<td>Respond on behalf of emulated OS</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
</tbody>
</table>
Morph State Table

- Remote host sends packet
- Morph generates a “random” sequence number based on emulated OS
- Morph state table maintains session sequence number offset information
- Sequence number gets modified on the way to remote OS
Fooling other OS scanners

- p0f (passive OS fingerprinting)
- RING (packet timing analysis)
- Snacktime (packet timing and passive analysis)
New OS Fingerprinting Techniques

- CanSecWest talk on new OS fingerprinting techniques
- Instead of sending single packet to solicit response, sends multiple packets
  - Uses layer 7 info
  - Expands timing analysis
  - Measures window behavior under congested conditions
How can you avoid being fingerprinted?

- New RFC needed to address currently unspecified behavior
- Place hardened critical servers behind intermediate proxying devices
Challenges to Defeating OS Fingerprinting

- Advertising different window size than what underlying OS support
- Having to maintain state of connections to distinguish between normal vs abnormal connections
- Not necessarily having access to standard implementations, and having to glean information through fingerprints
- Even if responses to OS scanners are accurate, application scanning can reveal true OS (implement PolyMorph)
- Some automated attacks do not care what OS it’s attacking (NIMDA)
Future Directions for Morph

- Support more operating system emulation (Solaris, HP-UX, etc)
- Support Morph installs on more operating systems (Windows 2000/XP)
- Fool other OS scanners (p0f, RING, etc)
- Fool application scanners (PolyMorph)
- Add GUI support for Morph
Acknowledgments

- Todd MacDermid
- Bill Neugent
- Don Bailey
- Dan Aiello
- Dave Wilburn
- Bob Fleck
- Dave Dandar
Questions?