Passive Fingerprinting

by Lance Spitzner

last updated Wednesday, May 3, 2000

Relevant Links

Snort
Martin Roes

Know Your Enemy
Lance Spitzner

One of the challenges of network security is learning about the bad guys. To understand your threats and better protect against them, you have to Know Your Enemy. Passive Fingerprinting is a method to learn more about the enemy, without them knowing it. Specifically, you can determine the operating system and other characteristics of the remote host using nothing more then sniffer traces. Though not 100% accurate, you can get surprisingly good results. Craig Smith has developed a proof of concept tool based on the concepts covered in this paper.

Fingerprinting

Traditionally, Operating System fingerprinting has been done using active tools, such as queso or nmap. These tools operate on the principle that every operating system's IP stack has its own idiosyncrasies. Specifically, each operating system responds differently to a variety of malformed packets. All one has to do is build a database on how different operating systems respond to different packets. Then, to determine the operating system of a remote host, send it a variety of malformed packets, determine how it responds, then compare these responses to a database. Fyodor's nmap is tool of choice when using this methodology. He has also written a detailed paper on this.

Passive fingerprinting follows the same concept, but is implemented differently. Passive fingerprinting is based on sniffer traces from the remote system. Instead of actively querying the remote system, all you need to do is capture packets sent from the remote system. Based on the sniffer traces of these packets, you can determine the operating system of the remote host. Just like in active fingerprinting, passive fingerprinting is based on the principle that every operating system's IP stack has its own idiosyncrasies. By analyzing sniffer traces and identifying these differences, you may be able determine the operating system of the remote host.

The Signatures

There are four areas that we will look at to determine the operating system (however there are other signatures that can be used). These signatures are:

- TTL - What the operating system sets the Time To Live on the outbound packet.
- Window Size - What the operating system sets the Window Size at.
- DF - Does the operating system set the Don't Fragment bit.
- TOS - Does the operating system set the Type of Service, and if so, at what.

By analyzing these factors of a packet, you may be able to determine the remote operating system. This system is not 100% accurate, and works better for some operating systems then others. No single signature can reliably determine the remote operating system. However, by looking at several signatures and combining the information, you increase the accuracy of identifying the remote host. An example would be the easiest way to explain. Below is the sniffer trace of a system sending a packet. This system launched a mountd exploit against me, so I want to learn more about it. I do not want to finger or nmap the box, that could give me away. Rather, I want to study the information passively. This signature was captured using snort, my sniffer of choice.

```
04/20-21:41:48.129662 129.142.224.3:659 -> 172.16.1.107:604
TCP TTL:45 TOS:0x0 ID:56257
***F**A* Seq: 0x9DD90553   Ack: 0xE3C65D7   Win: 0x7D78
```

Based on our 4 criteria, we identify the following:

- TTL: 45
- Window Size: 0x7D78 (or 32120 in decimal)
- DF: The Don't Fragment bit is set
- TOS: 0x0

We then compare this information to a database of signatures. First, we look at the TTL used by the remote host. From our sniffer trace above, you can see the TTL is set at 45. This most likely means it went through 19 hops to get to us, so the original TTL was set at 64. Based on this TTL, it appears this packet was sent from a Linux or FreeBSD box, (however, more system signatures need to be added to the database). This TTL is confirmed by doing a traceroute to the remote host. If you are concerend about the remote host detecting your traceroute, you can set your traceroute time-to-live (default 30 hops), to be one or two hops less then the remote host (-m option). For example, in this case we would do a traceroute to the remote host, but using only 18 hops (traceroute -m 18). This gives you the path information (including their upstream provider) without actually touching the remote host. For more information on TTLs, check out this Research Paper on Default TTL values.

The next step is too compare the Window size. I have found the Window Size to be another effective tool, specifically what Window Size is used and how often the size changes. In the above signature, we see it set at 0x7D78, a default Window Size commonly used by Linux. Also, Linux, FreeBSD, and Solaris tend to maintain the same Window Size throughout a session (as this one did). However, Cisco routers (at least my 2514) and Microsoft Windows/NT Window Sizes are constantly changing. I have found that Window Size is more accurate if measured after the initial three-way handshake (due to TCP slow start). For more information on Window Size, see Stevens, "TCP/IP Illustrated, Volume 1" Chapter 20.

Most systems use the DF bit set, so this is of limited value. However, this does make it easier to identify the few systems that do not use the DF flag (such as SCO or OpenBSD). After further testing, I feel that TOS is also of limited value. This seems to be more session based then operating system. In other words, its not so much the operating system that determines the TOS, but the protocol used. TOS defintely requires some more testing. So, based on the information above, specifcally TTL and Window size, you can compare the results to the database of signatures and with a degree of confidence determine the OS (in our case, Linux kernel 2.2.x).

Keep in mind, just as with Active Fingerprinting, Passive Fingerprinting has some limitations. First, applications that build their own packets (such as nmap, hunt, teardrop, etc) will not use the same signatures as the operating system. Second, it is relatively simple for a remote host to adjust the TTL, Window Size, DF, or TOS setting on packets. For example, to change the default TTL value:

- **Solaris:** ndd -set /dev/ip ip_def_ttl 'number'
- **Linux:** echo 'number' > /proc/sys/net/ipv4/ip_default_ttl
- **NT:** HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters

However, by combining a variety of different packets and signatures, in this case TTL and Window Size, you can reliably approximate the remote system.

Other Signatures and Uses

We are not limited to the four signatures discussed so far. There are other areas that can be tracked, such as initial sequence numbers, IP Identification numbers, and TCP or IP options. For example, Cisco routers tend to start IP Identification numbers at 0, instead of randomly assigning them. Also, ICMP payloads can be used. Max Vision discusses using ICMP payload type or TCP options for remote host identification. For example, Microsoft ICMP REQUEST payloads contain the alphabet, while Solaris or Linux ICMP REQUEST payloads have number and symbols. For TCP Options, the option Selective Acknowledgement SackOK is commonly used by Windows and Linux, but not commonly used by FreeBSD or Solaris. Another source of signatures is packet state, what type of packet is being used. To quote Fyodor "For example, the initial SYN request can be a gold mine (as can the reply to it). RST packets also have some interesting features that can be used for identification." These and other signatures can be combined with the signatures listed above to help identify remote operating systems.

Passive fingerprinting can be used for several other purposes. It can be used by the bad guys as 'stealthy' fingerprinting. For example, to determine the Operating System of a 'potential victim', such as a webserver, one only needs to request a webpage from the server, then analyze the sniffer traces. This bypasses the need for using an active tool that can be detected by various IDS systems. Also, Passive Fingerprinting may be used to identify remote proxy firewalls. Since proxy firewalls rebuild connection for clients, it may be possible to ID the proxy firewalls based on the signatures we have discussed. Organizations can use Passive Fingerprinting to identify 'rogue' systems on their network. These would be systems that are not authorized on the network. For example, a Microsoft or Sun shop can quickly identify 'rogue' Linux or FreeBSD systems that mysteriously appeared on their network. Passive Fingerprinting can be used to quickly inventory an organizations operating systems without touch or affecting any systems or network performance.

Building the Database

The database was built by testing a variety of systems with the Telnet, FTP, HTTP, and SSH protocol. More testing needs to be conducted using various other protocols, sessions, and systems. If you have any signatures to add to the database, please send them to lance@spitzner.net. I am especially interested in TCP or IP options or systems not listed in the database.

Conclusion

Passive fingerprinting gives you the ability to learn about the enemy, without them knowing it. Though no single piece of information can positively identify a operating system, by combining several signatures, you can make an approximation of the remote system.

A big thanks to the following people for their help and ideas:

Fyodor
Max Vision
Marty Roesch
Edward Skoudis
Dragos Ruiu
Craig Smith

Lance Spitzner enjoys learning by blowing up his Unix systems at home. Before this, he was an Officer in the Rapid Deployment Force, where he blew up things of a different nature. You can reach him at lance@spitzner.net.