

# Passive Network Analysis

*Stephen Barish* 2007-09-28

In sports, it's pretty much accepted wisdom that home teams have the advantage; that's why teams with winning records on the road do so well in the playoffs. But for some reason we rarely think about "the home field advantage" when we look at defending our networks. After all, the best practice in architecting a secure network is a layered, defense-in-depth strategy. We use firewalls, DMZs, VPNs, and configure VLANs on our switches to control the flow of traffic into and through the perimeter, and use network and host-based IDS technology as sensors to alert us to intrusions.

These are all excellent security measures – and why they are considered "best practices" in the industry – but they all fall loosely into the same kind of protection that a castle did in the Middle Ages. While they act as barriers to deter and deny access to known, identifiable bad guys, they do very little to protect against unknown threats, or attackers that are already inside the enterprise, and they do little to help us understand our networks so we can better defend them. This is what playing the home field advantage is all about - knowing our networks better than our adversaries possibly can, and turning their techniques against them.

## Paranoid? Or maybe just prudent...

Our objective is to find out as much as possible about our own networks. Ideally we could just stroll down and ask the IT folks for a detailed network topology, an identification of our address ranges and the commonly used ports and protocols on the network. It seems counter-intuitive, but smaller enterprises actually do better about tracing this kind of information than gigantic multinational companies, partially because there is less data to track, and also because security and IT tend to work better together in smaller organizations.

In fact, large companies have a real problem in this area, especially if their business model includes growth by acquisition of other companies. Sometimes the IT staff doesn't even know all the routes to the Internet, making it pretty tough to defend these amalgamated enterprises. This is especially common in organizations that grow through mergers and acquisition.

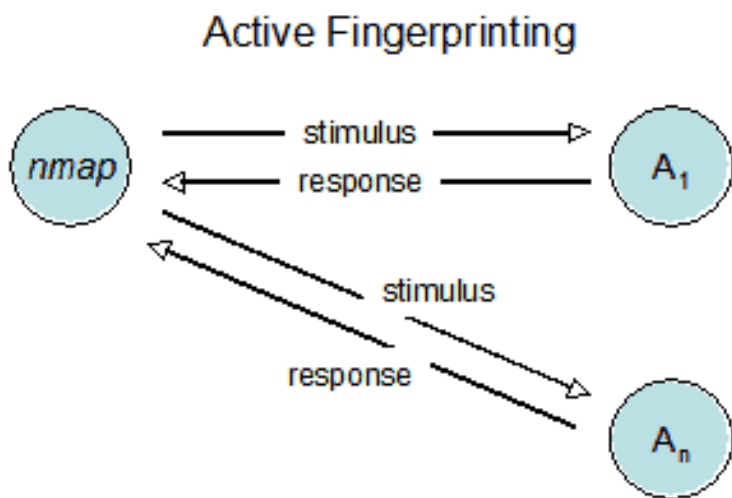
The first, most basic information, we need about our networks in order to defend them well is the network map. Traditionally, attackers and defenders use network mapping technologies such as nmap [1], which use a stimulus-response method to confirm the existence of a host (depending on the options used) to identify its operating system and open ports. This technique relies on non-RFC compliant responses to "odd" packets, and has been around a long time. (Fyodor provides a great paper [2] on the technique, and pretty much pioneered the field of active operating system identification.) Active network mapping is a very powerful technique, but it does have its limitations. It introduces a significant amount of traffic on the network, for one, and some of that traffic can cause problems for network applications. In some cases, nmap can cause operating system instability, although this has become less

common in recent years. They also only provide a snapshot in time of the enterprise topology and composition. Also, active mapping tools generally have difficulties or limitations dealing with firewalls, NAT, and packet-filtering routers. Fortunately there are passive analysis techniques that generate similar results.

## Passive Analysis Theory

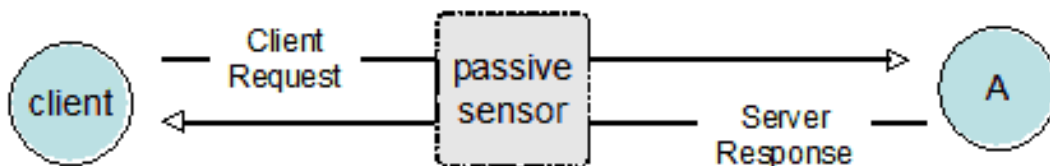
Passive network analysis is much more than intrusion detection, although that is the form of it most commonly used. Passive techniques can map connections, identify ports and services in use in the network, and can even identify operating systems. Lance Spitzner of the HoneyNet project [3] and Michael Zalewski [4] helped pioneer passive fingerprinting techniques that reliably identify operating systems from TCP/IP traces. Zalewski's p0f v 2.0.8 [5] is one of the best passive OS fingerprinting tools available, and is the one used in this article to demonstrate some of the capabilities of the technique.

The key to passive network analysis is understanding that it works almost the same as active mapping and OS fingerprinting. All passive techniques rely on a stimulus-response scenario; they just rely on someone else's stimulus and then collect the response (Figure 1).



Many stimuli from single point-source

## Passive Fingerprinting



Sensor observes network in normal use

## Sensor observes network in normal use

Figure 1 – Active and Passive Network Analysis

In the active scenario, the target (A) responds to stimulus provided by our mapping engine, which is useful, but an artificial observation condition we created just to conduct the mapping exercise. In the passive scenario, the target (A) responds to stimuli resulting from normal use. In both cases we can see the ports and services involved, connection flow, timing information, and can make some educated guesses about our network's operating characteristics from the resulting data. But the passive technique allows something the active one does not: we can see the network from the perspective of the user and application behavior during normal operations.

This has some advantages over active scanning solutions. Passive techniques introduce zero traffic on the monitored network, which can be important in high-availability situations, or where the network is not resilient enough to handle large volumes of scanning traffic. Passive techniques do not generate IDS alerts or log entries in the hosts and servers on the monitored network, reducing the overall analytical burden. In some circumstances, passive techniques can actually identify the presence of firewalls, routers, and switches performing NAT, and potentially characterize the hosts behind them.

In spite of all the advantages associated with the technique, there are limitations as well. Passive analysis always requires the ability to insert a sensor somewhere in the monitored network, either in hardware or software. Sensors also have to be placed in a topological location that allows them to see the traffic of interest, a non-trivial task in modern switched enterprises. Finally, the toolkit for passive analysis is much less mature than traditional active techniques, forcing a higher level of effort on the analyst during sensor deployment, data fusion, and analysis.

To better explain, let's examine a sample capture using Ethereal [6]. You can accomplish the same thing using tcpdump, provided you know how to write good BPF expressions (Berkley Packet Filters). Ethereal provides a lot of tools to save time for the lazy man, including a more rich filtering language and the ability to rapidly sort collected packets on pretty much any data field. Of course, before we start collecting traffic and writing filter code, it's useful to have an idea of what we're looking for – specifically, services running inside our network. Fortunately, there are a couple of rules of thumb that make this a bit easier. First, most services are associated with an assigned port number. These "standard" ports are assigned by the IANA (Internet Assigned Numbers Authority) [7] and are incorporated in the RFCs for standard TCP/IP services and protocols. Less common or malicious source ports can be researched using the Internet Storm Center [8] provided by the SANS Institute [9]. Both also provide excellent reading material on network analysis in general, as well as a snapshot of what other network defenders are seeing on their networks.

Another rule of thumb in analyzing traffic helps us differentiate clients from servers. Generally speaking, servers issue packets on the port number associated with the service. So to find a Web server, you look for TCP Source Port 80 emanating from inside your network. To demonstrate this, I captured some traffic off my home network using Ethereal and sorted it based on TCP Source Port. Sure enough, TCP 80 was there, and when I validated it using simple banner grabbing, the service on 192.168.254.2 was in fact a Web server. Note that

our sensor did not have to initiate traffic in order to make this deduction – the data was sniffed off the wire; I could have accomplished the same thing with nmap, but that would have required the introduction of traffic on the network. One other note of caution – not all TCP protocols obey this rule of thumb, so study your copy of TCP/IP Illustrated!

No.	Time	Source	SPort	Dest	DPort	Protocol	Information
3246	69.141776	192.168.254.2	80	192.168.254.3	61151	TCP	http > 61151 [SYN, ACK] Seq=0 Ack=1 Win=6553
3247	69.143736	192.168.254.2	80	192.168.254.3	61151	TCP	http > 61151 [SYN, ACK] Seq=0 Ack=1 Win=6553
3250	69.147880	192.168.254.2	80	192.168.254.3	61151	TCP	http > 61151 [ACK] Seq=1 Ack=241 Win=65535 L
3251	69.149536	192.168.254.2	80	192.168.254.3	61151	TCP	[TCP Dup ACK 3250#1] http > 61151 [ACK] Seq=
3252	69.224166	192.168.254.2	80	192.168.254.3	61151	TCP	[TCP segment of a reassembled PDU]
3253	69.224219	192.168.254.2	80	192.168.254.3	61151	HTTP	HTTP/1.1 200 OK (text/html)
3254	69.228816	192.168.254.2	80	192.168.254.3	61151	TCP	[TCP Retransmission] [TCP segment of a reas
3255	69.229883	192.168.254.2	80	192.168.254.3	61151	HTTP	[TCP Retransmission] HTTP/1.1 200 OK (text/h
3258	69.367311	192.168.254.2	80	192.168.254.3	61151	TCP	http > 61151 [ACK] Seq=1869 Ack=526 Win=6553
3259	69.367652	192.168.254.2	80	192.168.254.3	61151	TCP	[TCP segment of a reassembled PDU]
3260	69.367669	192.168.254.2	80	192.168.254.3	61151	HTTP	HTTP/1.1 200 OK (GIF89a)
3262	69.375737	192.168.254.2	80	192.168.254.3	61152	TCP	http > 61152 [SYN, ACK] Seq=0 Ack=1 Win=6553
3263	69.376257	192.168.254.2	80	192.168.254.3	61151	TCP	http > 61151 [ACK] Seq=1869 Ack=526 Win=6553
3264	69.378176	192.168.254.2	80	192.168.254.3	61151	TCP	[TCP Retransmission] [TCP segment of a reas
3265	69.380368	192.168.254.2	80	192.168.254.3	61151	HTTP	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a
3266	69.381806	192.168.254.2	80	192.168.254.3	61152	TCP	http > 61152 [SYN, ACK] Seq=0 Ack=1 Win=6553
3270	69.385626	192.168.254.2	80	192.168.254.3	61152	TCP	http > 61152 [ACK] Seq=1 Ack=252 Win=65535 L
3271	69.386082	192.168.254.2	80	192.168.254.3	61152	HTTP	HTTP/1.1 404 Not Found (text/html)
3272	69.388171	192.168.254.2	80	192.168.254.3	61152	TCP	http > 61152 [ACK] Seq=1 Ack=252 Win=65535 L
3273	69.389230	192.168.254.2	80	192.168.254.3	61152	HTTP	[TCP Retransmission] HTTP/1.1 404 Not Found
3529	74.461879	192.168.254.2	80	192.168.254.3	61152	TCP	http > 61152 [ACK] Seq=519 Ack=253 Win=65535
3530	74.461965	192.168.254.2	80	192.168.254.3	61152	TCP	http > 61152 [FIN, ACK] Seq=519 Ack=253 Win=
3532	74.463178	192.168.254.2	80	192.168.254.3	61151	TCP	http > 61151 [ACK] Seq=4481 Ack=527 Win=6553
3533	74.463243	192.168.254.2	80	192.168.254.3	61151	TCP	http > 61151 [FIN, ACK] Seq=4481 Ack=527 Win
3534	74.467071	192.168.254.2	80	192.168.254.3	61152	TCP	[TCP Keep-Alive] http > 61152 [ACK] Seq=519
3535	74.468042	192.168.254.2	80	192.168.254.3	61152	TCP	http > 61152 [FIN, ACK] Seq=519 Ack=253 Win=
3536	74.469486	192.168.254.2	80	192.168.254.3	61151	TCP	[TCP Keep-Alive] http > 61151 [ACK] Seq=4481
3537	74.479453	192.168.254.2	80	192.168.254.3	61151	TCP	http > 61151 [FIN, ACK] Seq=4481 Ack=527 Win

Figure 2 – Capture of Web Sessions Using Ethereal

Thanks to the magic of TCP/IP fingerprinting, which works pretty much the same in passive mode as it does in active mode, we can also make some educated guesses about the operating system of the systems involved in the traffic capture. The technique works because different operating systems implement the TCP/IP stack slightly differently. Spitzner's "Know Your Enemy: Passive Fingerprinting" paper [10] (4 March 2002) discussed four parameters that seemed to vary consistently between operating systems: TTL, Window Size, DF, and TOS. Zalewski's p0f 2.0 expands on these, providing much more granular tests to identify operating systems passively (Figure 3).

Operating System	WSS	Initial TTL	DF	SYN Size	Options <sup>1</sup>	Otions Expanded
Linux Kernel 2.0.3x on Mac	2	64	0	44	M*	MSS
Linux Kernel 2.6	4 x MSS	64	1	60	M*,S,T,N,W5	MSS, Selective ACK OK, Timestamp, NOP, Window Scaling 5
Solaris 9.1	32850	64	1	64	M*,N,W1,N,N,T,N,N,S	MSS, NOP, Window Scaling 1, NOP, NOP, Timestamp, NOP, NOP, Selective ACK OK
Windows XP SP1+	64512	128	1	48	M*,N,N,S	MSS, NOP, NOP, Selective ACK OK
Windows 2003	65535	64	1	48	M1460,N,W2,N,N,S	MSS (1460 bytes), NOP, Window Scaling 2, NOP, NOP, Selective ACK OK
WSS = Window Size		MSS = Maximum Segment Size				
TTL = Initial TTL		DF = Don't Fragment Bit				
1. TCP Options and their order						

Figure 3 – Sample pOf Signatures

Running pOf against the traffic we captured earlier identifies the Web server as a FreeBSD 6.x system, which is consistent with the operating system of the Web server.

```
$ sudo p0f -s capture1.pcap -A
<Fri Aug 24 18:33:17 2007> 192.168.254.2:80 -
FreeBSD 6.x (2) [high throughput] (up: 715 hrs)
```

This example demonstrates the basic principles in passive network analysis. We can use similar tools and techniques to characterize traffic statistics (the percentage of TCP, UDP, ARP, etc.), connection tracking, bandwidth used, the number and size of packets transmitted, etc.

## Potential Uses of Passive Analysis

The power of passive analysis is that you really don't have to do anything to collect data – it is being generated all the time during normal network operations. After all, if users and applications did not need data on servers, the network would not exist in the first place. Our networks are constantly sending data to and from hosts around the enterprise, and every one of those transactions is a potential data source for us to capture and analyze. The challenge for us as analysts is to figure out what information we want from that data:

- **Situational Awareness**

Passive analysis techniques can tell us a lot about our network and how it normally operates. Without a solid understanding of the enterprise, it is very difficult to develop effective security policy and countermeasures. For instance, if you don't know what your address space is, what routes to the Internet are available, or the operating systems that comprise the network, how can you possibly assess whether a particular vulnerability affects your network security posture? Or whether you have deployed firewalls and IDS sensors to the right locations in the network?

- **Policy Enforcement**

Passive analysis can help identify illicit services and other user misbehavior on the network almost instantly. A simple network capture with Ethereal or any other sniffer will identify the presence of streaming media, peer-to-peer file sharing, gaming activity, and other unauthorized use of the network. The easiest way to do this using Ethereal is to filter on packets with a source IP internal to your network, then sort on the TCP or UDP port numbers. In most cases, you will see a common collection of services that are easily identifiable as benign. These tend to be TCP ports used by the operating systems in use on your network and commonly used services (DNS, FTP, HTTP, etc.). The key is to validate the sources you identify as authorized to serve that material, and to use an active measure to validate the results of your passive analysis. After all, there is no reason a Trojan cannot be bound to TCP 80 instead of some arbitrary ephemeral port.

- **Detecting Insider Threats**

Moving beyond policy enforcement, passive analysis has the potential to help identify compromises that were not detected at the perimeter. A good example might be the Wualess back door reported by Symantec [11, Backdoor.Wualess.C]. This threat opens a back door and attempts to contact an IRC server on TCP port 5202 on the domain dnz.3322.org using the channel "#Phantom". There are three discrete criteria we can easily key on in order to detect this particular threat: the presence of TCP 5202, IRC protocol in general, and outbound connections to this unusual domain. If we have good situational awareness (knowing what kinds of traffic are permitted in our enterprise) this would be an easy threat to identify.

- **Incident Response**

Passive analysis is an invaluable tool during incident response operations. Attackers don't compromise systems just to own them; they use them. In most cases automated malicious code follows this same principle. Monitoring the network passively during incident response operations allows real-time visibility into the scope of a compromise, provides clues as to other systems that may be affected, and can provide clues as to where the attack originated.

- **Indications and Warnings**

Many of us maintain "gray lists" of domains that tend to originate attacks. Some gray lists include domains or specific sites whose access violates corporate policy (Internet gaming, pornography, online auctions, etc.). Passively monitoring outbound connections using tools like dsniiff can provide a potential indicator that an attack or misuse of the network has already occurred. Similarly, a pOf log entry that indicated a specific IP address changed operating system would be cause for concern – perhaps a user has dual-booted the host to conceal their activities or to facilitate some kind of attacker behavior. Windows to Linux shifts are particularly concerning for this reason.

These techniques remain cumbersome today, mostly because there are so few integrated tool suites that present the full range of passive analysis capabilities. Nonetheless, they have tremendous potential and are easily implemented in most small to mid-sized networks using

open-source software. By knowing what our networks look like and what they are used for, we can develop that "Home Field Advantage" and steal a march on those attacking our systems.

## About the Author

*Stephen Barish is a Senior Director at MacAulay Brown, Inc., and has been a security researcher and practitioner since 1992. He holds a B.Sc. in Electrical Engineering and is a CISSP.*

[Privacy Statement](#)

Copyright 2006, SecurityFocus