# Global Information Assurance Certification Paper

# Passive Application Mapping

GCIA Gold Assignment

Benjamin Small
Advised By: Jeff Holland



© SANS Institute 2006, Author retains full rights.

# Table of Contents:





© SANS Institute 2006, Author retains full rights.

## Abstract

Tracking and correlating data concerning hosts on a network is an arduous task, but it is immensely beneficial in many aspects. Knowing the version of the software a server is running can tell you if it's vulnerable to exploits. Knowing the types of software a host is offering can help determine what the host is used for. To know this information you have to actively run scans on your network. Although scanning is a solution for testing your applications for security updates, it can have undesirable effects. Nmap is known to cause a system to crash. You also run the risk of utilizing an excessive amount of network bandwidth. Passive Application Mapping (PAM) is a solution for this problem. In this paper I cover the topics that are vital to understanding and utilizing PAM. I also cover the commercial and public efforts that incorporate PAM to better aid in Intrusion Analysis and network maintenance.

## The Concept

You are an intrusion analyst. You receive hundreds of security events an hour and observe seemingly countless firewall drops. You have to decide whether a host attempting port 25 TCP outbound connections is for a legitimate reason or whether it's a mass mailer. You have to determine the impact level of an attack on a web-server. You want to be as accurate as possible so your credibility never comes into question. Knowing if a host is running a mail server or whether a host is running Apache 2.0 or ISS is information that can prevent unnecessary escalation. You are a network admin. You have hundreds of hosts on a network and you need a way to automatically determine which hosts are doing what. Commercial and public tools can utilize PAM to aid in keeping you aware of what is happening on your network. At the very least it can help you determine the severity of an attack or compromise. Passive Application Mapping (PAM) is a tool track vulnerable hosts and to aid in intrusion analysis.

PAM is the ability to identify a service that is being offered on a host by passively analyzing its traffic. Meaning we don't generate any traffic from our utility to determine what is being offering. This gives us the ability to safely map a host where scanning has the potential of causing damage to a server. As traffic on a network is watched, a PAM sensor will detect certain characteristics of what you would expect an application to generate. By doing this, PAM can make a determination as to what is being offered. Knowing what service is being attacked and its version gives us invaluable information for the intrusion analysis process. With this information PAM has great

potential to reduce the likelihood of error when analyzing security events. When an analyst has this information, we are able to make a better decision as to what action is needed. Ultimately, the goal of PAM is to give insight as to what is vulnerable on a network and to aid in security analysis. For example, it is likely that an analyst would determine an inappropriate impact level of an IIS attack against a web-server, because the analyst didn't know that the host that was being attacked was actually running an Apache web-server. Having a database of applications running on a network's server is invaluable in determining if a host vulnerable to an exploit. This information is invaluable in determining what hosts need to be patched as well [8].

PAM works similar to the way intrusion detection systems and passive os fingerprinting (p0f) [4,5] works. Switches and routers can be configured to forward all traffic to a specific physical port. Cisco calls this a Switch Port ANalyzer (SPAN) session. A PAM/IDS sensor is placed on the SPAN port of a switch and is configured to monitor traffic. In it's normal state, a Network Interface Card (NIC) will disregard network packets that are not related to its mac address. To be able to monitor all traffic within its collision domain, the NIC needs to be in promiscuous mode. In promiscuous mode we will be able to see all packets, regardless of the host related to the mac address in the frame header. The PAM sensor is then able to consider the source, destination, protocol, payload, and other characteristics of individual packets that travel to and from hosts on a network. Ideally, software utilizing PAM would first identify the direction of the traffic; which host is offering the service is determined from this information. Once the host offering the service is determined, the protocol and destination port(s) is applied to a set of filters. These filters will try to match software that commonly uses the port(s) and protocol of the

traffic. Going even further in depth, each software offers it's service in a unique way; minute differences between different softwares can be detected to help determine what a server is offering.

## Basic Mapping

To understand fully how sensors utilizing PAM work, we can use the command line utility tcpdump. Using tcpdump, we can put a NIC in promiscuous mode and detect all traffic within the collision domain.

Using *tcpdump -n*, Diagram 1.1:

| |
|---|
| 00:19:57.529954 IP client1.1095 > server1.6667: S 178008520:178008520(0) win 5840 <mss 1460,sackOK,timestamp 22939939 0,nop,wscale 0> |
| 00:19:57.642239 IP server1.6667 > client1.1095: S 15299981:15299981(0) ack 178008521 win 5792 <mss 1452,sackOK,timestamp 111508680 22939939,nop,wscale 2> |
| 00:19:57.642269 IP client1.1095 > server1.6667: . ack 1 win 5840 <nop,nop,timestamp 22939950 111508680> |
| 00:19:57.760082 IP server1.6667 > client1.1095: P 1:47(46) ack 1 win 1448 <nop,nop,timestamp 111508799 22939950> |
| 00:19:57.760126 IP client1.1095 > server1.6667: . ack 47 win 5840 <nop,nop,timestamp 22939962 111508799> |
| 00:19:57.865755 IP server1.6667 > client1.1095: P 47:118(71) ack 1 win 1448 <nop,nop,timestamp 111508902 22939962> |
| 00:19:57.865793 IP client1.1095 > server1.6667: . ack 118 win 5840 <nop,nop,timestamp 22939973 111508902> |
| 00:19:57.890183 IP server1.6667 > client1.1095: P 118:154(36) ack 1 win 1448 <nop,nop,timestamp 111508929 22939962> |
| 00:19:57.890196 IP client1.1095 > server1.6667: . ack 154 win 5840 <nop,nop,timestamp 22939975 111508929> |
| 00:19:58.976063 IP client1.1095 > server1.6667: P 1:7(6) ack 154 win 5840 |

Diagram 1.1 is a packet capture of an IRC connection between

*client1* and *server1*. *client1* has sent a SYN packet to *server1* on port 6667 and *server1* has responded with a SYN/ACK. This tells us that *server1* is offering a service on port 6667. In the first packet we see right away that this connection is on port 6667 TCP. Our first assumption would be that *server1* is offering IRC on port 6667, since this port has been classically used for IRC. This information alone is not enough to determine the actual service that is being hosted; software can be configured or programmed to use a different port. To actually determine the service being offered, we need to test the payload. Typically software will offer a banner of some sort. Grabbing a banner can be extremely effective at determining the service being offered. Often, a version is included as well as the name of the software. The PAM software can grab this information and associate it with the asset we are mapping.

Using *tcpdump -n port 53* Diagram 1.2:

| |
|---|
| 21:23:33.430321 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: UDP (17), length: 58) client1.1025 > 66.153.128.98.53: [udp sum ok] 53503+ A? www.sans.org. (30) |
| 21:23:33.679727 IP (tos 0x0, ttl 50, id 19249, offset 0, flags [none], proto: UDP (17), length: 159) 66.153.128.98.53 > client1.1025: 53503* q: A? www.sans.org. 2/3/0 www.sans.org. A 64.112.229.132, www.sans.org.[\|domain] |

The above packet capture is of a DNS lookup for www.sans.org [10]. When analyzing TCP packets we have had packet flags to help us determine direction and nature of each packet. UDP does not use flags. With out flags, a software much track state on the application layer. When dealing with UDP connections PAM will have to attempt to handle state internally. The order in which packets arrive and whether the ports are ephemeral or not will be the key factors in determining which host is the source and which host is the

destination. The first packet is destined for port 53. Since this isn't ephemeral and our source port, being 1025, it's safe to assume that this packet is initiating the connection. Now that the host offering the service has been determined, we will have to test the payload for typical port 53 UDP traffic. I'll demonstrate this in the next few pages.

## Filtering

Matching common services of ports related to that traffic you see is only an indication of what service is being offered. To accurately map what a host is serving, you have to analyze the data inside of each packet. As traffic is generated, a PAM sensor will watch this traffic and look for pieces of information in the payload. Certain strings will be caught and accounted for. When a sensor sees a connection to a host on port 80 and that the first packet with payload contains a string similar to “GET / HTTP/1.1” it will determine this to be an HTTP connection.

Using *tcpdump -n port 21* Diagram 1.3:

| |
|---|
| 14:41:11.164060 IP client1.45085 > server1.21: S 3798156035:3798156035(0) win 5840 <mss 1460,sackOK,timestamp 6387684 0,nop,wscale 2> |
| 14:41:11.164185 IP server1.21 > client1.45085: S 1497216824:1497216824(0) ack 3798156036 win 5792 <mss 1460,sackOK,timestamp 43283979 6387684,nop,wscale 0> |
| 14:41:11.164206 IP client1.45085 > server1.21: . ack 1 win 1460 <nop,nop,timestamp 6387685 43283979> |
| 14:41:11.225683 IP server1.21 > client1.45085: P 1:266(265) ack 1 win 5792 <nop,nop,timestamp 43283985 6387685> |

Above is a packet capture of a FTP connection between *client1* and *server1*. Sensors utilizing PAM will see that the first packet is a

TCP packet with the SYN flag set. This packet is destined for port 21. The following packet is the SYN/ACK response from *server1*. PAM knows that server1 is offering a service on port 21, since *server1* received a SYN on port 21 and responded with a SYN/ACK. Now that the serving host has been determined, PAM will gather information from the payload of the packets following the last packet of the three-way-handshake(SYN -> SYN/ACK -> ACK).

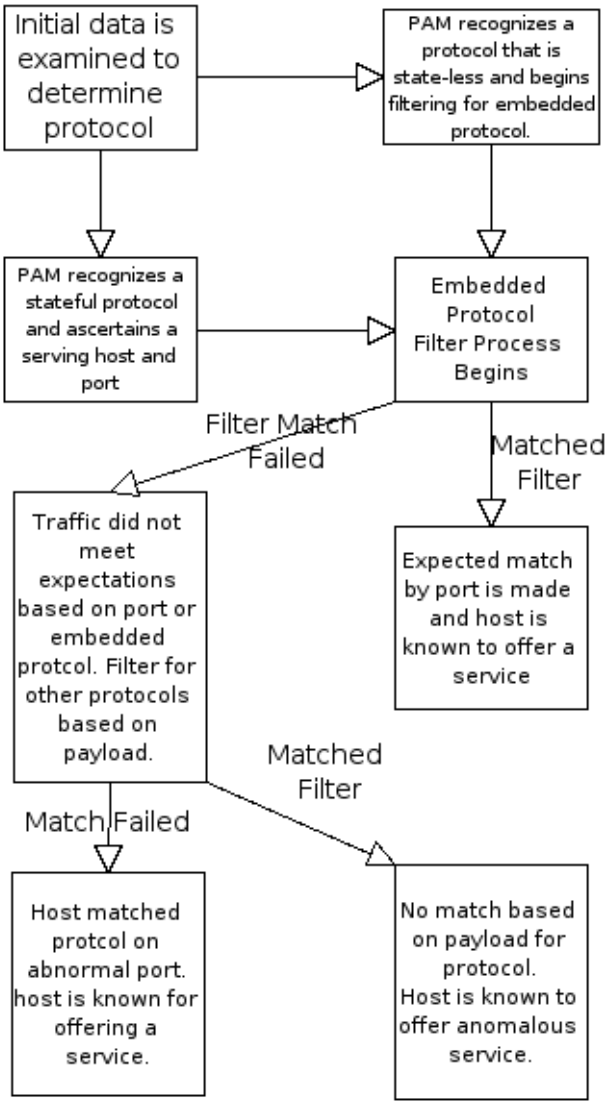
Using *tcpdump -n -X -s 1500 port 21* Diagram 1.4:

```
14:41:11.225683 IP server1.21 > client1.45085: P 1:266(265) ack 1 win 5792 <nop,nop,timestamp 43283985 6387685>
        0x0000:  4510 013d a29b 4000 4006 12f6 c0a8 0164  E..=..@.@......d
        0x0010:  c0a8 0165 0015 b01c 58a3 ccb6 e048 e320  ...e....X....H..
        0x0020:  8018 16a0 f3af 0000 0101 080a 0294 7611  ..............v.
        0x0030:  0061 77E5 3232 302d 2d2d 2d2d 2d2d 2d2d  .aw.220---------
        0x0040:  2d20 5765 6c63 6f6d 6520 746f 2050 7572  -.Welcome.to.Pur
        0x0050:  652d 4654 5064 205b 7072 6976 7365 705d  e-FTPd.[privsep]
        0x0060:  205b 544c 535d 202d 2d2d 2d2d 2d2d 2d2d  .[TLS].---------
        0x0070:  2d0d 0a32 3230 2d59 6f75 2061 7265 2075  -..220-You.are.u
        0x0080:  7365 7220 6e75 6d62 6572 2031 206f 6620  ser.number.1.of.
        0x0090:  3530 2061 6c6c 6f77 6564 2e0d 0a32 3230  50.allowed...220
        0x00a0:  2d4c 6f63 616c 2074 696d 6520 6973 206e  -Local.time.is.n
        0x00b0:  6f77 2031 363a 3037 2e20 5365 7276 6572  ow.16:07..Server
        0x00c0:  2070 6f72 743a 2032 312e 0d0a 3232 302d  .port:.21...220-
        0x00d0:  5468 6973 2069 7320 6120 7072 6976 6174  This.is.a.privat
        0x00e0:  6520 7379 7374 656d 202d 204e 6f20 616e  e.system.-.No.an
        0x00f0:  6f6e 796d 6f75 7320 6c6f 6769 6e0d 0a32  onymous.login..2
        0x0100:  3230 2059 6f75 2077 696c 6c20 6265 2064  20.You.will.be.d
        0x0110:  6973 636f 6e6e 6563 7465 6420 6166 7465  isconnected.afte
        0x0120:  7220 3135 206d 696e 7574 6573 206f 6620  r.15.minutes.of.
        0x0130:  696e 6163 7469 7669 7479 2e0d 0a         inactivity...
```

Diagram 1.4 is a detailed packet capture following the three way handshake. We see the banner that *server1* responded with indicates that it is offering FTP on port 21. We also see from the banner that the application being used to offer this service is Pure-FTPd. This is the default response for this version Pure-FTPd. The filtering system that PAM uses, will successfully match the protocol as being FTP and the application as Pure-FTPd.

This particular filter will search for the string “Welcome to Pure-FTPd”. The sensor will only utilize this on first packet following the three way handshake that is destined for port 21 TCP. The sensor only matches the first packet following the three way handshake with this filter. Since the author of the filter knows that Pure-FTP will respond with this, they have limited the banner detection to the first packet. Filtering can be effective, but has the possibility of being exploited to evade proper detection. With this in mind, a PAM sensor should confirm the determination by matching the protocol that follows. Continuing further and matching protocol prevents incorrect mapping due to evasion tactics and aids in determining other useful information about a host and what it's offering. If the sensor can't determine the version of the software being mapped initially, it may be revealed by the characteristics of the protocol that follows. When solicited with a netbios name lookup, a host responds with a host name that can be cached and for later viewing.

Diagram 1.5:




In review, diagram 1.5 shows that PAM sensors will have to determine the host that is offering a service and what port it is offering this service on. PAM will use packet flags or match payload and consider port numbers, if the traffic is not TCP, to determine the serving host. If an originating host is unable to be determined, PAM can continue on to match the protocol and try to determine direction by the protocol being used. If the serving host and port are determined, a filter system will try to match packet payload to common protocol expected based on the serving port. When PAM successfully maps serving protocol it will record the information it has gathered, preferably in a database and the mapping process has completed. If the a match fails, it will then testing against all filters to determine if the service is being offered on an abnormal port. If the sensor is unable to match the payload with a protocol, the host will be regarded as hosting an anomalous service.

## Benefits to security

### Extra data for analyzing events

In the world of security, data is the key to being accurate. An intrusion analyst will attempt to utilize all resources to determine the impact level of a possible attack. The location of the hosts, the operating systems they may be running, the port they are using for certain services, the software, and the version of the software, that is offering the service on that port, is all invaluable information when analyzing security events. As previously mentioned, if a host is attacked with an IIS exploit, and an analyst knows that the web-server that is being attacked is running Apache; he would know that the host that is being attacked wouldn't have been compromised and would determine a more appropriate impact level for that particular attack. If that web-server were running Apache 2.0, and the web-server was attacked with an exploit that Apache 2.0 is vulnerable to, the analyst would regard the attack with a much higher impact level.

### Reduced rate of analytical false positives

False positives are the bane of every intrusion analysts existence. Unnecessary security events detract from an analyst's ability to spend the needed time researching actual attacks. When a signature on an IDS misfires, it generates a false positive. This event will now be analyzed by a security professional who will take time to

determine that the event has no security value. False positives occur for several reason. Typically, a false positive is generated by a signature that matches a broad range of conditions. For example, a snort signature detecting /etc/passwd files being transferred across the net typically misfire since the signature only looks for "/etc/passwd" in traffic's payload. When a host is known for offering a particular service, ESM's and PAM sensors can be tuned to detect erroneous event creation. For example,  a web-server is hosting HTTP on port 8080 TCP. PAM has not detected any indication of a change in that service offered. The server occasionally triggers signatures that detects file sharing and IRC traffic. These signatures triggered because the server was offering logs of an IRC server. The filesharing signature fires because the index has the name of a popular filesharing software's index page. The sensor has already determined that this host is offering HTTP on port 8080 TCP using Apache. With this information the IDS or ESM can automatically determine that these events are false positive and suppress them. At the very least it can give insight to an analyst that isn't familiar with the server and what it's offering.

**Profiling**

With the data aggregated by PAM a sensor or ESM can generate a profile for each individual host on a network. Since PAM is constantly mapping the traffic it watches, it can automatically contribute to or generate these profiles. These profiles can be used in many different ways. A network admin can look up a host to determine if a host needs to be updated. There is potential for an IDS to integrate these profiles and consider the data when generating

events. The initial severity of an event can be indicated by the IDS if it knows that the host being attacked is actually vulnerable to the attack. Seeing an attack that is dated against a server we know is patched can generate a low priority event. Where as an event related to an unpatched exploit against a known vulnerable server would generate a high priority event.

With this information at hand, a sensor utilizing both IDS and PAM could generate a report based on its signatures to inform the customer of whether it's assets are vulnerable to an attack or not. In the situation that a company believes their software is securely patched, and it's not, PAM would give them information that would help them discover hosts that may have been over looked or misconfigured and are not being patched properly.

## Challenges of effective application mapping

### SPAN

PAM can only map applications from network traffic it can see. Hosts that don't pass their traffic through the sensor's collision domain can't be mapped. SPAN sessions from switches provide a method of bringing traffic with in a sensor's collision domain. Sometimes this isn't possible and multiple sensors will have to be used. By placing sensors in each pertinent collision domain on a network, each segment can be mapped. Centralizing data via a logging method, such as syslog, will give the ability to easily correlate this data with other information related to a network.

### Sensor Load

In situations where a sensor is monitoring large amounts of a traffic there will be a loss of accuracy in PAM's ability to map. Restricted by its hardware and software limitations, packets will be dropped if the PAM software becomes overwhelmed. This is typical with IDS and the known solution is to place multiple sensors on smaller network segments. By strategically placing sensors on a network, packet loss and loss of accuracy can be prevented.

### False Positive Mapping

The possibilities of false positive mapping arises when we have a protocol that uses an embedded protocol. File sharing software often uses HTTP in its connections. This gives the possibility of a

filesharing client being mapped as offering HTTP on an abnormal port. Other services transfer logs of traffic on the protocol level. If the protocol being used isn't matched, the embedded protocol will be, causing a host to be falsely recorded as offering the embedded protocol. PAM as a concept doesn't offer a solution to this problem. Sensors utilizing PAM analyze mapping records and determine a confidence level of its results. Each sensor has a unique method of determining this confidence score. Confidence may be based on the ratio of the service it has mapped the host as offering and other services that host has been mapped as offering. When a host has been mapped only a few times offering IRC on port 8080, and has been mapped thousands of times as offering HTTP on 8080, a sensor will have a high confidence score for the host as offering HTTP.

#### Encryption

PAM maps the application layer on the OSI model. It does this by matching expected strings to a list of filters for expected responses. With that said, encryption poses a huge problem. If an application doesn't offer a banner before the traffic becomes encrypted, there is no way to map it. Once traffic becomes encrypted, the underlying protocol appears to be random characters; unless you have a way to decrypt it. Since PAM is Dependant on matching patterns with-in traffic,  mapping encrypted data is impossible.

#### Subterfuge

Like IDS, there are situations where proper detection may be

evaded. If an attacker compromises a system and suspects that a host is being mapped he may evade detection by offering fake banners or embedding malicious activity in a friendly protocol. In the situation an attacker writes a custom trojan or backdoor they can offer an Apache banner through an HTTP connection as his command and control. With this type of evasion a host will appear to offer a benign service when the connection is actually. PAM sensors can inform staff of new services by creating an event or through reports ran on the sensor's database. Staff can then follow up and confirm that the hosts offering new services are known and authorized to be hosting them.

## Real-world Examples

### Passive Asset Detection System(PADS)

PADS is an open source project that utilizes the concepts of PAM to map applications on a network. PADS maps applications with basic string matching with regular expressions. PADS comes with a Perl script, peds-report, to read it's log file, assets.csv, and produce a human readable report via shell output. PADS can be ran as a daemon or on the command line to map a specific host or network.

PADS was developed with IDS in mind. Quoting the “about” section on the project page [11], “*By day I am an IDS analyst for a managed security provider. One of the challenges I face on a daily basis is the inability to obtain customer asset data, particularly with the larger customers. This information is critical not only for analysis but also device placement and tuning.*” [3]. The head developer of the project is Matt Shelton. The project is a bit dated and there hasn't been an update on the website since June 18th, 2005 when version 1.2 was released. There is no news of official abandonment.

PADS uses libpcap and logs to a CSV file. PADS is intentionally designed to be lightweight and portable, as it doesn't use a database for data storage and uses as little external libraries as possible. This makes PADS ideal for someone who needs to utilize PAM on several hosts on a diverse network or on network where the resources are analyzing a high load.

PADS, by default, maps all the traffic that it sees. You can specify the networks that it maps with the -n option and listing the networks separated by commas. PADS also takes Berkley Packet Filters (BPF) as input so that you can specify which hosts, networks, protocols, and ports are mapped. By default, it dumps all mapped hosts to assets.csv in the directory you run pads from. Alternatively, you can specify the file it should write to with the -w option.

Testing version 1.2 has proven to produce reliable protocol information, however the version of the application it maps seems to be less reliable. I attempted connecting using several protocols to internal and external hosts. In the follow examples I am running PADS with syntax "*pads -v"* for verbose output*:*

Making a ssh connection to a box on my local network it was able to determine the software and version:

| |
|---|
| 192.168.1.100,0,0,ARP (Abit Computer Corporation),0:50:8D:51:9D:3E,1137690463 |
| 192.168.1.100,22,6,ssh,OpenSSH 3.8.1p1 (Protocol 1.99),1137690466 |

Making n HTTP connection to www.sans.org [10] it ascertained a software, but not a version:

| |
|---|
| 64.112.229.132,80,6,www,Apache,1137690531 |

In the following detailed packet capture of a HTTP connection to www.sans.org [10], we see that the webserver responded with "Server: Apache". Note there was no version information offered,

thus PADS did not determine the version of the software being hosted.

| |
|---|
| 08:43:16.134863 IP (tos 0x0, ttl 39, id 52878, offset 0, flags [DF], proto: TCP (6), length: 1420) 64.112.229.132.80 > 172.16.1.101.57906: ., cksum 0x9f23 |
| (correct), 1:1369(1368) ack 468 win 8576 <nop,nop,timestamp 201643951 591501929> |
| 0x0000: 4500 058c ce8e 4000 2706 ac73 4070 e584 E.....@.'..s@p.. |
| 0x0010: ac10 0165 0050 e232 661a 97ec 890f 4bc2 ...e.P.2f.....K. |
| 0x0020: 8010 2180 9f23 0000 0101 080a 0c04 d7af ..!..#..... |
| 0x0030: 2341 9a69 4854 5450 2f31 2e31 2032 3030 #A.iHTTP/1.1.200 |
| 0x0040: 204f 4b0d 0a44 6174 653a 204d 6f6e 2c20 .OK..Date:.Mon,. |
| 0x0050: 3232 204d 6179 2032 3030 3620 3132 3a34 22.May.2006.12:4 |
| 0x0060: 333a 3135 2047 4d54 0d0a 5365 7276 6572 3:15.GMT..Server |
| 0x0070: 3a20 4170 6163 6865 0d0a 4b65 6570 2d41 :.Apache..Keep-A |
| 0x0080: 6c69 7665 3a20 7469 6d65 6f75 743d 3530 live:.timeout=50 |
| 0x0090: 2c20 6d61 783d 3339 380d 0a43 6f6e 6e65 ,.max=398..Conne |
| 0x00a0: 6374 696f 6e3a 204b 6565 702d 416c 6976 ction:.Keep-Aliv |
| 0x00b0: 650d 0a54 7261 6e73 6665 722d 456e 636f e..Transfer-Enco |
| 0x00c0: 6469 6e67 3a20 6368 756e 6b65 640d 0a43 ding:.chunked..C |
| 0x00d0: 6f6e 7465 6e74 2d54 7970 653a 2074 6578 ontent-Type:.tex |
| 0x00e0: 742f 6874 6d6c 0d0a 0d0a 3231 6336 630d t/html....21c6c. |
| 0x00f0: 0a0a 3c73 6372 6970 7420 6c61 6e67 7561 ..<script.langua |

Making an FTP connection to game.org produces reliable information regarding the software and version:

| |
|---|
| 216.251.47.10,21,6,ftp,ProFTPD Server 1.2.8,1137690573 |

Making an ssh connection to a local box on a ephemeral port, again produces reliable information:

| |
|---|
| 192.168.1.100,0,0,ARP (Abit Computer Corporation),0:50:8D:51:9D:3E,1137692974 |
| 192.168.1.100,46221,6,ssh,OpenSSH 3.8.1p1 (Protocol 1.99),1137692977 |

PADS signatures are very basic. They contain a Perl

Compatible Regular Expression (PCRE) which will match for a string. Although simplistic, it has proven able to match reliably. Combining this with a script to analyze the results over a period of time, a sensor can generate a confidence score.

PADS signatures are formatted as follows: <service>,<version info>,<signature>. Each line contains a signature that will detect an application and specify how it should be presented in the PADS assets.csv file. These signatures are stored in /etc/pads/pads-signature-list on a Debian system.

Examples of a few Apache signatures:

| |
|---|
| www,v/Apache/$1//,Server: Apache\/([\S]+)[\r\n] |
| www,v/Apache/$1/$2/,Server: Apache\/([\S]+)[\s]+\((.*)\) |
| www,v/Apache/$1/$2/,Server: Apache\/([\S]+)[\s]+([\S]+) |
| www,v/Apache///,Server: Apache[\r\n] |

The project's webpage [11] is: http://passive.sourceforge.net/

## SourceFire Real-time Network Awareness(RNA)

SourceFire RNA is a commercial appliance that can be utilized in junction with SourceFire IDS/IPS [2]. SourceFire RNA provides extra context, that would normally require a large amount of research on an analyst's part, by proactively gathering asset information [1,6]. It does this by:

- Passive OS Fingerprinting
- Passive Application Mapping
- Passive Network Detection
- Network Topology Mapping
- Asset Profiling
- Asset Vulnerability Status

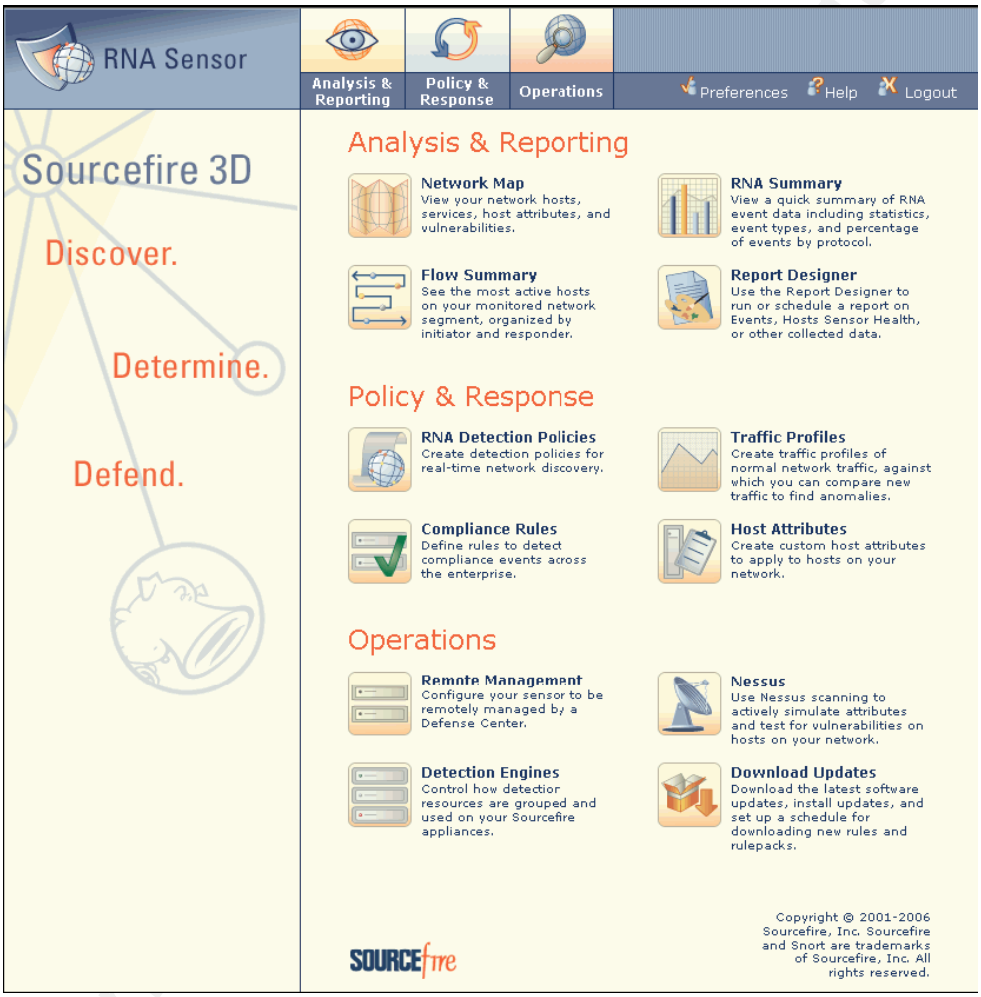
RNA's passive data aggregation gives up-to-date information concerning a host's operating system, the applications it offers, and its network related information [7]. Traditionally, you would have to actively pursue this information by scanning. This method is invasive, can tie-up a large amount of network resources, and can produce undesired results. RNA pro-actively tracks a host's information giving it the ability to detect a change as soon as it takes effect. It does this with “detection engines”. These engines analyze the traffic they can see and attempt to detect “fingerprints”. You can limit this functionality to specific networks to prevent unwanted hosts from being fingerprinted. These fingerprints identify characteristics of the hosts generating traffic. Fingerprints do more than just map applications. The relevant data they aggregate:

- the operating system
- services offered on a server
- the client that a host uses to connect to a server

This information is profiled for each individual host and is stored in the sensor's database. Later, you can configure the sensor to understand how critical a host is. This criticality level allows you to customize how and what alerts are fired. RNA also keeps track of what exploits a host is vulnerable to with this information.

RNA uses a web based console over a secure http connection. Users can customize this web interface based on their preferences. Each user will have it's own permissions that either restrict or enable a them from viewing data, configuring settings, or maintaining the sensor.

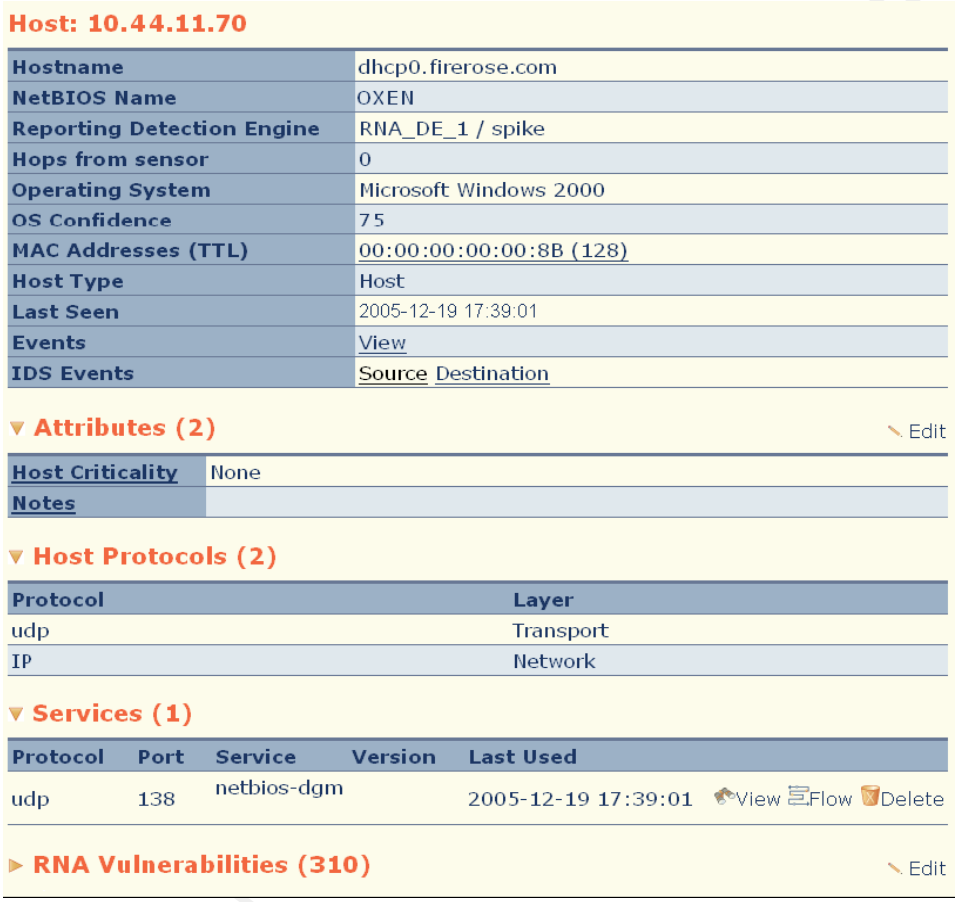
RNA's Default Login Page [9]:




The web console allows users to generate reports, run nessus scans, and manage the sensor. You can also view sensor events, profiles, and RNA statistics.

RNA keeps a profile of all known hosts. These profiles contain, but are not limited to, the hostname, netbios name, distance in hops from the sensor, the host's operating system, and the data and time

the host was last seen generating traffic. From this profile you can view RNA and IDS events related to the host. This profile also contains a list of services the host is known for offering.



**Host: 10.44.11.70**

| | |
|---|---|
| **Hostname** | dhcp0.firerose.com |
| **NetBIOS Name** | OXEN |
| **Reporting Detection Engine** | RNA_DE_1 / spike |
| **Hops from sensor** | 0 |
| **Operating System** | Microsoft Windows 2000 |
| **OS Confidence** | 75 |
| **MAC Addresses (TTL)** | 00:00:00:00:00:8B (128) |
| **Host Type** | Host |
| **Last Seen** | 2005-12-19 17:39:01 |
| **Events** | View |
| **IDS Events** | Source Destination |

**Attributes (2)** Edit

| | |
|---|---|
| **Host Criticality** | None |
| **Notes** | |

**Host Protocols (2)**

| Protocol | Layer |
|---|---|
| udp | Transport |
| IP | Network |

**Services (1)**

| Protocol | Port | Service | Version | Last Used | |
|---|---|---|---|---|---|
| udp | 138 | netbios-dgm | | 2005-12-19 17:39:01 | View Flow Delete |

**RNA Vulnerabilities (310)** Edit

From this profile [9] you can view each individual service by clicking on its “view” link. This will take you to an extended listing of details concerning the service and how that service pertains to the host. This listing contains the service's protocol, port, service name, the service's vendor, the service's version, RNA's confidence on it's determination, the number of times the service has been accessed, and the last recorded instance of it being accessed. The service name, vendor, and version are only displayed if they are known to RNA. Diagram 1.6 contains a chart of services known to RNA.

Diagram 1.6 Default services that RNA can detect [9]:

| |
|---|
| aim - AOL Instant Messenger service (AIM) |
| bgp - Border Gateway Protocol services (BGP) |
| bootps - Bootstrap Protocol services (bootps) |
| dcerpc – Distributed Computing Environment – Remote Procedure Call services (DCE-RPC) |
| domain - Domain Name Service (DNS) servers |
| exec - Remote execution services (exec, rexec) |
| ftp - File Transfer Protocol (FTP) services |
| http - Hypertext Transfer Protocol (HTTP/Web) services |
| imap - Internet Message Access Protocol (IMAP) mail servers |
| imap3 - IMAP version 3 mail servers |
| ircd - Internet Relay Chat services |
| login - Login services (login, rlogin) |
| mysql - MySQL database services |
| netbios-dgm - NETBIOS Datagram Service |
| netbios-ns - NETBIOS Name Service |
| netbios-ssn - NETBIOS Session Service |
| nntp - Network News Transfer Protocol (NNTP) Internet News services |
| ntp - Network Time Protocol (NTP) services |
| pop3 - Post Office Protocol Version 3 (POP3) mail services |
| printer - Printer services |
| radacct - Remote Authentication Dial-In User Service (radius) client services |
| radius - Remote Authentication Dial-In User Service |
| rfb - Remote Frame Buffer services (which may include remote desktop applications such as VNC or Bochs) |
| rsync - File synchronization services (rsync) |
| shell - Shell services (shell, rshell) |
| smtp - Simple Mail Transport Protocol (SMTP) service |
| snmp - Simple Network Management Protocol (SNMP) service |
| ssh - Secure Shell services |
| ssl - Secure Sockets Layer services |
| sunrpc - Sun RPC services (which may include ToolTalk Server, cmsd, status, alis, rexd, mountd, nfs, portmap) |
| telnet - RFC 854 Telnet services |
| tftp - Trivial File Transfer Protocol (TFTP) services |



© SANS Institute 2006, Author retains full rights.

## References


1 Andrew Conry-Murray reviews RNA on itarchitect.com

2 Sourcefire's product information for RNA
http://www.sourcefire.com/products/rna.html

3 Passive Asset Detection(PADS) project page
http://passive.sourceforge.net/

4 Intrusion Detection FAQ: What is p0f and what does it do?
http://www.sans.org/resources/idfaq/p0f.php

5 The new pOf: 2.0.6
1http://lcamtuf.coredump.cx/p0f.shtml

6 Network World review of RNA
http://www.networkworld.com/reviews/2004/0823revrna.html

7 Tools for Defense In-Depth
https://www.sans.org/whatworks/casestudy.php?id=54

8 Security Focus on Passive Network Traffic Analysis
http://www.securityfocus.com/infocus/1696

9 SourceFire 3D Systems: RNA Sensor User Guide

10 SANS Institute
http://www.sans.org/

11 Passive Asset Detection System
http://passive.sourceforge.net/

# Appendix

## a) PADS assets.csv file

| asset,port,proto,service,application,discovered |
|---|
| 192.168.1.100,0,0,ARP (Abit Computer Corporation),0:50:8D:51:9D:3E,1137690463 |
| 192.168.1.100,50022,6,ssh,OpenSSH 3.8.1p1 (Protocol 1.99),1137690466 |
| 216.239.37.104,80,6,www,GWS 2.1,1137690531 |
| 216.251.47.10,21,6,ftp,ProFTPD Server 1.2.8,1137690573 |
| 66.153.203.145,110,6,unknown,unknown,1137690779 |
| 192.168.1.100,0,0,ARP (Abit Computer Corporation),0:50:8D:51:9D:3E,1137692974 |
| 192.168.1.100,46221,6,ssh,OpenSSH 3.8.1p1 (Protocol 1.99),1137692977 |



© SANS Institute 2006, Author retains full rights.