Informative article July, 2003

FTP 'Bounce Attack' Fundamentals.

Stijn Huyghe (stijn.huyghe@thti.telindus.be)

Telindus High-Tech Institute @ http://thti.telindus.be.
Telindus @ http://www.telindus.com.

Target audience: for this article, basic knowledge of networking, FTP and basic network scanning techniques is assumed.

Description: this document is trying to provide the reader with a basic explanation of the so-called 'FTP Bounce Attack'.

Note: the article also provides a basic lab description, so that the reader can experiment in his/her lab using the information obtained through the article.
Together with this article you find an archive file (*.zip) containing all the tools you need to setup the lab and to operate it. You also find in the archive a screen cam movie (in *.avi format) and a freeware player for the Microsoft Windows platform.

FTP 'Bounce Attack'…

The FTP 'Bounce Attack' is a very old technique that is often mentioned in security related books, manuals and courses, but often explained badly or not explained after all without giving examples.
This article is going to explain how a basic FTP 'Bounce Attack' works and offers you an opportunity to experiment with the obtained knowledge it in your own lab.

Basic introduction to (active mode) FTP.

The FTP 'Bounce Attack' is related to some security issues with the FTP (File Transfer Protocol) 'PORT' command in the FTP protocol. It was first introduced to the world in 1995 by the *Hobbit*.

What are the different phases in a FTP session (running in active mode)?
- The FTP client opens a connection to the FTP control port (normally, port 21/TCP) of the FTP server.
A second (data) connection is required to be opened between the server and the client somewhat later to allow the server to send back the data (like files) that the client has requested to the client machine at a later time.
- The client wants to download a file from the FTP server. To establish the data connection (the second one), the client sends out a FTP 'PORT' command to the server machine. This command includes parameters that instruct the server with the IP address and port to connect to in order to deliver the file at the client machine (normally, a free, random port on the client machine).

At this point, during a normal FTP session (active mode), you have:
- One control connection running from the FTP client (random client port number/TCP) to the FTP server (well known port 21/TCP).
- When transferring information, one data connection from the FTP server (chosen port by server/TCP) to the FTP client and it is the server that connects to the FTP client on a port provided by the client through the clients' FTP 'PORT' command.

Core problem.

Conform to the FTP protocol, the 'PORT' command allows the originating client machine to specify an arbitrary destination machine and TCP port for the data connection.
Normally, during a normal FTP session; the FTP client provides the FTP server with its own IP address and chosen port, but it is not obliged to do so – and that is the core problem.
This means that a person can instruct the FTP server to open a connection to a port of a machine that might not be the originating client, offering some possibilities.

FTP 'Bounce Scan' scenarios.

When you have a FTP server or border devices that allow the FTP client to specify an IP address different than its own in the FTP 'PORT' command, you have a insecure situation.

The situation could allow an attacker to carry out a port scan against a host on the Internet from a third-party FTP server that is going to act as an intermediate host. This offers more stealth for the attacker as the victim site will see that the scans originate from the third-party and not from the attacker.

The attacker could also try to bypass filtering devices. If the vulnerable FTP server is standing behind a firewall and when it is reachable from the outside by its rule base, he/she might use the technique to start probing (internal/DMZ) hosts behind the filtering device, bypassing filtering rules.

It gets worse when dynamic packet filtering devices could get bypassed…
Suppose you have a corporation site shielded by a dynamic packet filtering device. An employee browses the web and executes some malicious scripts through his or her vulnerable web browser (when did you perform the last

update of your client software?).
The scripts automatically open up an outbound FTP connection to a FTP server under control of the attacker and a little later it issues a FTP 'PORT' command (active mode) instructing the FTP server to open a connection to the TCP port of a vulnerable service on the client behind the filtering device.
These FTP actions will be noted by the corporation firewall and triggers a reaction from the firewall to allow the FTP session to be established, but in this case, it could allow the attacker to connect to the client, bypassing its rules.

Example attack.

This section will show you a basic setup of a lab and the results, so that you can experiment yourself.

Network scheme.

The lab network is constructed of three interconnected hosts:
- The target machine (10.0.18.35) is a Windows 2000 server, offering FTP (21/TCP) and HTTP services (80/TCP) to the world.
- The FTP Bounce host (10.0.18.44) is running a vulnerable FTP service (ArgoSoft FTP server*) with either anonymous access or a named account. It is essential for the attacker to have basic access to the FTP server in order to succeed in the attack. In our lab, the attacker obtained a named account for user "sthuy" in advance.
- The attacker is located on IP address 10.0.18.30, running Red Hat Linux 9 with Nmap 3.20.

*ArgoSoft FTP server (freeware): most FTP services (and firewall devices) running today include countermeasures (or should include countermeasures) against the old FTP 'Bounce Attack'.
Protected FTP services do not allow the client to specify another IP address than its own and do not allow the server to connect to ports < 1024. For the lab to work with the ArgoSoft FTP server, you need to disable it explicitly in it's configuration (see lower in the document).

The results of the attack.

The result of the port scan using the FTP 'Bounce Attack' technique can be found in figure 1 and shows you the results as seen on the screen of the attacker. Ports 21/TCP (FTP) and ports 80/TCP (HTTP) are being indicated as open ports, which is correct.

```
[root@pretorian nmap-3.20]# nmap -b sthuy:hexid@10.0.18.44 -p 80,8080,21,666 10.0.18.35
Hint: if your bounce scan target hosts aren't reachable from here, remember to use -P0 so we
 don't try and ping them prior to the scan

Starting nmap 3.20 ( www.insecure.org/nmap/ ) at 2003-04-30 13:42 CEST
Interesting ports on 10.0.18.35:
Port       State       Service
WARNING!  The following files exist and are readable: /usr/local/share/nmap/nmap-services an
d ./nmap-services.  I am choosing /usr/local/share/nmap/nmap-services for security reasons.
 set NMAPDIR=. to give priority to files in your local directory
21/tcp     open        ftp
80/tcp     open        http

Nmap run completed -- 1 IP address (1 host up) scanned in 31.112 seconds
```

Figure 1 Results seen on the screen of the attacker.

In figure 2, you can find the obvious traces of the FTP 'Bounce Scan' in the log files of the server.
In the log file, you see the attacker logging into the FTP server using the credentials of the user "sthuy" and his password on the FTP server.
After the login sequence, the attacker starts issuing FTP 'PORT' commands, specifying the ports he/she wants to scan and the IP address of his target host.
E.g. 30/04/2003 13:16:15 – (1) PORT 10.0.18.35,0,80
At this log line, the attacker located at 10.0.18.30 issues a 'PORT' command to the target host (10.0.18.35) at TCP port 21.

Also note in the FTP server log file the information "226 Transfer complete" or "426 File Transfer Aborted.[100611] Connection Refused". This is vital information for Nmap to be able to indicate whether the ports scanned through the intermediate host (ftp server) are open (allowed initial TCP connection) or closed (did not allow initial TCP connection to be established).

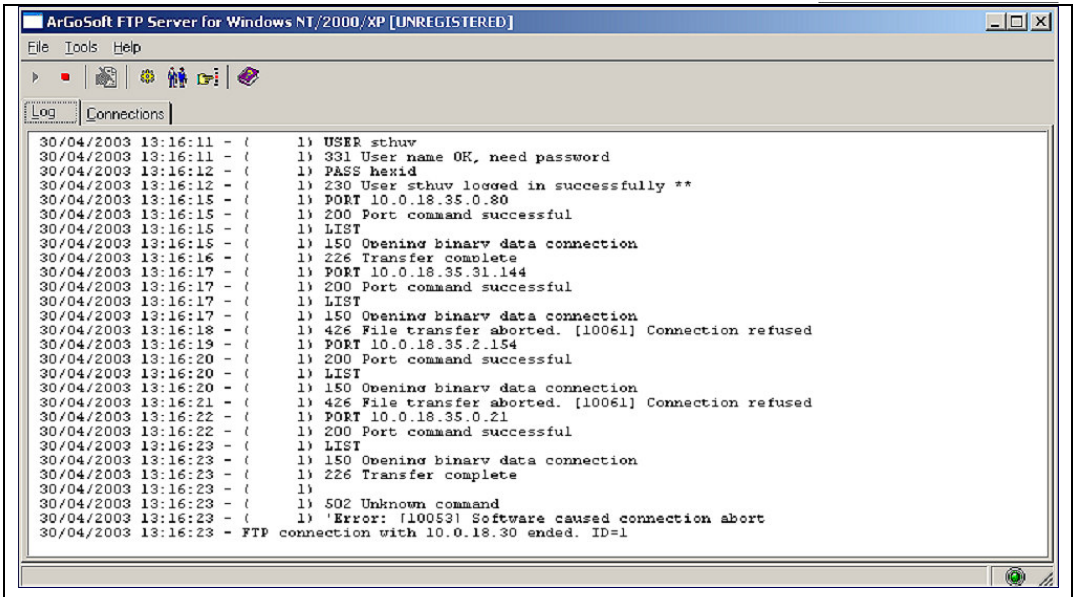

Figure 2 Log files on the FTP server during the attack.

Lab instructions for your own lab.

Make sure that you have three hosts that you can use:
- One target host running some services you are going to probe for.
- One *nix system running Nmap.
- One vulnerable FTP host (like ArgoSoft FTP server*, notes: see “network scheme” in this document).

Nmap setup.
You can get Nmap from http://www.insecure.org for several platforms. It is advised to go with the *nix versions.
For the installation of the latest Nmap versions, you need to be able to extract *bzip2* compressed files. For more information on bzip2: http://sources.redhat.com/bzip2.
Basic installation procedures can be found in figure 3. Make sure that you read through the Nmap documentation and MAN pages.
Note: the “file” command is used in *nix to determine the file type of a file.

```
[root@pretorian stijn]# file nmap-3.20.tar.tar
nmap-3.20.tar.tar: bzip2 compressed data, block size = 900k
[root@pretorian stijn]# bzip2 -d nmap-3.20.tar.tar
bzip2: Can't guess original name for nmap-3.20.tar.tar -- using nmap-3.20.tar.te
r.out
[root@pretorian stijn]# file nmap-3.20.tar.tar.out
nmap-3.20.tar.tar.out: GNU tar archive
[root@pretorian stijn]# tar -xvf nmap-3.20.tar.tar.out
```

Figure 3 Basic Nmap installation.

ArgoSoft Setup.

Make a default installation of the FTP server software, create a user account (named/anonymous) and remove the FTP security functionality for the lab. Removing FTP security functionality can be found in figure 4.
Make sure that logging is enabled and that you modify the port ranges for data transfers!

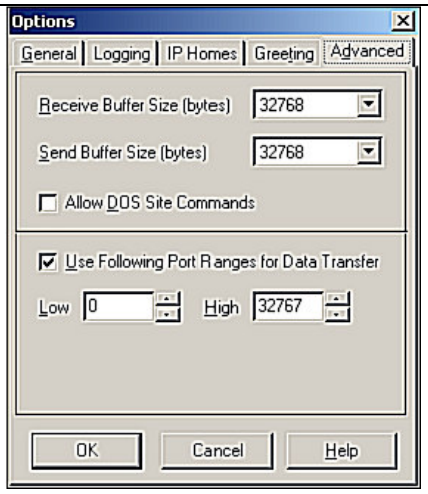

Figure 4 Configuring ArgoSoft FTP server for the lab.

Appendix A: Nmap MAN page on the FTP 'Bounce Attack':

You can find more information on Nmap and related topics on http://www.insecure.org.

```
-b <ftp relay host>
       FTP  bounce attack: An interesting "feature" of the ftp protocol
       (RFC 959) is support  for  "proxy"  ftp  connections.  In  other
       words,  I  should  be  able  to connect from evil.com to the FTP
       server of target.com and request that the  server  send  a  file
       ANYWHERE on the internet!  Now this may have worked well in 1985
       when the RFC was written. But in todayâs Internet, we canât have
       people  hijacking  ftp  servers and requesting that data be spit
       out to arbitrary points on the internet. As *Hobbit* wrote  back
       in  1995,  this  protocol  flaw  "can  be used to post virtually
       untraceable mail and news, hammer on servers at  various  sites,
       fill  up  disks, try to hop firewalls, and generally be annoying
       and hard to track down at the same time." What we  will  exploit
       this  for  is  to  (surprise,  surprise)  scan  TCP ports from a
       "proxy" ftp server. Thus you could  connect  to  an  ftp  server
       behind  a  firewall, and then scan ports that are more likely to
       be blocked (139 is a good one). If the ftp server allows reading
       from  and writing to some directory (such as /incoming), you can
       send arbitrary data to ports that you do find open (nmap doesnât
       do this for you though).

       The  argument  passed  to the Â´bÂ´ option is the host you want to
       use as a proxy, in standard URL notation.  The format is:  user-
       name:password@server:port.   Everything  but  server  is optional.
       To determine what servers are vulnerable to this attack, you can
       see  my  article in Phrack 51.  And updated version is available
       at the nmap URL (http://www.insecure.org/nmap).
```

Appendix B: playing the FTP 'Bounce Attack' screen cam.

The screen cam is recorded in *.avi format. For Microsoft Windows users it is advised to use the CamStudio Movie Player 2.1 by RenderSoft (freeware). You can find CamStudio at http://www.rendersoftware.com/products/camstudio/.
In the screen cam you can see a basic FTP Port scan attack using the same scenario as described in this article.