Analysis of the File Transfer Protocol (FTP)

by Priscilla Oppenheimer

There are many myths about how FTP works. The goal of this white paper is to briefly describe FTP protocol processes to dispel some of the myths. More detailed information about FTP behavior is available in Chapter 9 of the **Troubleshooting Campus Networks** book by Priscilla Oppenheimer and Joseph Bardwell.

FTP was one of the first Internet protocols. It was designed for use on the Internet when the Internet was still a closed system that connected universities, government agencies, and a few commercial companies involved in the development of the system. FTP's behavior on modern networks, especially networks where security is a big concern, is problematic. This paper provides the theoretical background you will need to get FTP working properly on a modern network that uses firewalls and Network Address Translation (NAT).

FTP was designed long before the term *client/server* came in vogue, but it does behave like a client/server protocol nonetheless. FTP uses two TCP connections, one for control information and one for the actual data. Analysis of an FTP session involves an examination of the FTP commands sent on the control connection and an assessment of the TCP segments sent on the data connection. With *normal* or *active FTP*, the control connection is initiated by the client side and the data connection is initiated by the server side. (Active mode is also sometimes called *port mode*). The other option is *passive FTP*, in which case the client side establishes the data connection.

FTP and TCP Port Numbers

FTP uses different TCP port numbers depending on whether active or passive FTP is in use. Before we cover FTP in more detail, we'll briefly discuss some basic concepts regarding TCP port numbers. TCP uses port numbers to identify the sending and receiving application. A port number helps TCP demultiplex byte streams and deliver bytes to the correct application.

TCP ports can be semi-permanent or ephemeral (short-lived). Servers listen on the semi-permanent ports for clients wishing to access services. Clients use ephemeral ports to identify their end of a conversation. The client side only lasts while the client is using a service, whereas a server port is usually open the entire time that a server is running.

TCP port numbers also fall into these three categories:

- Well-known port numbers are used to identify standard services that run above TCP, including FTP, HTTP, Telnet, SMTP, and so on. Well-known port numbers are 0 to 1,023.
- Registered port numbers identify an application that has been registered with the Internet Assigned Numbers Assigned Numbers Authority (IANA). Registered port numbers are 1,024 to 49,151.
- Private port numbers are unregistered and can be dynamically assigned to any application. Private port numbers are 49,152 to 65,535.

A registered port number is intended for use by only the registered application. However, you will see port numbers that are supposedly "registered" get used as an ephemeral port by applications that are not related to the registered application. You can get an up-to-date list of port numbers from IANA here.

FTP Active Versus Passive Mode

When troubleshooting FTP problems, one of the first questions you should ask is whether active or passive mode is in use. Because their behaviors are quite different, the two modes experience different problems. In the past, client implementations defaulted to active mode. These days, many client implementations default to passive mode due to security concerns with active mode.

FTP Active Mode

The steps for active FTP are as follows:

1. The client sends a TCP SYN to the well-known FTP control port (port 21) on the server. The client uses an ephemeral port as its source port.
2. The server sends the client a SYN ACK from port 21 to the ephemeral port on the client.
3. The client sends an ACK. The client uses this connection to send FTP commands and the server uses this connection to send FTP replies.
4. When the user requests a directory listing or initiates the sending or receiving of a file, the client software sends a PORT command that includes an ephemeral port number that the client wishes the server to use when opening the data connection. The PORT command also includes an IP address, which is usually the client's own IP address, although FTP also supports a third-party mode where a client can tell a server to open a connection with a different host.
5. The server sends a SYN from port 20 to the client's ephemeral port number, which was provided to the server in the client's PORT command.
6. The client sends a SYN ACK from its ephemeral port to port 20.
7. The server sends an ACK.
8. The host that is sending data uses this new connection to send the data in TCP segments, which the other host ACKs. (With some commands, such as STOR, the client sends data. With other commands, such as RETR, the server sends data.)
9. After the data transfer is complete, the host sending data closes the data connection with a FIN, which the other host ACKs. The other host also sends its own FIN, which the sending host ACKs.
10. The client can send more commands on the control connection, which may cause additional data connections to be opened and then closed. At some point, when the user is finished, the client closes the control connection with a FIN. The server ACKs the client's FIN. The server also sends its own FIN, which the client ACKs.

Figure 1 shows a graphical representation of the first few steps of FTP active mode.

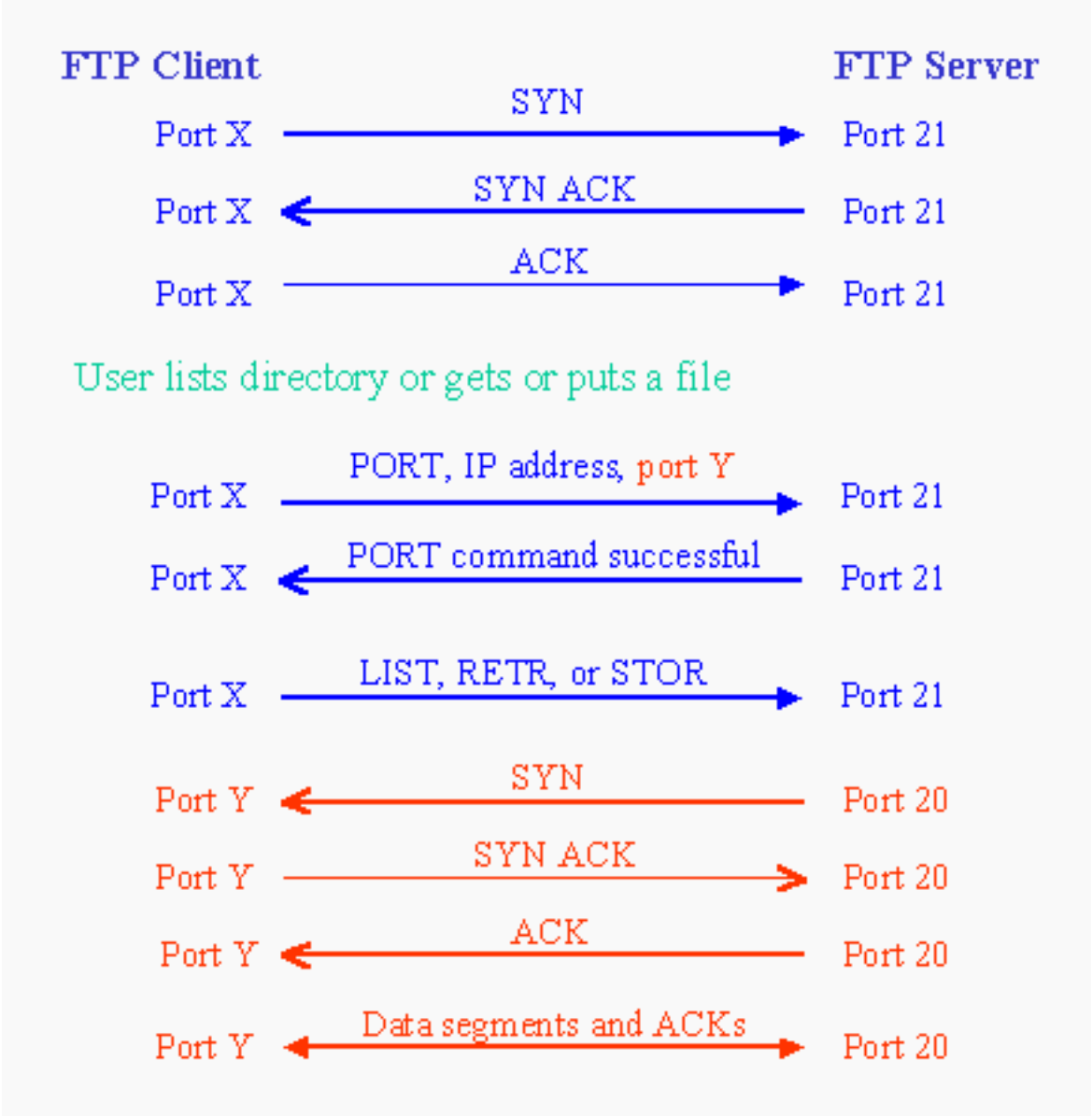

Figure 1. FTP active mode.

The FTP PORT Command

The FTP PORT command causes problems for network support engineers in many ways. For one thing, the encoding of the IP address and port number in a PORT message is not straightforward. In addition, an application-layer protocol command theoretically shouldn't include network-layer information (an IP address). This breaks the principles of protocol layering and can cause both interoperability and security problems. Figure 2 shows an example of an FTP PORT command.

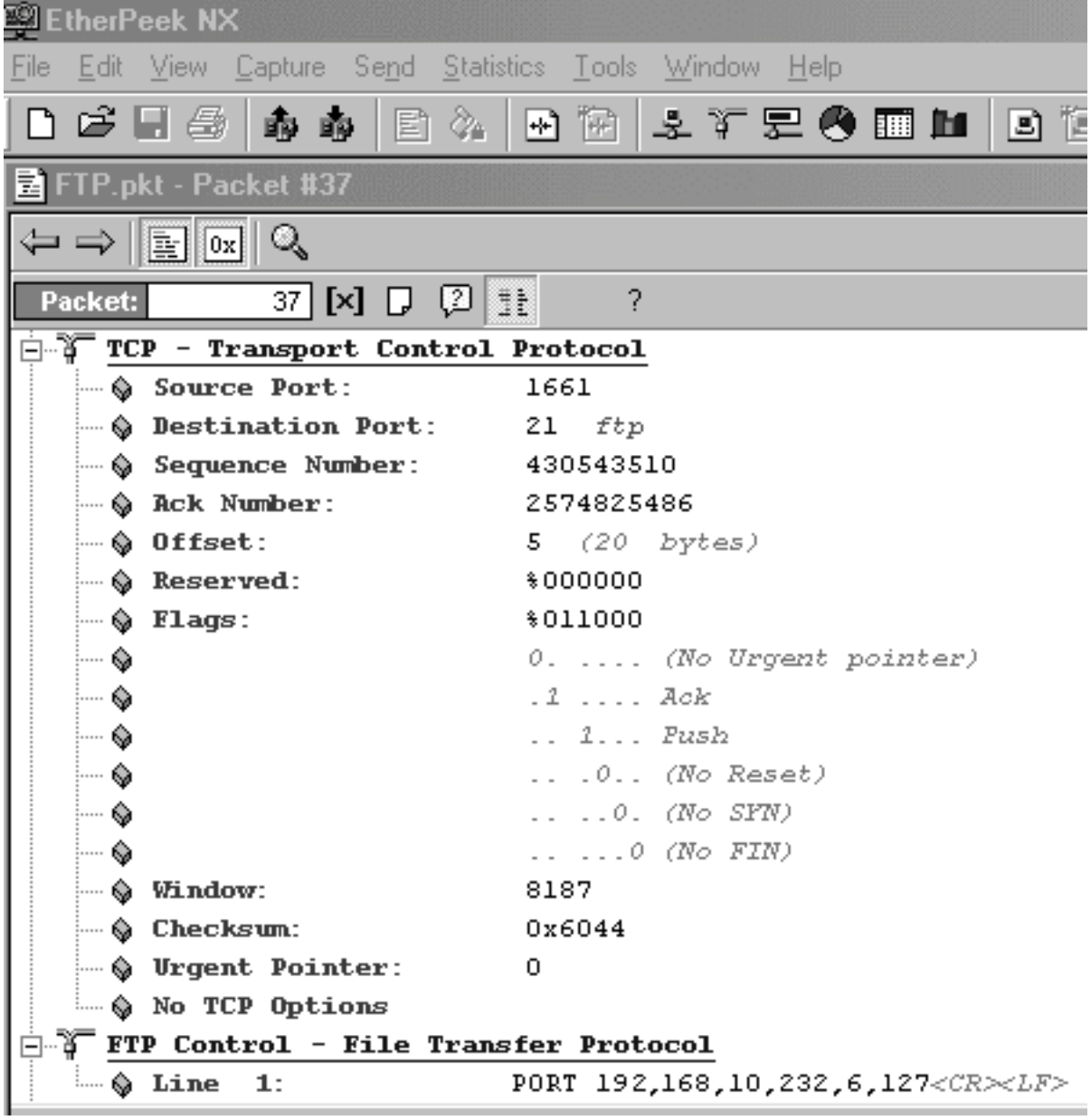

Figure 2. An FTP PORT command from a client to a server.

In the detail view of the packet shown in Figure 2, the WildPackets EtherPeek protocol analyzer decodes the IP address parameter for the PORT command, followed by the port number. (See PORT 192,168,10,232,6,127.) The 6, 127 portion becomes a port number by multiplying the first digit by 256 and adding the second digit. So the client specified a port number that is (6 x 256) + 127, which equals 1663. Figure 3 verifies that the server did indeed open a TCP connection from port 20 to port 1663.

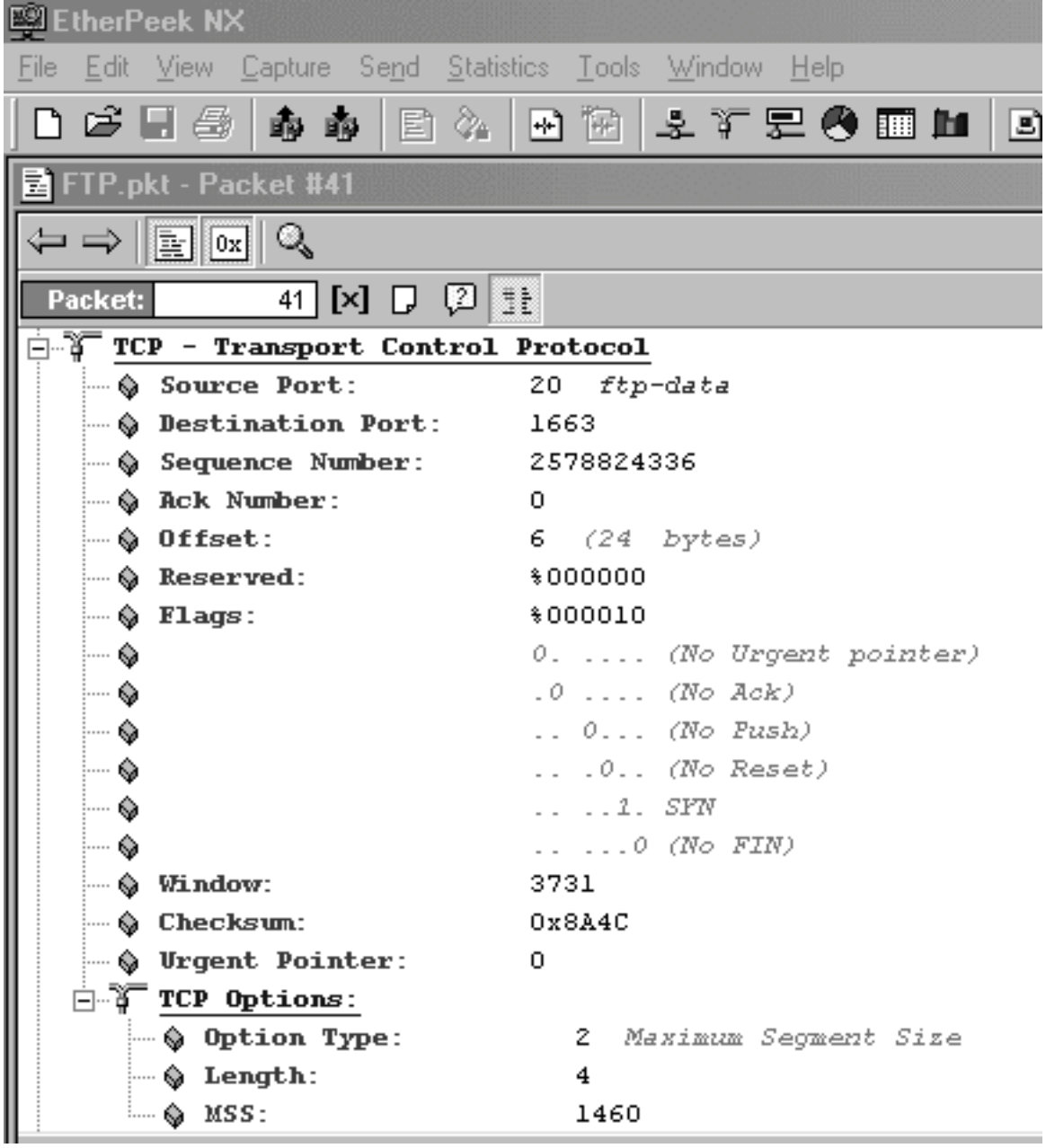

Figure 3. A server in active mode opens a TCP connection for data transfer.

When FTP is used, network firewalls must be "stateful." The firewall must track FTP sessions and be on the lookout for FTP PORT commands. The firewall needs to anticipate the connection establishment coming from the FTP server into the client's port specified in the PORT command. If the network uses NAT, the NAT gateway must be stateful also. The gateway needs to translate the IP address in the FTP PORT command to the address assigned to the client, and then recalculate the TCP checksum. If the gateway doesn't perform these operations correctly, FTP can fail.

Hackers can exploit the third-party feature of FTP by setting the IP address and port number parameters in the PORT command to a target host's address and port numbers (sometimes called an *FTP bounce attack*). For example, a hacker can cause an FTP server to continually send TCP SYN packets from port 20 to a set of target ports, making it look like the server is initiating a port scan. The target won't know that the attack is coming from the hacker's machine. It will appear that it's coming from the FTP server.

Some commonly-used FTP implementations set the IP address in the PORT command to 0.0.0.0, with the intention being that the server should simply open the data connection on the same client that opened the control connection. Setting the IP address to 0.0.0.0 can confuse firewalls. For exmple, with Cisco IOS release 6.x(x), the PIX fixup protocol for FTP does not allow the IP address for the data connection to be different from the one already in use for the control connection. The reason for this is to thwart a hacker who could use the PORT command to launch an attack on another host. Although the FTP implementations that set the IP address to 0.0.0.0 are not intentional hacks, they do trigger a problem with the fixup protocol and other firewalls that were designed to disallow FTP third-party mode and to avoid FTP bounce attacks.

FTP Passive Mode

The steps for passive FTP are described in the following list. Steps 1 through 3 are the same as the first three steps for active mode. Also, steps 9 through 11 are the same as the last three steps for active mode.

1. The client sends a TCP SYN to the well-known FTP control port (port 21) on the server. The client uses an ephemeral port as the source port.
2. The server sends the client a SYN ACK from port 21 to the ephemeral port on the client.
3. The client sends an ACK. The client uses this connection to send FTP commands and the server uses the connection to send FTP replies.
4. When the user requests a directory listing or initiates the sending or receiving of a file, the client software sends a PASV command to the server indicating the desire to enter passive mode.
5. The server replies. The reply includes the IP address of the server and an ephemeral port number that the client should use when opening the connection for data transfer.
6. The client sends a SYN from a client-selected ephemeral port to the server's ephemeral port number, which was provided to the client in the reply to the client's PASV command.
7. The server sends a SYN ACK from its ephemeral port to the client's ephemeral port.
8. The client sends an ACK.
9. The host that is sending data uses this new connection to send the data in TCP segments, which the other host ACKs. (With some commands, such as STOR, the client sends data. With other commands, such as RETR, the server sends data.)
10. After the data transfer is complete, the host sending data closes the data connection with a FIN, which the other host ACKs. The other host also sends its own FIN, which the sending host ACKs.
11. The client can send more commands on the control session, which may cause additional data connections to be opened and then closed. At some point, when the user is finished, the client closes the control connection with a FIN. The server ACKs the client's FIN. The server also sends its own FIN, which the client ACKs.

Figure 4 shows a graphical representation of the first few steps of FTP passive mode.

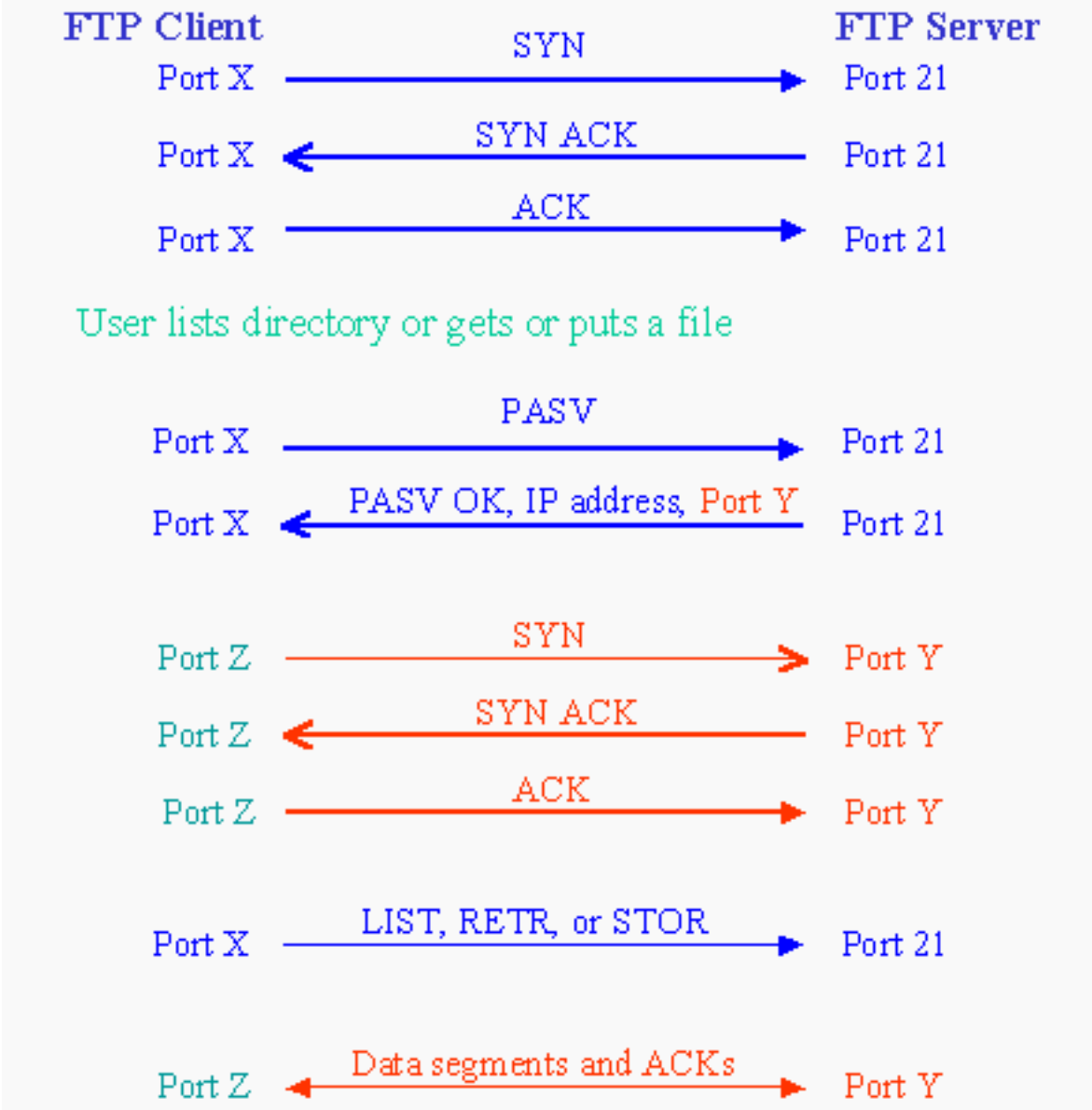

Figure 4. FTP passive mode.

The FTP PASV Command

A PASV request asks the server to accept a data connection on a new TCP port selected by the server. There are no parameters to the PASV command. The server's response is a single line showing the IP address of the server and the TCP port number where the server is accepting connections.

Figure 5 shows a server's reply to a client's PASV command. The server tells the client that the server is listening on port 5365. (See 192,168,179,100,20,245). To calculate the port number, multiply 20 times 256 and add 245. (20 x 256 +245 = 5365.)

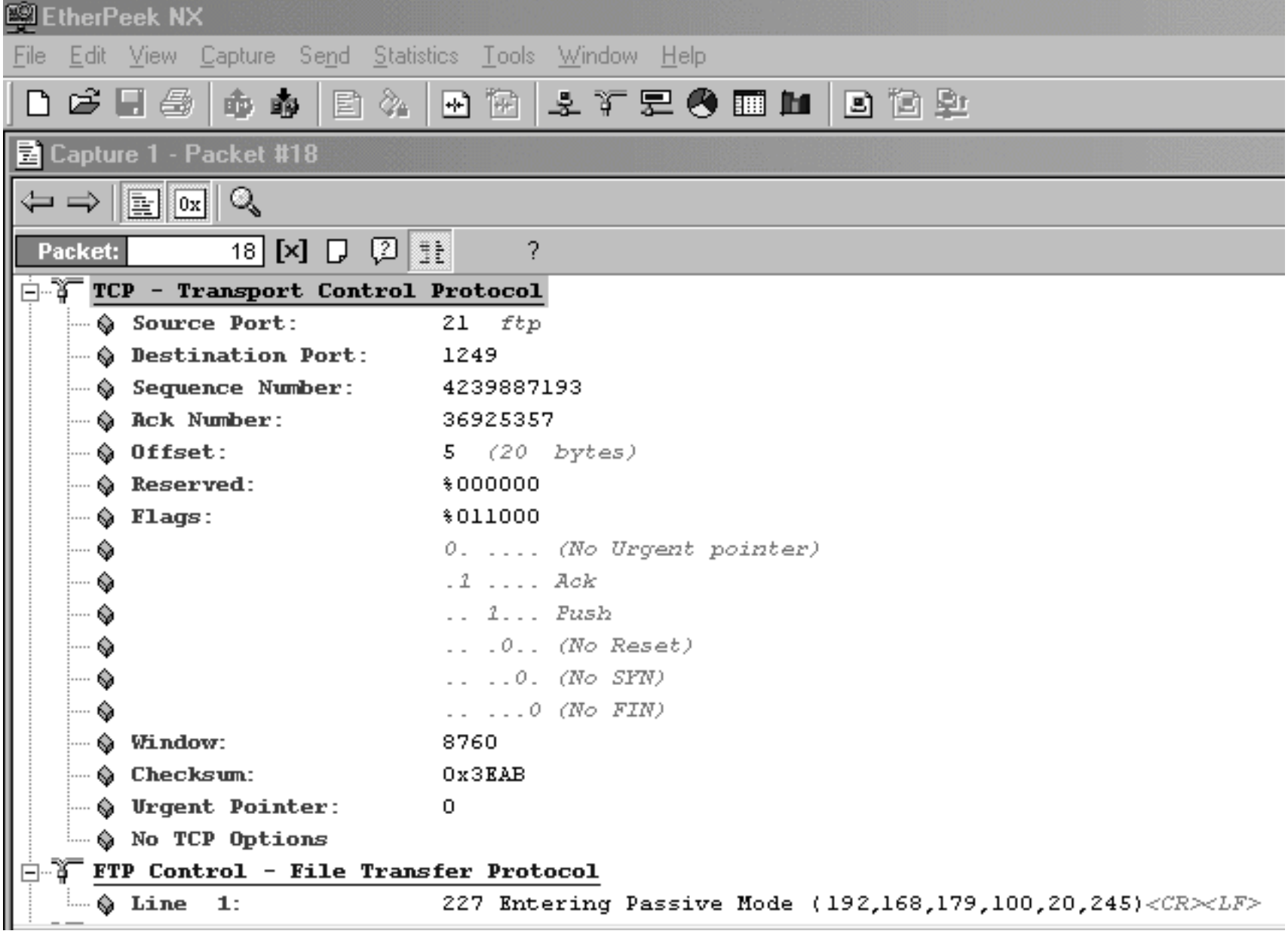

Figure 5. A server's reply to a client's PASV command.

After receiving a reply to its PASV command, the client opens a TCP connection from an ephemeral port to the port number supplied by the server. Figure 6 shows the TCP connection establishment from the client. Notice that the destination port is 5365, as expected.

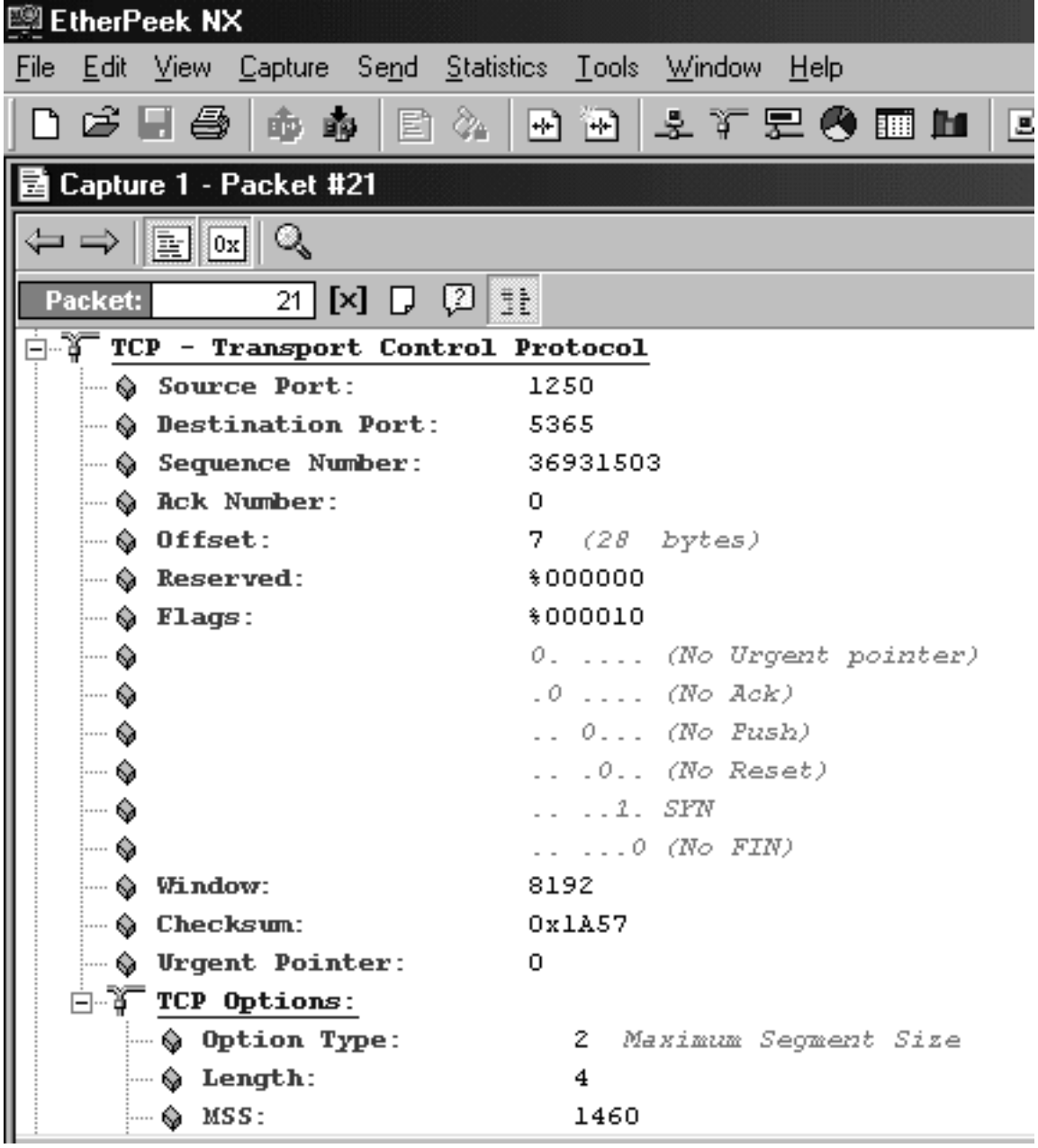

Figure 6. A client in passive mode opens a TCP connection for data transfer.

Most people assume that passive mode causes fewer problems for firewalls than active mode, but note that the client opens a connection to an ephemeral (not well-known) port when using passive mode. Some firewalls and Cisco access lists block this. Also, the response from the server comes from an ephemeral port and goes to an ephemeral port. Firewalls and Cisco access lists might block this also. On a Cisco router you can use the **established** keyword with an access list to avoid this second problem. The **established** keyword tells the router to allow packets with the ACK bit set. The SYN ACK from the server has the ACK bit set.

Clear Text Username and Password

Another infamous problem with FTP is that it sends the username and password as clear text, that is, unencrypted. Anyone with a strategically-placed protocol analyzer can see the name and password. FTP sends the data in the files being transfered as clear text also. This might not seem like a huge problem because it's difficult to place a protocol analyzer in the correct place to glean the unencrypted information, but it's not impossible to do so. The problem is made worse by the fact that users tend to use the same password for many different applications. If hackers glean an FTP password, they may be getting a password that is also used to access online checking accounts or other confidential data.

Alternatives to FTP

In this day and age, there's probably no excuse for using FTP any more, considering that there are other more secure options for file transfer. Both SCP and SFTP, for example, are similar in function to FTP but use Secure Shell (SSH) authentication and encryption. If you use a Unix-based server, you should be able to

invoke *scp* or *sftp* from the command line. For more information about SecureShell, see the OpenSSH Web site.

If your use for FTP is limited to updating Web pages, there is another alternative called Web-based Distributed Authoring and Versioning (WebDAV). WebDAV is a set of extensions to the Hypertext Transfer Protocol (HTTP) to allow users to collaboratively edit and manage files on remote Web servers. See RFC 2518 for more information.

Summary

FTP was designed in the 1970s. At that time, the Internet was a closed network and security was not a big concern. When FTP is used in modern environments with NAT gateways, firewalls, and Cisco access lists, problems can arise, whether you use active or passive mode. FTP is often used for mission-critical applications on the public Internet, which is probably a mistake. There have been many attempts to make FTP more secure. These attempts cause troubleshooting issues and also fail to fix the most glaring security problem with FTP, which is that it sends the username and password as clear text. Many alternatives to FTP are available, including SCP, SFTP, and WebDAV.

Other FTP Resources

Bernstein, D. J., **FTP: File Transfer Protocol**.

CERT Coordination Center, **Problems with the FTP PORT Command or Why You Don't Want Just any PORT in a Storm**.

Postel, J. and J. Reynolds, **RFC 959** (the official specification for FTP).
