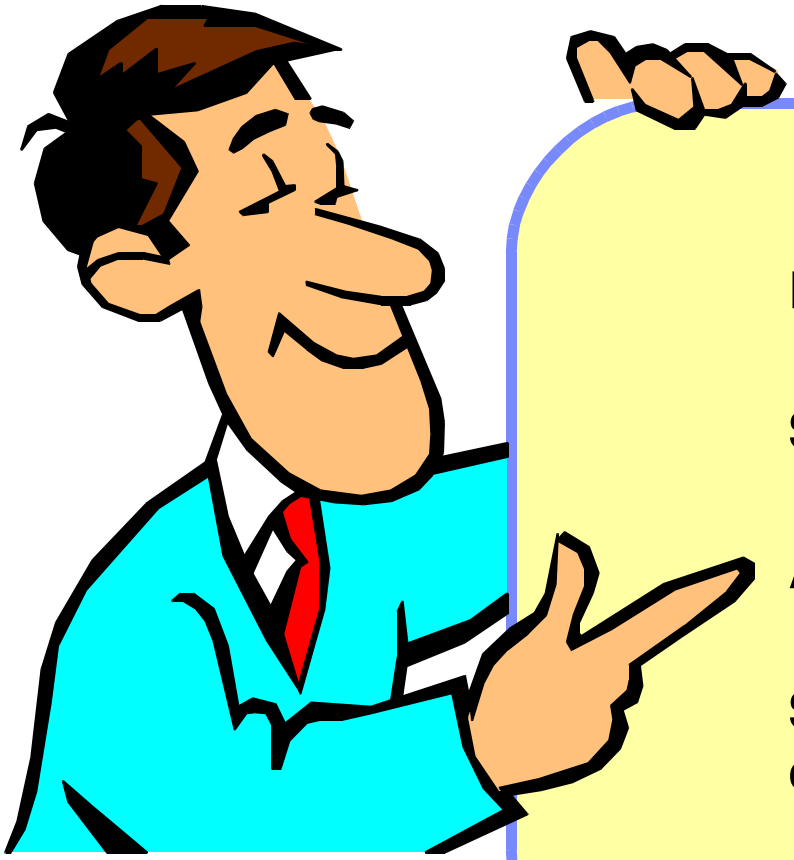
Safe and Secure Transfers with z/OS FTP

Original presentation by Alfred B Christensen
Modified by Andy Tracy - andyt@us.ibm.com
December 2013

Agenda



- Base FTP protocol review
- Secure FTP Session startup
- A few words about certificates
- Secure FTP and network traversal challenges and solutions

Base FTP protocol review

Let's clear a little confusion from start

- **FTP or RFC959 FTP or "normal" FTP:**
  - The FTP protocol we all know and have used for years
  - What the z/OS CS FTP client and server support
    - An RFC959 FTP client talks to an RFC959 FTP server, and not to an sftp server
  - Sometimes referred to as normal FTP
    - Mostly because its been around for ever and much longer than any of its alternatives

- **sftp:**
  - Secure Shell file transfer protocol
    - A sub-protocol of SSH (Secure Shell)
    - Supported on z/OS by "IBM Ported tools for z/OS"
    - Has nothing to do with RFC959 FTP - incompatible protocols
    - An sftp client talks to an sftp server and not an RFC959 FTP server

- **ftps or ftp auth-tls or ftp auth-ssl:**
  - Secure RFC959 FTP using a standard security mechanism, such as Kerberos or SSL/TLS
  - The normal FTP protocol but with full network security (authentication and confidentiality)

- **Is normal FTP today a secure technology?**
  - The RFC959 FTP protocol has been extended numerous times since the original RFC 959 was issued in 1985
    - Specific support for both Kerberos-based and SSL/TLS-based security has been added
      - RFC4217 "Securing FTP with TLS"
  - The FTP protocol today includes numerous security features that, when enabled and implemented correctly, do make FTP a secure file transfer technology

FTP is as secure as you set it up to be

A little FTP history lesson

- **The FTP protocol is a convenient protocol to use for moving data between a variety of hardware and software platforms**
  - FTP has been around since early 1970 - earliest attempts at an FTP RFC are from 1971
  - There are FTP clients and servers for literally every single operating system environment available to us
    - There are thousands of products that implement elements of or the full FTP protocol standard
      - FTP client functions are imbedded into WEB browsers and numerous GUI tools
  - FTP is simple to use
    - Especially with one of a multitude of GUI-based FTP client tools on Windows, Linux, UNIX, etc.

- **FTP was originally designed back when networks were local and somewhat isolated**
  - FTP was designed with maximum flexibility in mind
    - different operating systems,
    - different file systems,
    - ability to control transfers between remote computers from a single control point (3-way proxy),
    - etc.
  - There was no need for authentication beyond user ID and password
  - There was no need for encryption of the data
  - The concept of "firewalls" did not exist
  - No one could imagine that IPv4 addresses one day would run out and spawn the need for something like Network Address Translation (NAT)

- **Firewalls initially addressed the peculiar behavior of the FTP protocol by implementing technologies that depended on the ability to peek into and even change the FTP protocol exchanges**
  - Because of this approach, it has been very difficult to implement security technologies that encrypt the FTP protocol exchanges end-to-end

File Transfer Protocol (FTP) - revisited

There is both an FTP client and an FTP server on z/OS. The client can be used from TSO, the UNIX shell, batch jobs, or any z/OS program using the FTP client API (C, REXX, Java, Assembler, Cobol, PL/I, etc.)

FTP uses two separate TCP connections to transfer a file:

1. One is the ***control connection*** that is used for exchange of FTP commands and replies. Standard FTP server port is 21. This connection stays up during the whole FTP session.
2. The other is the ***data connection*** that is used for the actual data transfer. Stays up for the duration of a single file transfer.

This is **not** a standardized interface

FTP Client

User Interface
Graphical or command line.

Client **protocol interpreter**

Client **data transfer** process

This is the standardized FTP protocol interface (RFC 959 ++)

Control Connection
FTP commands and replies

Data Connection
Data transfer

FTP Server

Server **protocol interpreter**

Server **data transfer** process

Client file system

Server file system

Copying files between the two file systems

FTP – Commands and replies

- **The FTP protocol RFCs define commands and replies.**
  - FTP client sends commands and server sends replies
    - Reply codes
    - 3 digit reply code followed by human readable text
    - Indicate success or failure of previous FTP command
    - 3 digit code can be parsed by client and automated
    - 3 digit reply codes
      - 1xx - Positive preliminary reply
      - 2xx - Positive completion reply
      - 3xx - Positive intermediate reply
      - 4xx - Transient negative completion reply
      - 5xx - Permanent negative completion reply
    - A '-' indicates more replies will follow with the same code
      - 220-FTPD1 IBM FTP CS V2R1 at MVS054
      - 220 Connection will close if idle for more than 12 minutes.

- This is a successful login. Most servers require userid/password

```
EZA1554I Connecting to: localhost.raleigh.ibm.com 127.0.0.1 port: 21.
220-FTPD1 IBM FTP CS V2R8 at MVS054, 19:52:20 on 2013-06-18.
220 Connection will close if idle for more than 12 minutes.
EZA1459I NAME (loopback:TIFFANY):
EZA1701I >>> USER tiffany
331 Send password please.
EZA1789I PASSWORD:
EZA1701I >>> PASS
230 TIFFANY is logged on.  Working directory is "TIFFANY".
```

FTP – Commands and replies

- **There are many FTP commands. These are the most widely used ones**
  - •PWD - Print working directory
    EZA1736I pwd
    EZA1701I >>> PWD
    257 "'TIFFANY.'" is working directory.
  - •SITE and LOCSITE commands
  - • Used to prepare the environment for the file to be transferred
  - • For MVS, sets up Recfm, Lrecl, Blksize, etc...
    EZA1736I site recfm=vb lrecl=133 blksize=137
    EZA1701I >>> SITE recfm=vb lrecl=133 blksize=137
    200 Site command was accepted
  - •TYPE, MODE and STRU commands control how files are transferred
    - •TYPE - ASCII , Image(Binary), EBCDIC, or DBCS
    - •MODE - STREAM, Block, Compressed
    - •STRU - FILE or Record
  - •Most transfers use Type ASCII, Mode Stream, and Stru File
  - •PUT - Send file to remote machine (STOR)
    - •Syntax: PUT local.file remote.file
  - •GET - Retrieve file from remote machine (RETR)
    - •Syntax GET remote.file local.file (REPLACE
  - •NLST/LIST (LS/DIR)
    - •Causes server to list files in current directory

FTP – Data connections

- **A data connection is set up when user enters LS, DIR, PUT or GET command**
  - The FTP client and server must complete another TCP connection.
  - Client sends either a PORT or PASV command to set up connection

  •PORT a,b,c,d,x,y

    •Tells the server what IP address and port the client is listening on. The server will connect back to this IP address and port from port 20.

      •IP address is a.b.c.d. The port is x*256 + y.

      •PORT 127,0,0,1,6,81 = 127.0.0.1, port 1617

  •PASV

    •PASV tells the server to listen on a port and send the IP address and port to the client. Same algorithm as Port command

      •227 Entering Passive Mode (127,0,0,1,4,4)

    •Client will initiate data connection to the server. PORT a,b,c,d,x,y

Secure FTP session setup

Two ways to initiate an SSL/TLS FTP session

- **There are two ways to indicate if an FTP session is to use SSL/TLS or not:**
  - ***Explicit mode*** (also known as AUTH SSL or AUTH TLS mode)
    - FTP client connects to usual FTP server port 21 and sends an FTP command (AUTH) to request use of SSL/TLS
    - This mode is defined in RFC 4217
    - This is the recommended mode according to the RFC standards
    - The FTP server may have both secure and non-secure connections on the same port
  - ***Implicit mode*** (also known as SSL direct or FTPS mode)
    - FTP client connects to an alternate FTP server port (for example, port 990) and the client and server implicitly enter SSL/TLS mode as a result of the connection being established to that alternate port number
    - There are no RFC standards that govern how implicit mode FTP sessions are to be set up
    - Use of implicit mode FTP is generally based on how the original Netscape implementation worked
    - z/OS FTP originally chose an alternative method
- **You configure the z/OS FTP server to use implicit mode by specifying the TLSPORT option:**
  - TLSPORT 0 — Disable the protected port - no port is implicitly secured with TLS.
  - TLSPORT port — Specify the protected port - port 990 is the default value.
- **You configure the z/OS FTP server to use explicit mode by specifying the SECURE_FTP option (along with other security-related options):**
  - SECURE_FTP ALLOWED — Clients are allowed to send an AUTH command
  - SECURE_FTP REQUIRED — Clients must send an AUTH command

FTP protocol extensions for explicit secure FTP

- "FTP Security Extensions", RFC2228 - defines a set of new FTP protocol commands and replies for negotiating secure FTP sessions. RFC 4217 documents how to use these commands to implement SSL/TLS security.

- The commands and replies are generic and are used to implement both Kerberos-based and SSL/TLS-based secure FTP sessions.

- AUTH
  - Sent by client to server with information about which security mechanism the client requests to use. Supported values for z/OS are GSSAPI (used with Kerberos V5 support) and TLS, TLS-C, TLS-P, and SSL (used with SSL/TLS support)

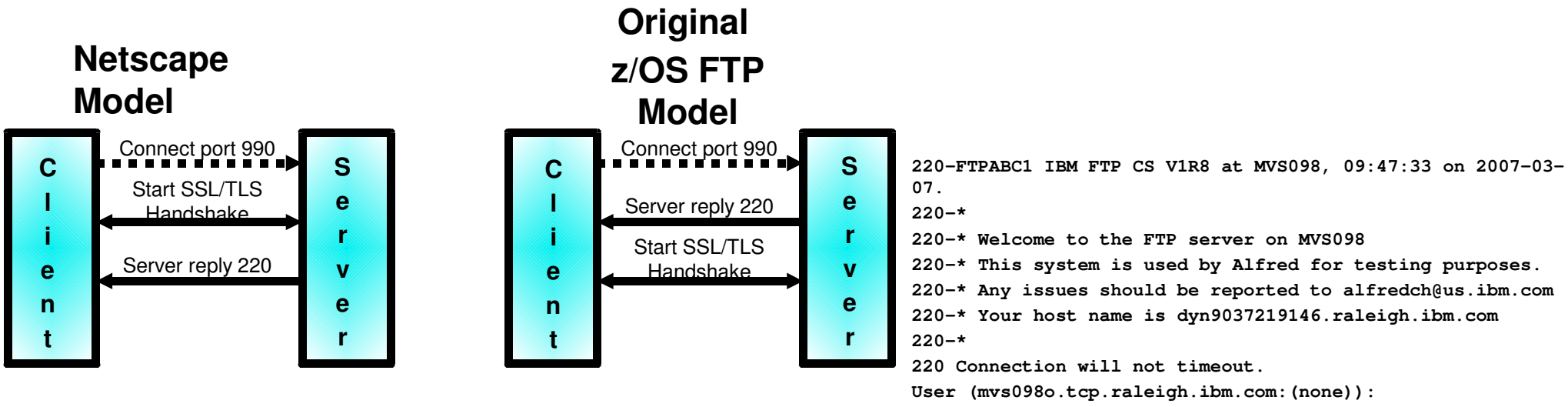
- ADAT
  - Contains optional security data as required by the security mechanism established with the AUTH command. Examples of security data to be sent via an ADAT command is a Kerberos ticket. Binary data is encoded in Base64 encoding. Server reply may include security data from server to client. The SSL/TLS support does not use the ADAT command.

- PBSZ
  - A numeric value to establish the size of the data buffer to be exchanged bwteen the client and the server. PBSZ is required, but SSL/TLS doesn't need it so for SSL/TLS you'll see a PBSZ 0 command,

- PROT
  - Requesting level of data connection protection:
    - C - Clear
    - S - Safe (authenticated, but not encrypted) - not supported by SSL/TLS, but is supported for Kerberos
    - P - Private (both encrypted and authenticated)

Implicit mode connection setup

- **Since implicit mode FTP wasn't ever defined precisely in the RFC standards, two different incompatible models were used:**
  - The Netscape model
  - The original z/OS FTP model



- **The z/OS FTP client was changed to optionally work according to the Netscape model via APAR PQ87711, which was PTFed back to z/OS V1R4 and rolled into z/OS V1R7:**
  - SECUREIMPLICITZOS=TRUE — The original z/OS FTP model (until V1R10 required for z/OS FTP server)
  - SECUREIMPLICITZOS=FALSE — The Netscape model (to non-z/OS FTP servers)

- **z/OS V1R10 CS adds similar support to the z/OS FTP server**
  - Reusing the SECUREIMPLICITZOS configuration option that is already used by the z/OS FTP client

Configuring TLS/SSL support for FTP

- **FTP supports two implementations of TLS/SSL support**
  - AT-TLS
  - FTP System SSL

- **FTP System SSL support was available in z/OS 1.2**
  - **FTP.DATA file used to configure FTP security parameters**
    - **Key ring and ciphers configured**
    - **Also contains general FTP configuration settings**
- **AT-TLS support is the preferred method**
  - **Support for new System SSL functions is implemented in AT-TLS only**
  - **AT-TLS requires:**
    - **Policy Agent needs to be running**
    - **AT-TLS policy configured to protect FTP traffic**
    - **TCPCONFIG TTLS coded in the TCPIP stack profile**
  - **Does not require any changes to key rings**
    - **Application identity used to access key ring, not TCPIP stack**
  - **FTP.DATA statements used to configure ATTLS as security mechanism**

z/OS FTP server security options - page 1 of 4

```
;EXTENSIONS        AUTH_GSSAPI       ; Enable Kerberos authentication
                                    ; Default is disabled.

 EXTENSIONS        AUTH_TLS          ; Enable TLS authentication
                                    ; Default is disabled.

 TLSMECHANISM      ATTLS             ; Server-specific or ATTLS
                                    ; ATTLS - use ATTLS
                                    ; FTP - server-specific (D)

 SECURE_FTP        ALLOWED           ; Authentication indicator
                                    ; ALLOWED         (D)
                                    ; REQUIRED

 SECURE_LOGIN      REQUIRED          ; Authorization level indicator
                                    ; for TLS
                                    ; NO_CLIENT_AUTH (D)
                                    ; REQUIRED
                                    ; VERIFY_USER

 SECURE_PASSWORD   REQUIRED          ; REQUIRED (D) - User must enter
                                    ;     password
                                    ; OPTIONAL - User does not have to
                                    ;     enter a password
                                    ; This setting has meaning only
                                    ; for TLS when implementing client
                                    ; certificate authentication
```

Switch between FTP's built-in SSL/TLS support and ATTLS support

Must all connections be secure or just those who wish to be?

Is client authentication required and if so, at what level?

If client authentication is used at level 3 and a user ID can be matched, is a password still required or not?

z/OS FTP server security options - page 2 of 4

```
;SECURE_PASSWORD_KERBEROS  REQUIRED  ; REQUIRED (D) - User must enter
                                     ;     password
                                     ; OPTIONAL - User does not have to
                                     ;     enter a password
                                     ; This setting has meaning only
                                     ; for Kerberos

 SECURE_CTRLCONN   CLEAR             ; Minimum level of security for
                                     ; the control connection
                                     ; CLEAR          (D)
                                     ; SAFE
                                     ; PRIVATE

 SECURE_DATACONN   CLEAR             ; Minimum level of security for
                                     ; the data connection
                                     ; NEVER
                                     ; CLEAR          (D)
                                     ; SAFE
                                     ; PRIVATE

;SECURE_PBSZ       16384             ; Kerberos maximum size of the
                                     ; encoded data blocks
                                     ; Default value is 16384
                                     ; Valid range is 512 through 32768
```

Server's requirement to security of the control connection. Must be set to CLEAR for the server to accept the CCC command

Server's requirement to security of the data connection

z/OS FTP server security options - page 3 of 4

```
; Name of a ciphersuite that can be passed to the partner during
; the TLS handshake. None, some, or all of the following may be
; specified. The number to the far right is the cipherspec id
; that corresponds to the ciphersuite's name.
;
; When using ATTLS, these are controlled via the ATTLS
; Policy
;
;CIPHERSUITE       SSL_NULL_MD5      ; 01
;CIPHERSUITE       SSL_NULL_SHA      ; 02
;CIPHERSUITE       SSL_RC4_MD5_EX    ; 03
;CIPHERSUITE       SSL_RC4_MD5       ; 04
;CIPHERSUITE       SSL_RC4_SHA       ; 05
;CIPHERSUITE       SSL_RC2_MD5_EX    ; 06
;CIPHERSUITE       SSL_3DES_SHA      ; 0A
 CIPHERSUITE       SSL_AES_128_SHA   ; 2F
;CIPHERSUITE       SSL_AES_256_SHA   ; 35
 CIPHERSUITE       SSL_DES_SHA       ; 09
```

Server's required ciphersuites

z/OS FTP server security options - page 4 of 4

```
; When using ATTLS, the keyring is controlled via the
; ATTLS policy
;
 KEYRING            TLSRING             ; Name of the keyring for TLS
                                        ; It can be the name of an hfs
                                        ; file (name starts with /) or
                                        ; a resource name in the security
                                        ; product (e.g., RACF)

;
; When using ATTLS, the TLS timeout value is controlled via the
; ATTLS policy
;
 TLSTIMEOUT         100                 ; Maximum time limit between full
                                        ; TLS handshakes to protect data
                                        ; connections
                                        ; Default value is 100 seconds.
                                        ; Valid range is 0 through 86400

 TLSRFCLEVEL        RFC4217             ; Specify what level of RFC 4217,
                                        ; On Securing FTP with TLS, is
                                        ; supported.
                                        ; DRAFT     (D) Internet Draft level
                                        ; RFC4217       RFC level
```

Server's keyring - prefixed with FTPD's started task userID: userID.TLSRING

Is z/OS FTP server to operate at the old draft RFC level for SSL/TLS or the now existing RFC? NB: default is to use draft - you may want to change that!!!!

z/OS FTP server AT-TLS policy - page 1 of 2

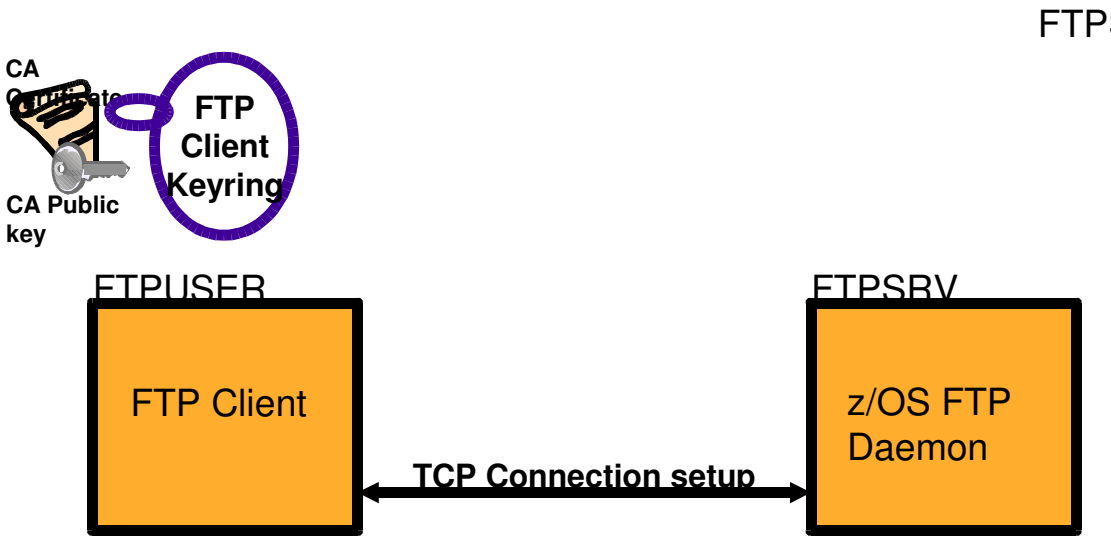
```
# This rule maps the FTP client traffic originating on z/OS
# Any connections outbound to port 21 will use this AT-TLS policy
# The rule identifies the AT-TLS configuration actions which control
# how AT-TLS will behave
TTLSRule        ftp_cli_21
{
 RemotePortRange 21
 Direction      Outbound
 TTLSGroupActionRef grp_act1
 TTLSEnvironmentActionRef env_act_cli2
}

# The TTLSGroupAction represents an instance of a C runtime environment
 TTLSGroupAction grp_act1
{
 TTLSEnabled On
}
```

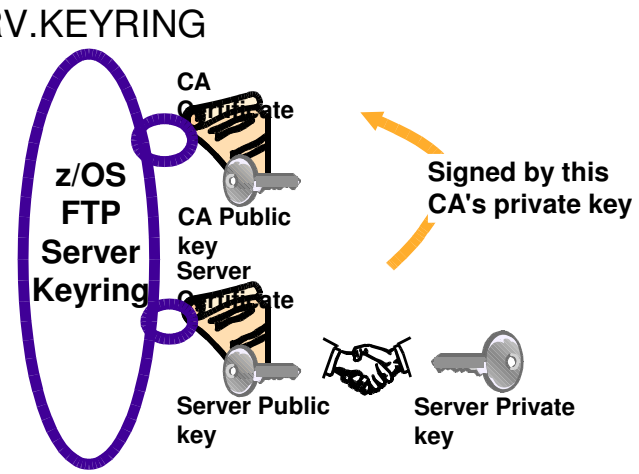
z/OS FTP server AT-TLS policy - page 2 of 2

```
# The TTLSEnvironmentAction configured how AT-TLS will use System SSL to
# protect connections.
TTLSEnvironmentAction env_act_cli2
{
   HandshakeRole Client
   TTLSKeyRingParms
   {
      Keyring User3_keyring  # RACF keyring to be accessed under user identity
   }
   TTLSEnvironmentAdvancedParms
   {
      TLSV1.1 On   # Configures which TLS/SSL protocols are supported
      TLSV1.2 Off
      CertificateLabel dh_rsa # Specifies a certificate to be used from keyring
      ApplicationControlled On # Required to allow FTP to start/stop security
      SecondaryMap On # Required to allow data connections to share SSL environment
                      # of the control connection
    }
# Configure the ciphers allowed to be used.  Either the name or the hex value
# can be used to identify the ciphers.  Ciphers should be coded with most preferred
# cipher first.
    TTLSCipherParms
    {
          V3CipherSuites TLS_DHE_DSS_WITH_AES_256_CBC_SHA
          V3CipherSuites 04
    }
}
```

A few words about certificates

Certificates needed for z/OS FTP server authentication only





Hello - I want to use SSL/TLS

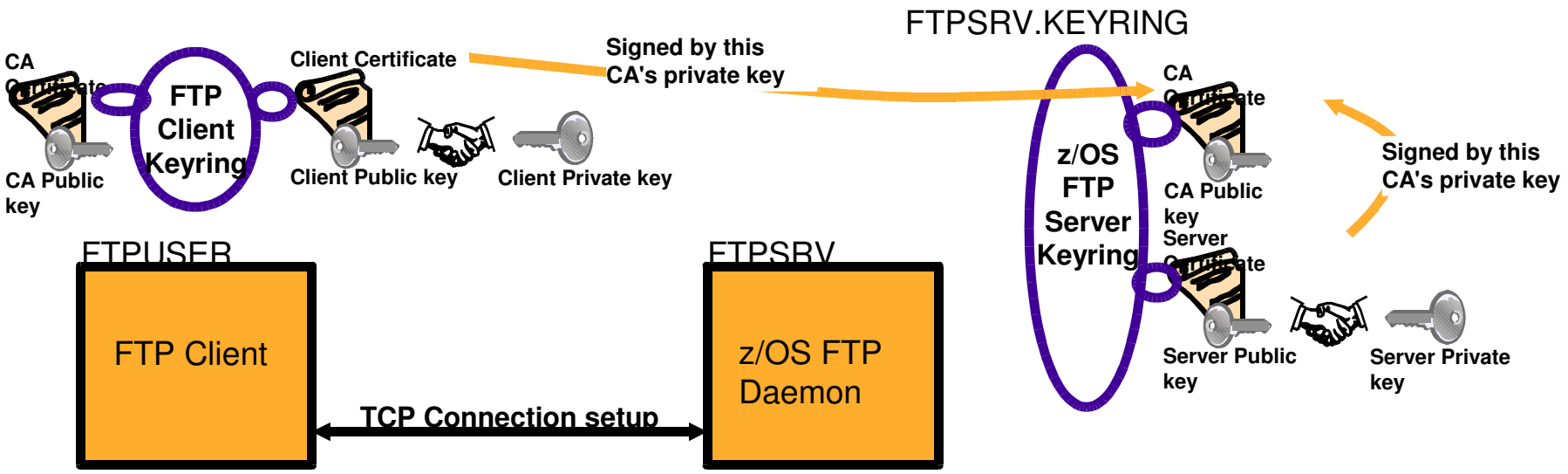
Hello - OK, me too
And here is my server certificate

1. Verify server certificate has not expired
2. Verify server certificate is valid using CA's public key
3. Do optional checks on the server certificate
4. Store server's public key for later use
5. Generate symmetric key and encrypt under server's public key

Here is our secret symmetric key encrypted under your public key

- **CA may be an external CA, such as Verisign, or it may be an in-house CA**
  - In both cases, the CA root certificate needs to be present at both the client and the server side
- **The server certificate is signed by the CA and is stored on the server side**
  - On z/OS, this will typically be the default certificate in the FTP server's started task userID's keyring in RACF
- **During SSL handshake, the server certificate (not the server private key) is sent to the client**
  - The client verifies the certificates signature using the CA public key in its copy of the CA certificate

Certificates needed for z/OS FTP server and FTP client authentication



Hello - I want to use SSL/TLS

Hello - OK, me too

Here is my server certificate, and I want to see your client certificate!

Server Certificate

Server Public key

1. Verify server certificate has not expired
2. Verify server certificate is valid using CA's public key
3. Do optional checks on the server certificate
4. Store server's public key for later use
5. Generate symmetric key and encrypt under server's public key

Here is our secret symmetric key encrypted under your public key

- and here is my client certificate

Client Certificate

Client Public key

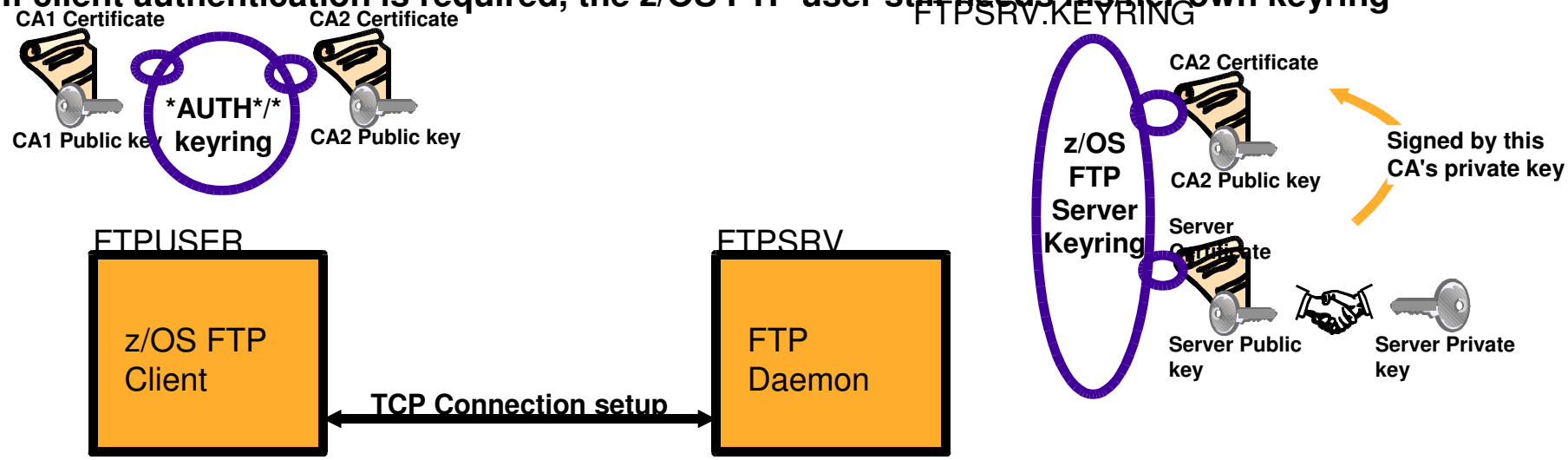
1. Verify client certificate has not expired
2. Verify client certificate is valid using CA's public key
3. Do optional checks on the client certificate
   1. Does it map to a RACF user ID (authentication level 2)
   2. Is the user permitted to use this service (authenitcation level 3)

FTP server client authentication levels

Authentication level	FTP server SECURE_LOGIN option	Description
Level 1	REQUIRED	The authenticity and validity of the client certificate is verified against the trusted roots in the FTP server's keyring.
Level 2	VERIFY_USER	Same as level 1 PLUS a verification that the client certificate is registered by RACF and mapped to a known RACF user ID.
Level 3	VERIFY_USER	Same as level 2 PLUS a verification that the user ID has permission to a SERVAUTH profile that represents this specific FTP server: EZB.FTP.sysname.ftpdaemonname.PORTnnnnn

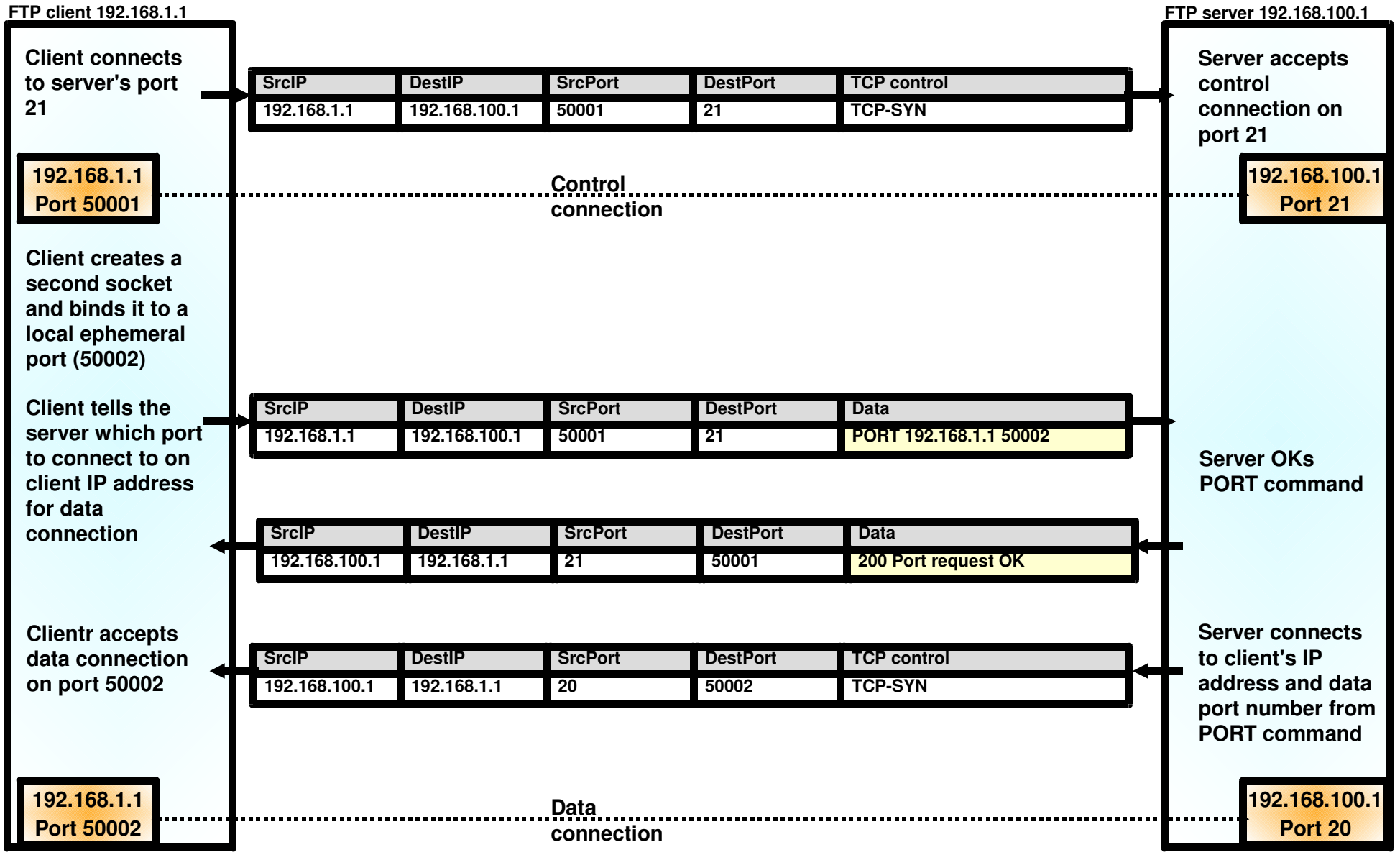
Virtual keyrings are very useful when z/OS is the FTP client

- **If z/OS is the FTP client, does every FTP user on z/OS have to have a key ring with a copy of the CA certificate?**
  - A few releases back, the answer was yes
    - –What we call an "administratively heavy process"
  - z/OS V1R8 added support for something known as a virtual keyring
- **To have System SSL check all CERTAUTH certificates in RACF when verifying a server certificate that was received during the SSL handshake, specify a key ring in the client FTP.DATA (or matching AT-TLS definitions) as:**
  - `KEYRING *AUTH*/*`
- **If client authentication is required, the z/OS FTP user still needs his/her own keyring**

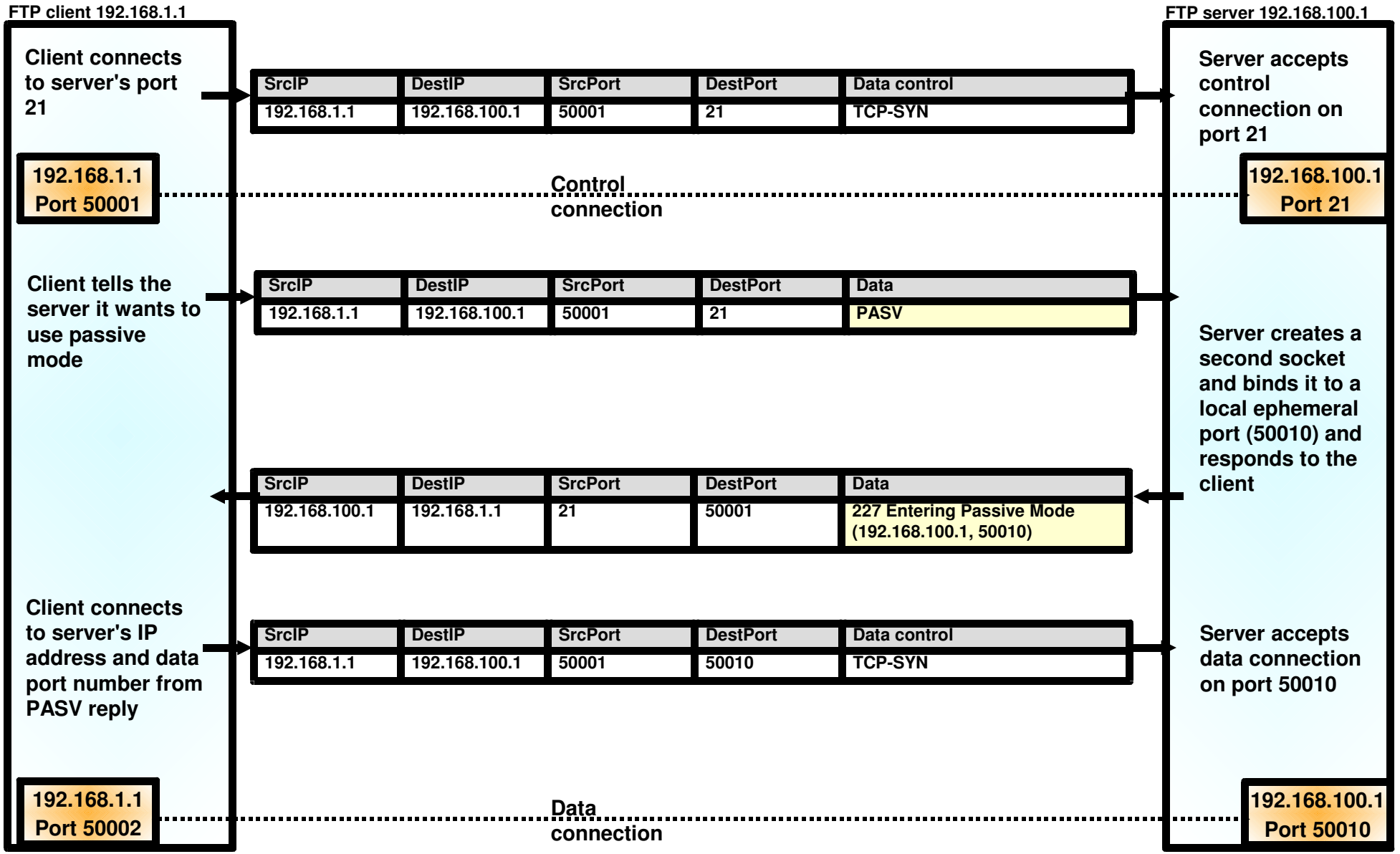


Secure FTP and network traversal challenges and solutions

Active mode data connection setup - revisited



Passive mode data connection setup - revisited



So what's the problem?

I am a firewall who wants to inspect the FTP control connection data !

No encryption:

SrcIP	DestIP	SrcPort	DestPort	Data
192.168.100.1	192.168.1.1	21	50001	227 Entering Passive Mode (192.168.100.1, 50010)

SSL/TLS encryption:

SrcIP	DestIP	SrcPort	DestPort	Data
192.168.100.1	192.168.1.1	21	50001	@%$#*&&^^!:"J)*GVM><

IPSec encryption:

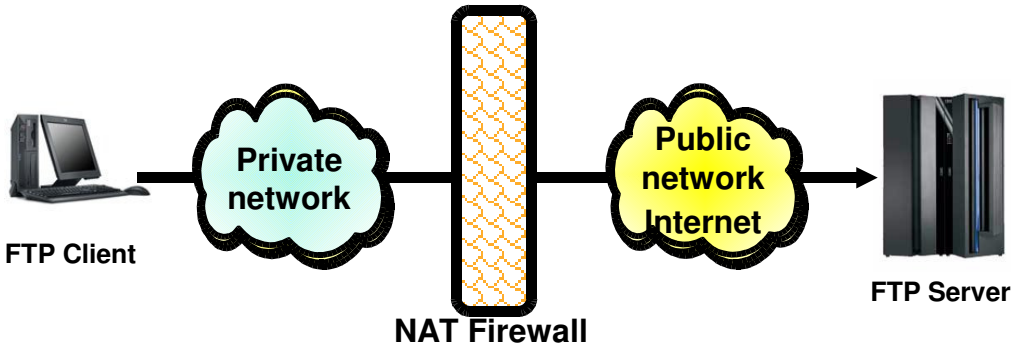
SrcIP	DestIP	SrcPort	DestPort	Data
192.168.100.1	192.168.1.1	>::"	*&hU$$$$	@%$#dd*&&^s^!:"J) *bGVM>(*hhgvvv<

IP header encryption varies based on transport/tunnel mode, and AH/ESP protocol

- **In an internal/secure network without firewalls and network address translation devices - there are no problems!**
  - Both secure and non-secure FTP transfers will work just fine
- **If the FTP control connection is insecure (not encrypted or authenticated), firewalls and NAT devices manage by "peeking" into the control connection data flows - and for NAT also by modifying the control connection data flows**
  - Firewalls enable dynamic IP filters based on the IP addresses in the PORT command or the PASV reply
  - NAT devices modify the IP addresses in the PORT command and the PASV reply - in addition to their normal NAT processing of the IP headers
- **If the FTP control connection is secure (encrypted and/or authenticated), the FTP data connection setup will generally fail if the control connection passes through firewalls and/or NAT devices**
  - Firewalls are not able to determine what the IP addresses in the PORT command and PASV reply are and cannot dynamically enable filter rules
  - NAT devices can't find the IP addresses they need to change and even if they did find them and changed them, the message authentication checking will fail the data when it arrives at its final destination

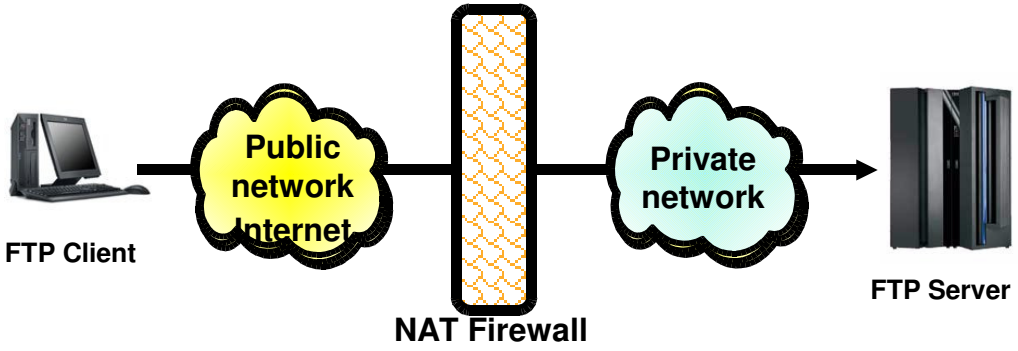
Why does secure FTP sometimes work through a NAT firewall?

- **Passive mode (PASV) may work**
- **Active mode (PORT) will *not* work**



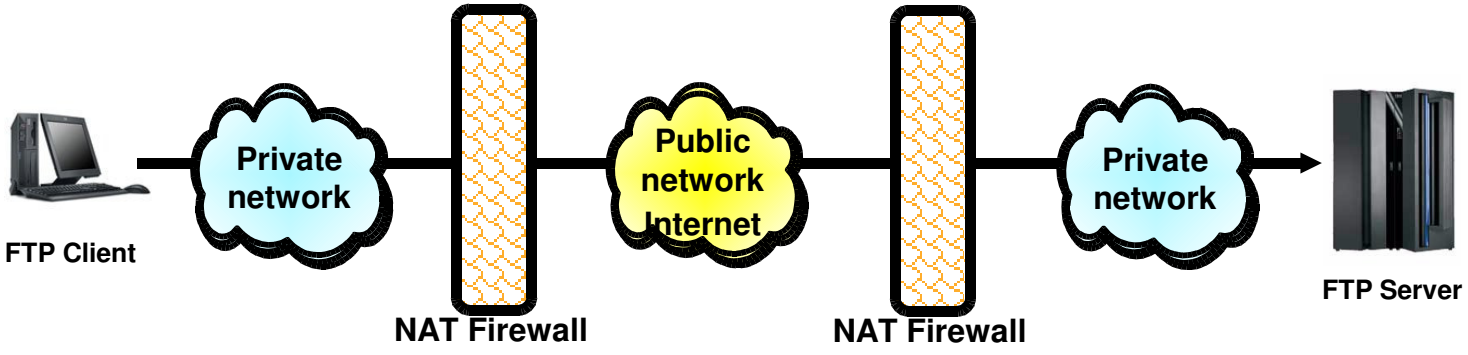
PASV reply will return a public IP address!
(This is the configuration where passive mode got the name Firewall-Friendly from)

- **Passive mode (PASV) will *not* work**
- **Active mode (PORT) may work**



PORT command will include a public IP address!

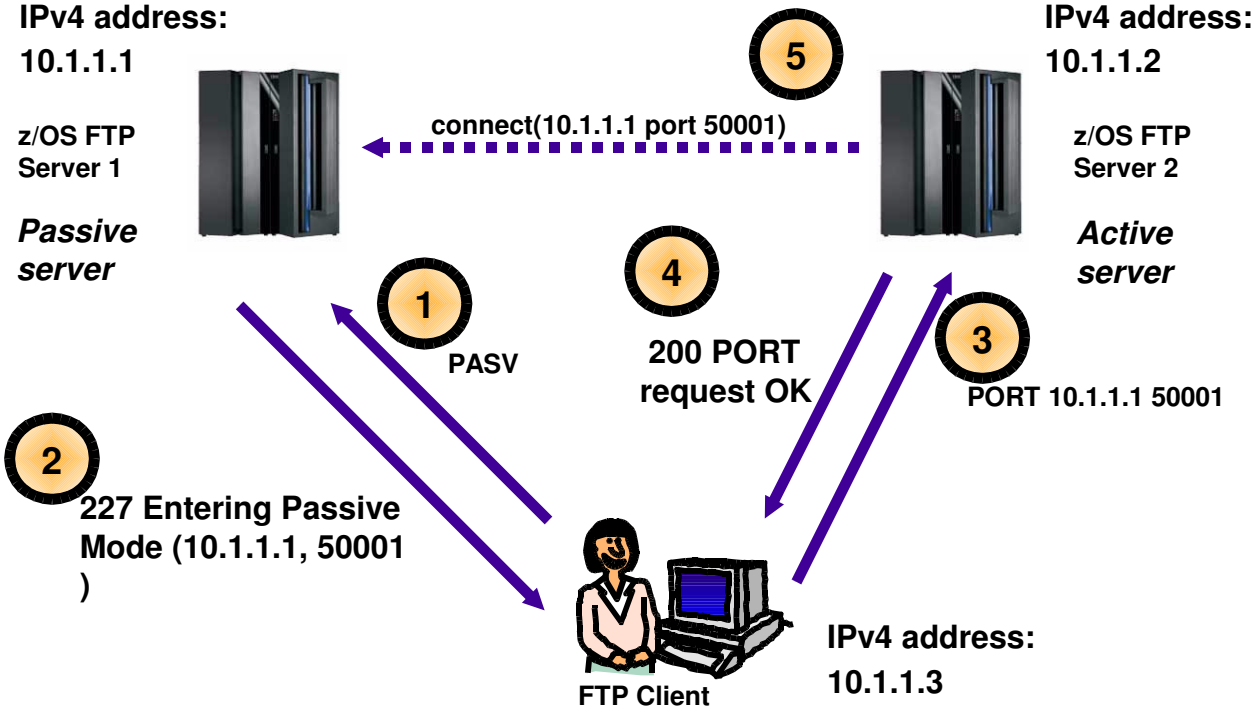
- **Neither passive mode (PASV) nor active mode (PORT) will work!**



What are those IP addresses there for in the first place?

- **Why do we need an IP address in the PASV reply or on the PORT command?**
  - In 99.999% of FTP operations, we don't need it!
  - We only need it if we're engaging into so-called three-way FTP proxy operations
    - Once considered a very important capability of the founding FTP fathers

- **Most z/OS installations do not like three-way proxy operation of their FTP servers**
  - It is generally considered an unnecessary security risk
  - The z/OS FTP server provides configuration options to disable it from being used in a three-way proxy operation
    - PASSIVEDATACONN
    - PORTCOMMANDIPADDR



PASSIVEDATACONN says what this server is to do when it receives a data connection setup request from a source IP address that isn't the same as the FTP client IP address

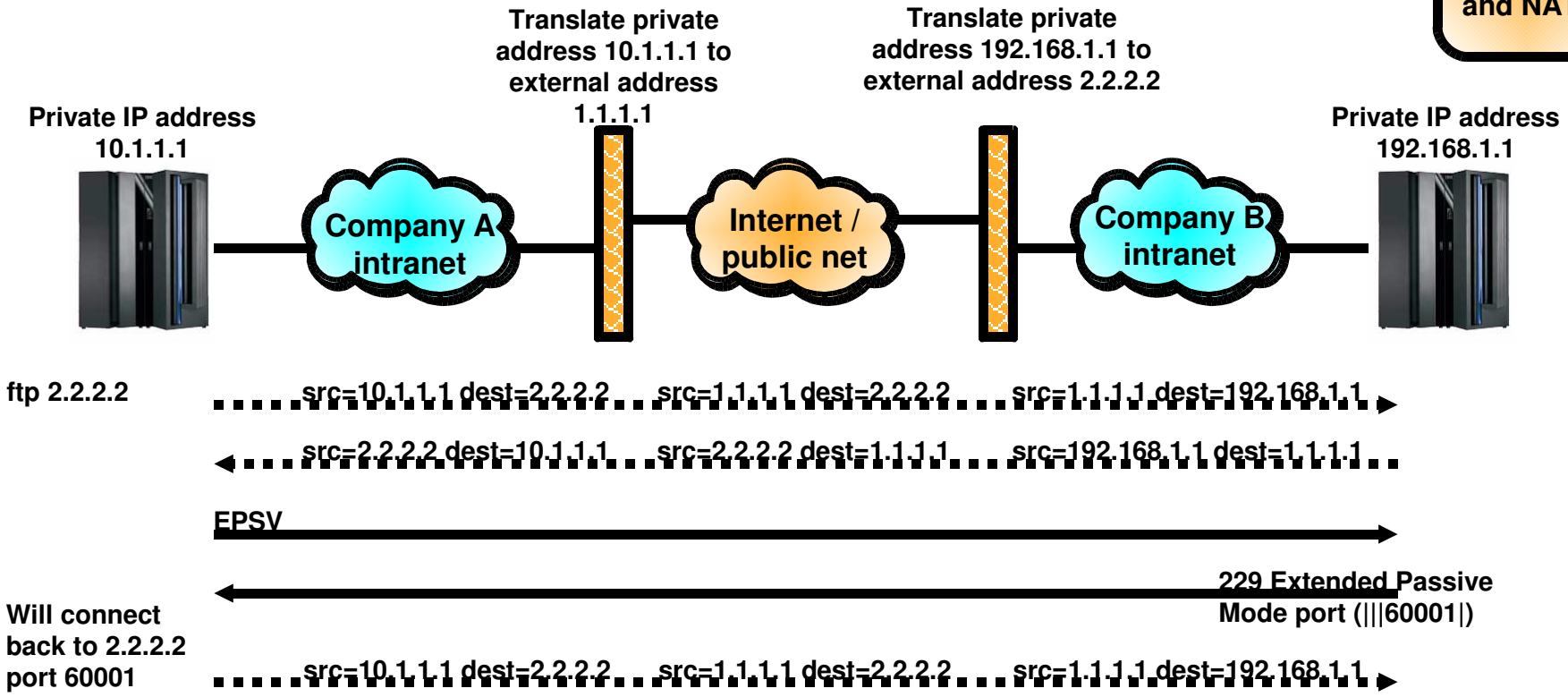
PORTCOMMANDIPADDR says what this server is to do when it receives a PORT command with an IP address that isn't the same as the FTP client IP address.

Extended passive mode gets rid of the IP address

- **In almost all FTP operations, the control connection and the data connection are between the same two IP addresses**

- **The extended passive mode FTP operation takes advantage of this**
  - The EPSV reply does not include an IP address, but only a port number
    - The FTP client will connect to the same IP address it used for the control connection
  - The EPSV and the accompanying extended port commend (EPRT) are also used to enable IPv6 support in FTP
    - EPSV and EPRT can be used with both IPv4 and IPv6
    - Used with IPv4, the EPSV command provides NAT firewall relief

RFC2428 FTP Extensions for IPv6 and NATs



Does EPSV solve all NAT firewall problems for FTP?

- **Not all FTP clients and servers support extended passive mode**
  - Many vendors mis-interpreted RFC 2428 and thought it was only needed if they wanted to implement IPv6 support for FTP

- **EPSV doesn't do anything for the port number**
  - If firewalls also implement static port-based filters, how would they know which port numbers to permit for FTP data connections?
    - Firewall administrators do generally not like adding a permit rule that allows connections to an IP address and any port number !

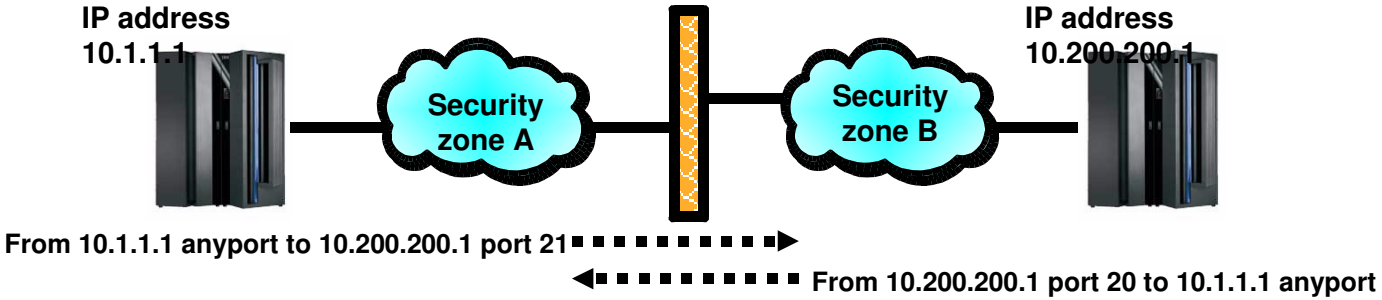
Connections to any port number except 21

  - If firewalls implement dynamic port filters, how would they know which port number is about to be used for a data connection?
    - The firewall cannot learn what the port number is - it is encrypted

Connections to any port number except 21 or a port seen in a recent PASV reply

How to deal with static filters in a firewall

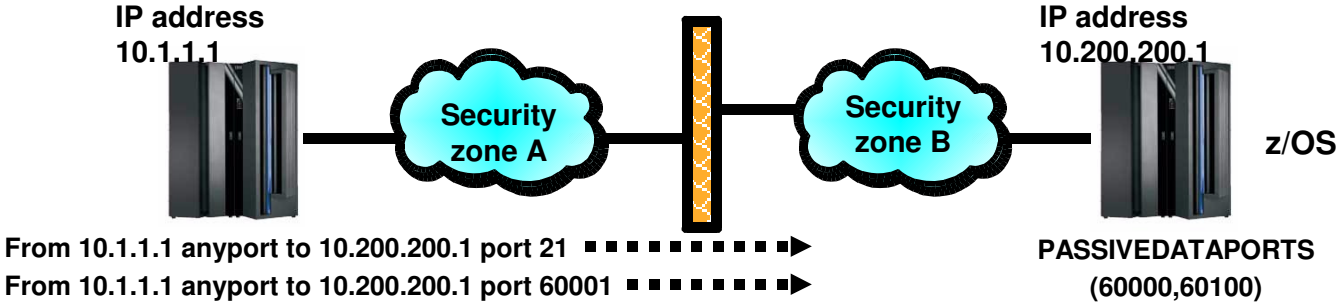
- **If you are able to use active mode FTP, the firewall filters can sometimes be managed:**
  - The control connection is permitted inbound to port 21
  - The data connection is permitted outbound from port 20
  - Will work for both standard active mode (PORT) and extended active mode (EPRT)



Static firewall filters

- Connection setup from 10.1.1.1 any port to port 21 on 10.200.200.1 - permit
- Connection setup from 10.200.200.1 port 20 to 10.1.1.1 any port - permit

- **If you use passive mode FTP, and your server is a z/OS FTP server, you can predefine a range of port numbers to be used for passive mode data connections**
  - The control connection is permitted inbound to port 21
  - The data connection is permitted inbound to a port in a pre-defined range
  - Will work for both standard passive mode (PASV) and extended passive mode (EPSV)

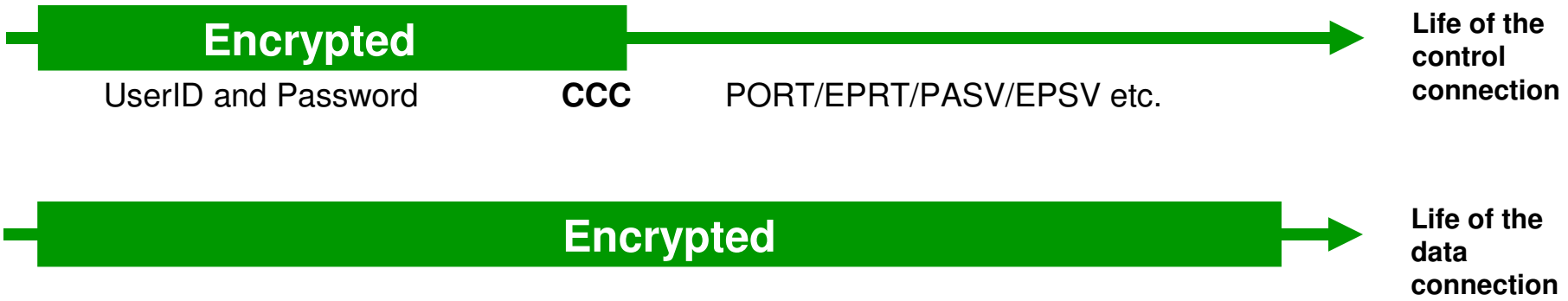


Static firewall filters

- Connection setup from 10.1.1.1 any port to port 21 on 10.200.200.1 - permit
- Connection setup from 10.1.1.1 any port to a port in the range from 60000 to 60100 on 10.200.200.1 - permit

How to deal with dynamic filters in a firewall

- **When using dynamic filters, the firewall enables (permits) ports based on IP address and/or port number information in the PORT/EPRT command or the PASV/EPSV reply**
  - The original FTP SSL/TLS draft RFC stated that the FTP control connection always had to be encrypted !
  - The final RFC (RFC 4217 "Securing FTP with TLS") relaxes on this requirement and implements a new Clear Command Channel (CCC) FTP command



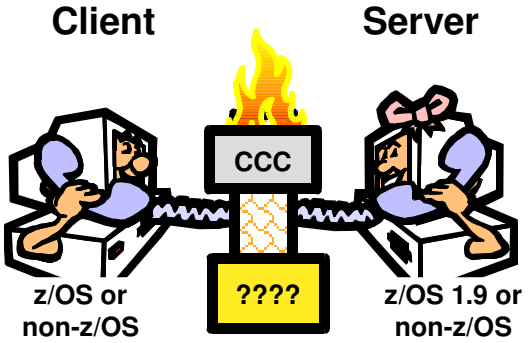
- **Both the FTP client and server need to support the CCC command according to RFC 4217**
  - Not all FTP clients and servers that support FTP SSL/TLS support the CCC command
    - z/OS added full support for the CCC command in z/OS V1R9 (both z/OS FTP client and server)
      - APAR PK26746 supplied this function for the z/OS FTP client in fall 2006 (back to z/OS V1R4)
  - For those products that claim support, some interoperability issues have been observed !
    - z/OS FTP client works with z/OS FTP server (big surprise !!)
    - CoreFTP client for Windows works fine with z/OS FTP server
    - I have personally had problems with WS_FTP Pro's implementation of CCC
  - So test with your non-z/OS FTP clients carefully before proceeding

- **In general, the CCC command is a solution that solves SSL/TLS-enabled FTP issues with both NAT firewalls and filtering firewalls**

A few simple steps to make secure FTP work through firewalls

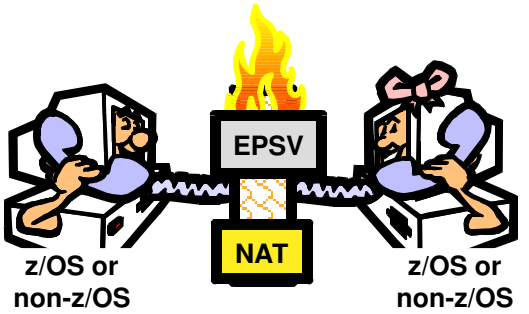
1 **If you don't know what your firewalls do and your z/OS system is at a V1R9 level and your partner secure FTP product supports CCC (and is compatible with RFC 4217)**

- Use the CCC command method - with z/OS as the client or as the server



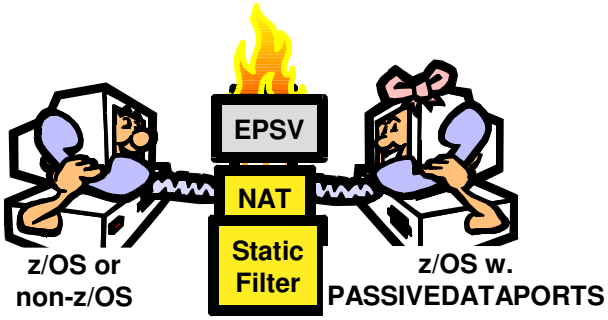
2 **If your firewalls only do NAT (no filtering) and your partner secure FTP product supports extended passive mode (EPSV)**

- Use extended passive mode - with z/OS as the client or as the server



3 **If your firewalls do NAT and static filtering, you use z/OS as the FTP server, and your partner secure FTP client supports extended passive mode (EPSV)**

- Use the PASSIVEDATAPORTS option on your z/OS FTP server
- Have your firewall administrator add static filters that allow connections to the range of port numbers in PASSIVEDATAPORTS
- Use extended passive mode from your FTP client - may be z/OS or other FTP products



4 **If your firewalls do dynamic filtering (with or without NAT)**

- You must get to z/OS V1R9 and use the CCC command method - it is the only method that will work