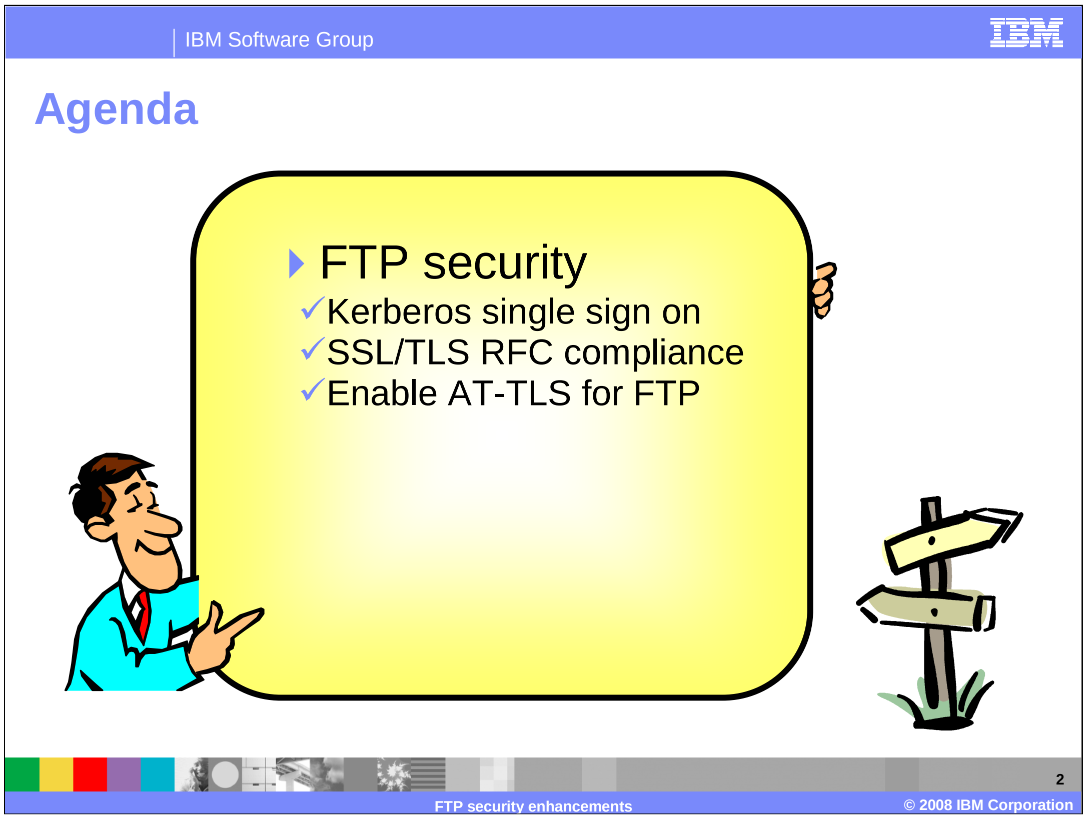
IBM Software Group

z/OS® V1R9 Communications Server

FTP security enhancements

Updated January 18, 2008

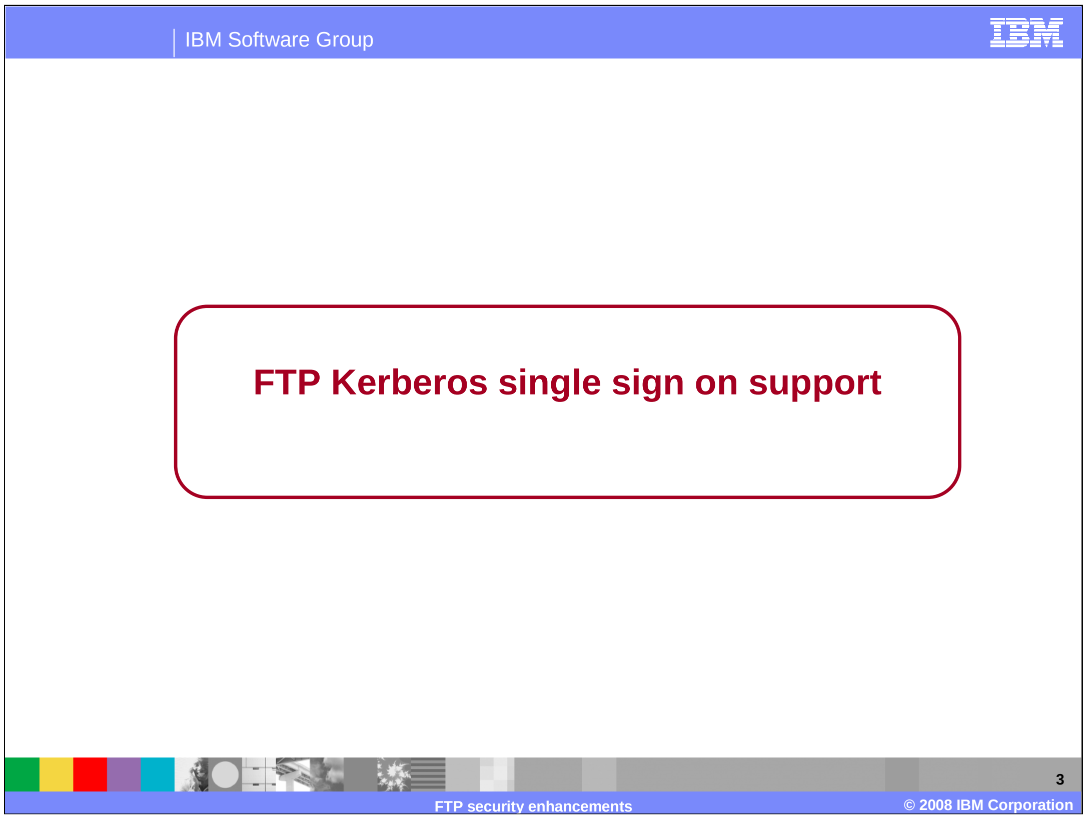
This presentation discusses the FTP security enhancements in Communications Server for the z/OS V1R9 Communications Server.



Agenda

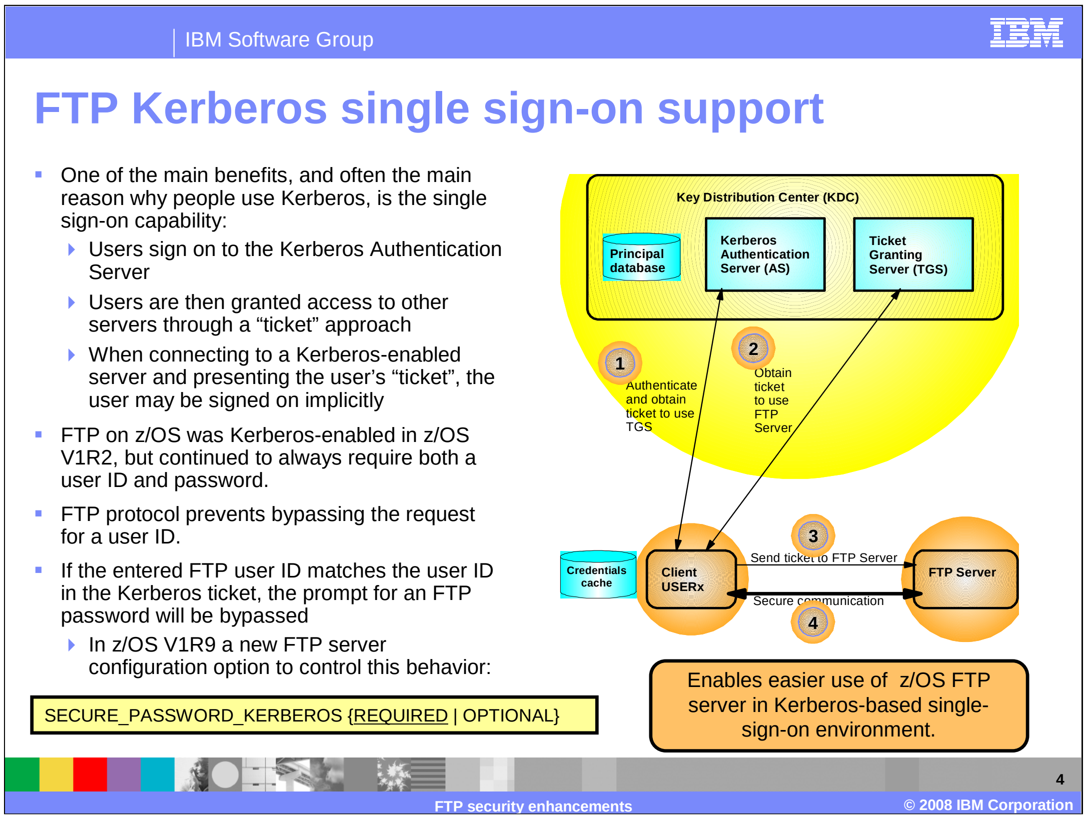
- FTP security
  - ✓Kerberos single sign on
  - ✓SSL/TLS RFC compliance
  - ✓Enable AT-TLS for FTP

This presentation explains security enhancements for FTP. These include, FTP Kerberos single sign on support, enhancements to FTP to comply with RFCs regarding TLS security, and the enablement of FTP to use AT-TLS.



FTP Kerberos single sign on support

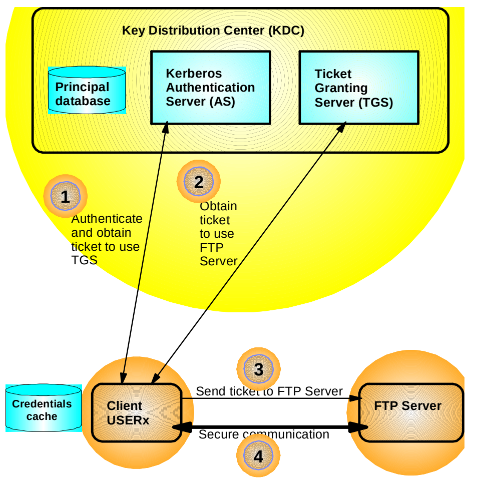
This section covers an extension to the Kerberos support which enables connections to the z/OS FTP server using Kerberos without requiring the FTP client to supply the user's password.



FTP Kerberos single sign-on support

- One of the main benefits, and often the main reason why people use Kerberos, is the single sign-on capability:
  - Users sign on to the Kerberos Authentication Server
  - Users are then granted access to other servers through a “ticket” approach
  - When connecting to a Kerberos-enabled server and presenting the user’s “ticket”, the user may be signed on implicitly
- FTP on z/OS was Kerberos-enabled in z/OS V1R2, but continued to always require both a user ID and password.
- FTP protocol prevents bypassing the request for a user ID.
- If the entered FTP user ID matches the user ID in the Kerberos ticket, the prompt for an FTP password will be bypassed
  - In z/OS V1R9 a new FTP server configuration option to control this behavior:

```
SECURE_PASSWORD_KERBEROS {REQUIRED | OPTIONAL}
```

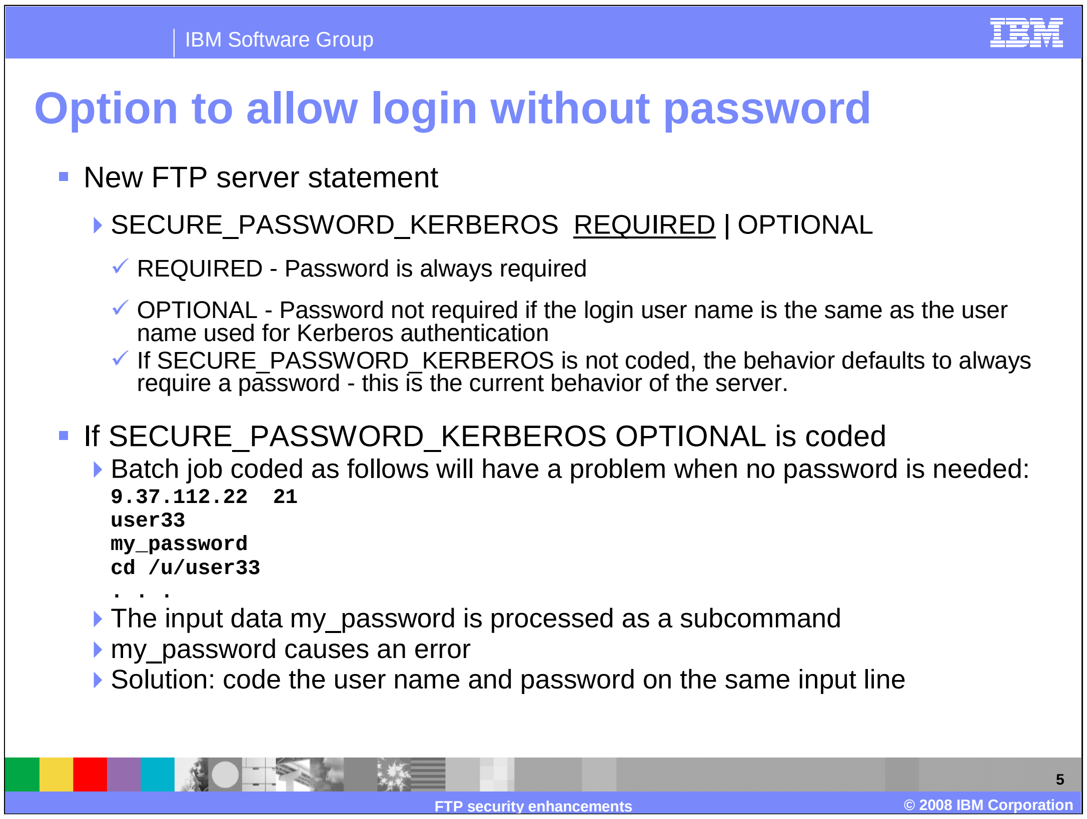

Key Distribution Center (KDC)
Principal database
Kerberos Authentication Server (AS)
Ticket Granting Server (TGS)
1
Authenticate and obtain ticket to use TGS
2
Obtain ticket to use FTP Server
3
Send ticket to FTP Server
Credentials cache
Client USERx
FTP Server
Secure communication
4

Enables easier use of z/OS FTP server in Kerberos-based single-sign-on environment.

In a Kerberos environment, users must authenticate to the Kerberos Key Distribution Center (KDC) by supplying their user name and password. Users are also accustomed to using single sign-on support. The user authenticates once to the Kerberos KDC and then should be able to access and be authenticated by other services without having to enter their password again. However, if they then login to a Kerberos enabled z/OS FTP server, they must enter their user name and password again.

The solution to the problem is to allow users to login to the z/OS FTP server without having to re-enter the password. First, the user must authenticate to the Kerberos KDC. Then the user starts the FTP client and connects to the z/OS FTP server using GSSAPI authentication. GSSAPI, or Generic Security Service Application Programming Interface, is the authentication method used by the FTP protocol to allow connections between Kerberos enabled clients and servers.

The FTP protocol still requires that the client supply a user name to the FTP server. If the user name supplied to the z/OS FTP server is the same user name used to authenticate to the Kerberos KDC, the z/OS FTP server will not prompt for the password.



Option to allow login without password

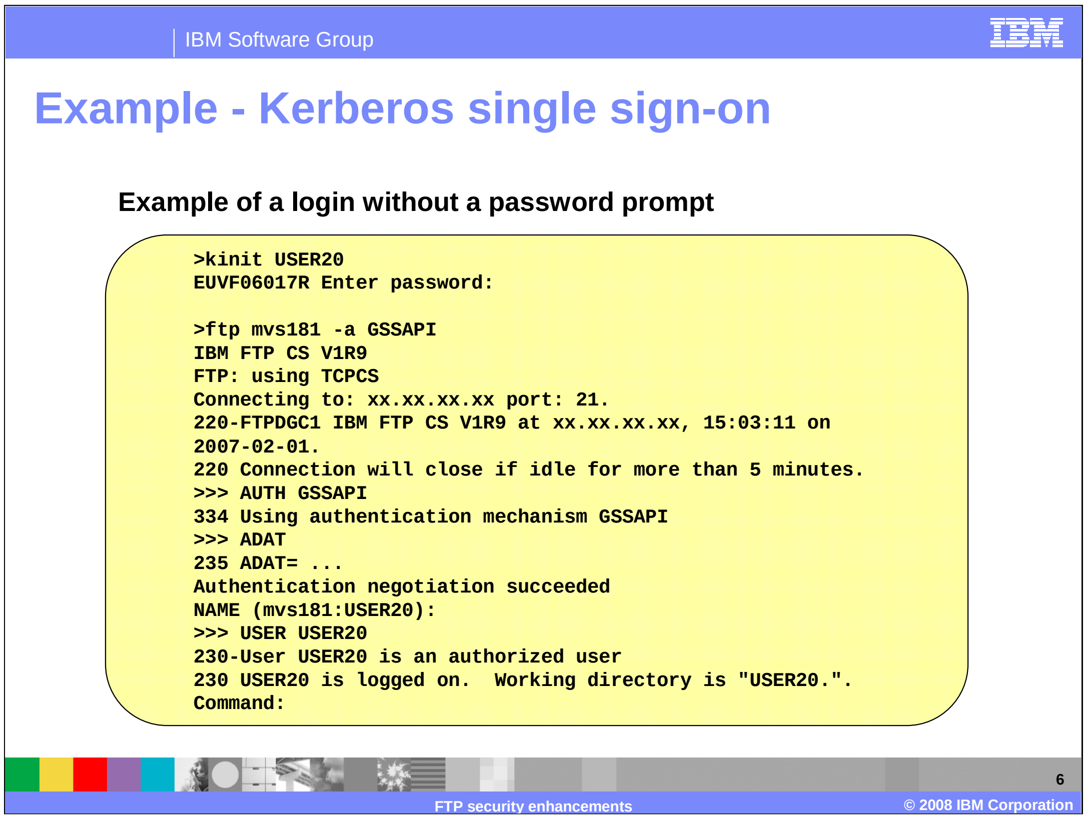
- New FTP server statement
  - SECURE_PASSWORD_KERBEROS <u>REQUIRED</u> | OPTIONAL
    - ✓ REQUIRED - Password is always required
    - ✓ OPTIONAL - Password not required if the login user name is the same as the user name used for Kerberos authentication
    - ✓ If SECURE_PASSWORD_KERBEROS is not coded, the behavior defaults to always require a password - this is the current behavior of the server.
- If SECURE_PASSWORD_KERBEROS OPTIONAL is coded
  - Batch job coded as follows will have a problem when no password is needed:

```
9.37.112.22  21
user33
my_password
cd /u/user33
. . .
```

  - The input data my_password is processed as a subcommand
  - my_password causes an error
  - Solution: code the user name and password on the same input line

A new statement, SECURE_PASSWORD_KERBEROS, has been added to the FTP.DATA file. It has two values, REQUIRED and OPTIONAL. If REQUIRED is coded, the z/OS FTP server will always prompt for the user's password. If OPTIONAL is coded and the user name used to log into the z/OS FTP server is the same as the user name used to authenticate to the Kerberos KDC, the z/OS FTP server will not require the user's password and the user will be logged in. If OPTIONAL is coded and the user name used to log into the z/OS FTP server is not the same as the user name used to authenticate to the Kerberos KDC, the z/OS FTP server will require the user's password before the user can be logged in. The default value is REQUIRED.

Batch jobs may have to be updated if this function is enabled. If the batch job specifies the user name and the password on separate lines and SECURE_PASSWORD_KERBEROS OPTIONAL is coded, the batch job may incorrectly supply the password when the server does not prompt for the password. The server will reject the password since the user will already be logged in. To avoid this problem, the batch job can be changed to code the user name and password on the same line.



Example - Kerberos single sign-on

Example of a login without a password prompt

```
>kinit USER20
EUVF06017R Enter password:

>ftp mvs181 -a GSSAPI
IBM FTP CS V1R9
FTP: using TCPCS
Connecting to: xx.xx.xx.xx port: 21.
220-FTPDGC1 IBM FTP CS V1R9 at xx.xx.xx.xx, 15:03:11 on
2007-02-01.
220 Connection will close if idle for more than 5 minutes.
>>> AUTH GSSAPI
334 Using authentication mechanism GSSAPI
>>> ADAT
235 ADAT= ...
Authentication negotiation succeeded
NAME (mvs181:USER20):
>>> USER USER20
230-User USER20 is an authorized user
230 USER20 is logged on.  Working directory is "USER20.".
Command:
```

This is an example of an FTP client connecting to a z/OS FTP server which has enabled single sign on support by specifying SECURE_PASSWORD_KERBEROS OPTIONAL in the server’s FTP.DATA file.

First, the user must authenticate to the Kerberos KDC. This is done by using the kinit application. In this case the user authenticates to Kerberos as USER20.

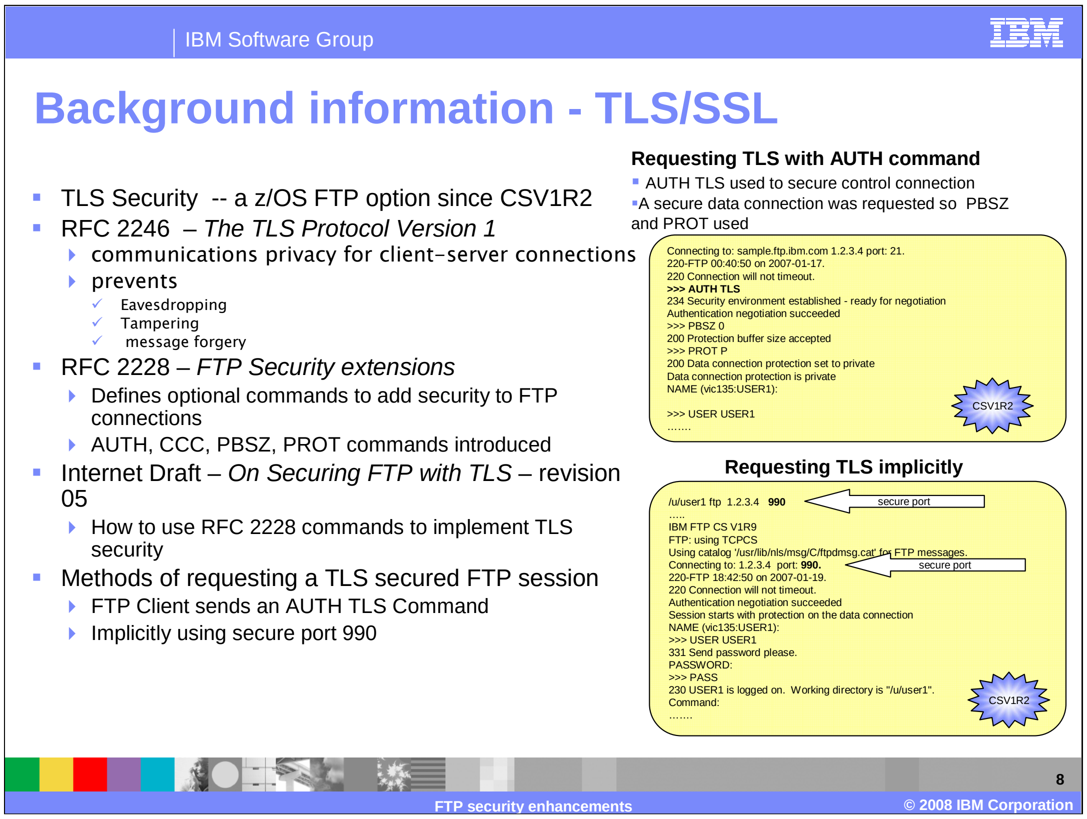
Next the client connects to the server requesting GSSAPI authentication.

When the GSSAPI authentication is successful, the prompt for the user name is issued. The client supplies the user name, which again in this case is USER20.

Since the user name is the same as the user name which was previously authenticated by Kerberos on the kinit, the FTP server does not require the password, and the user is successfully logged in.

FTP TLS/SSL compliance

This section describes enhancements to z/OS FTP to comply with RFCs regarding TLS security.


IBM Software Group

Background information - TLS/SSL

- TLS Security -- a z/OS FTP option since CSV1R2
- RFC 2246 – *The TLS Protocol Version 1*
  - communications privacy for client-server connections
  - prevents
    - ✓ Eavesdropping
    - ✓ Tampering
    - ✓ message forgery
- RFC 2228 – *FTP Security extensions*
  - Defines optional commands to add security to FTP connections
  - AUTH, CCC, PBSZ, PROT commands introduced
- Internet Draft – *On Securing FTP with TLS* – revision 05
  - How to use RFC 2228 commands to implement TLS security
- Methods of requesting a TLS secured FTP session
  - FTP Client sends an AUTH TLS Command
  - Implicitly using secure port 990

Requesting TLS with AUTH command

- AUTH TLS used to secure control connection
- A secure data connection was requested so PBSZ and PROT used

```
Connecting to: sample.ftp.ibm.com 1.2.3.4 port: 21.
220-FTP 00:40:50 on 2007-01-17.
220 Connection will not timeout.
>>> AUTH TLS
234 Security environment established - ready for negotiation
Authentication negotiation succeeded
>>> PBSZ 0
200 Protection buffer size accepted
>>> PROT P
200 Data connection protection set to private
Data connection protection is private
NAME (vic135:USER1):

>>> USER USER1
…….
```

Requesting TLS implicitly

```
/u/user1 ftp  1.2.3.4   990
…..
IBM FTP CS V1R9
FTP: using TCPCS
Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages.
Connecting to: 1.2.3.4  port: 990.
220-FTP 18:42:50 on 2007-01-19.
220 Connection will not timeout.
Authentication negotiation succeeded
Session starts with protection on the data connection
NAME (vic135:USER1):
>>> USER USER1
331 Send password please.
PASSWORD:
>>> PASS
230 USER1 is logged on.  Working directory is "/u/user1".
Command:
…….
```

8

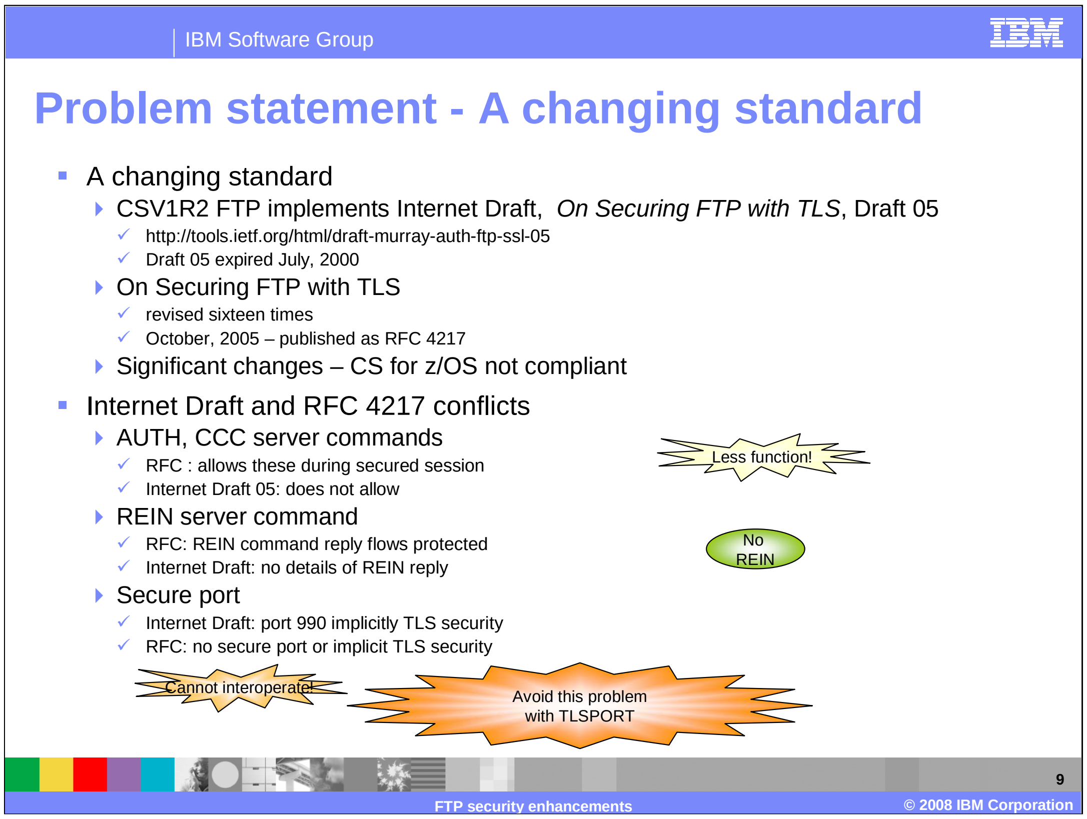
FTP security enhancements © 2008 IBM Corporation

In CSV1R2, z/OS FTP implemented TLS security based on the RFCs and Internet Draft listed. RFC 2246 defines TLS – Transport Layer Security. This is from the abstract: The TLS protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

File Transfer Protocols as defined in RFC 959 do not provide a protocol for requesting a secure session. RFC 2228 defines FTP commands to request a secure session, albeit not necessarily a TLS secure session. This presentation is concerned with the AUTH, CCC, PBSZ and PROT commands which are defined in that RFC. The Internet Draft, *On Securing FTP with TLS*, defines how the commands introduced in RFC 2228 should be used to request and maintain a TLS secured FTP session. The z/OS FTP TLS support is based on draft 05 of *On Securing FTP with TLS*.

One method of requesting a TLS secured FTP session is for the FTP client to send an AUTH TLS command to the server as illustrated in the first example. This establishes TLS security for the control connection. Securing the data connection is optional. In this case, the client was configured for a secure data connection, so the optional commands PBSZ and PROT were used to set up security for the data connection. The notes which follow this page describe the statements in the client's FTP.DATA that caused the client to request an FTP session and describe the related statements in the server's FTP.DATA.

The second example demonstrates how FTP establishes an **implicitly secure** connection. The Internet Draft, *On Securing FTP with TLS*, specified that connections to port 990 are assumed to be secure – no AUTH command is needed to secure the connection. This example, asks the ftp client to connect to port 990, the secure port. The same FTP.DATA sets are used for client and server as used in the first example. In those FTP.DATA sets, TLSPORT was set to the default value, 990. The first line shows the ftp client being started with the host name parameter of 1.2.3.4, and the optional port parameter of 990. Port 990 appears in **boldface** font. Contrasting this example with the previous example, notice the session is secure without the use of the AUTH, PBSZ, and PROT commands. The session is implicitly secured as opposed to explicitly securing the session with commands.


IBM Software Group

Problem statement - A changing standard

- A changing standard
  - CSV1R2 FTP implements Internet Draft, *On Securing FTP with TLS*, Draft 05
    - ✓ http://tools.ietf.org/html/draft-murray-auth-ftp-ssl-05
    - ✓ Draft 05 expired July, 2000
  - On Securing FTP with TLS
    - ✓ revised sixteen times
    - ✓ October, 2005 – published as RFC 4217
  - Significant changes – CS for z/OS not compliant
- Internet Draft and RFC 4217 conflicts
  - AUTH, CCC server commands
    - ✓ RFC : allows these during secured session
    - ✓ Internet Draft 05: does not allow
  - REIN server command
    - ✓ RFC: REIN command reply flows protected
    - ✓ Internet Draft: no details of REIN reply
  - Secure port
    - ✓ Internet Draft: port 990 implicitly TLS security
    - ✓ RFC: no secure port or implicit TLS security

Less function!

No REIN

Cannot interoperate!

Avoid this problem with TLSPORT

9

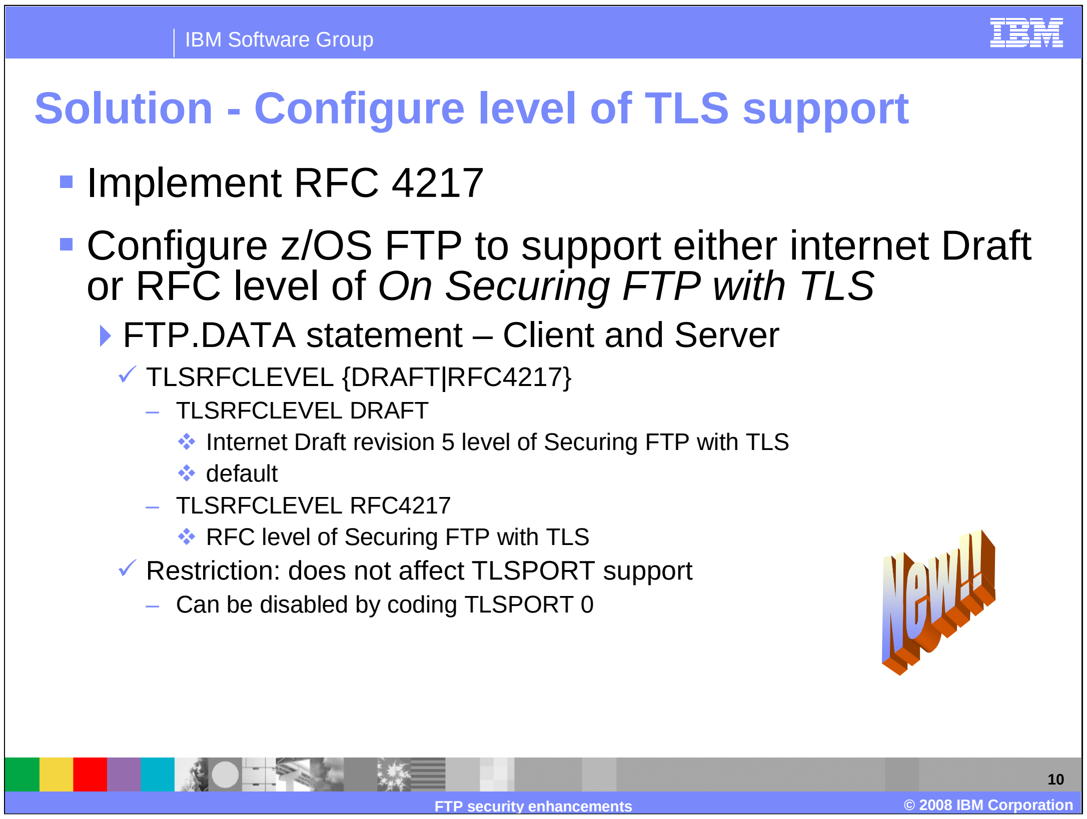
FTP security enhancements

© 2008 IBM Corporation

Since CSV1R2, Communications Server for z/OS FTP has supported TLS secured sessions. This support is based on revision 05 of the Internet Draft: *On Securing FTP with TLS*. The URL of revision 05 appears on this slide. The problem is that since the year 2000, the draft was revised eleven more times, and has now been published as an RFC. The CS for z/OS FTP support does not comply with the new standard, RFC 4217.

RFC 4217 has significant differences from the Internet Draft. The RFC is less restrictive than the draft about flowing the AUTH and CCC commands to the server during a secure session. The upshot of this is that the full RFC 4217 functionality of the AUTH and CCC commands is not available to z/OS FTP users. The RFC explicitly states that the REIN server command reply must flow on the protected connection – the server cannot clear the session before sending the reply. The Internet Draft did not specify this level of detail. The z/OS server implementation does not send the reply while the session is still protected; therefore, the z/OS FTP server does not interoperate with an RFC 4217 compliant FTP client when REIN is used during a TLS session. This is not as bad as you might think; REIN is not really recommended during an FTP session regardless of whether you are using TLS or not.

According to draft 05 of Securing FTP with TLS, when FTP clients connect to server port 990, the connection is secured with TLS without flowing an AUTH command – the connection is implicitly secured, as opposed to explicitly securing the connection by sending an AUTH command to the server. The RFC has dropped implicit security and secure port entirely. Thus, a connection between an RFC 4217 compliant FTP and an Internet Draft compliant FTP on the secure port cannot interoperate, because the Internet



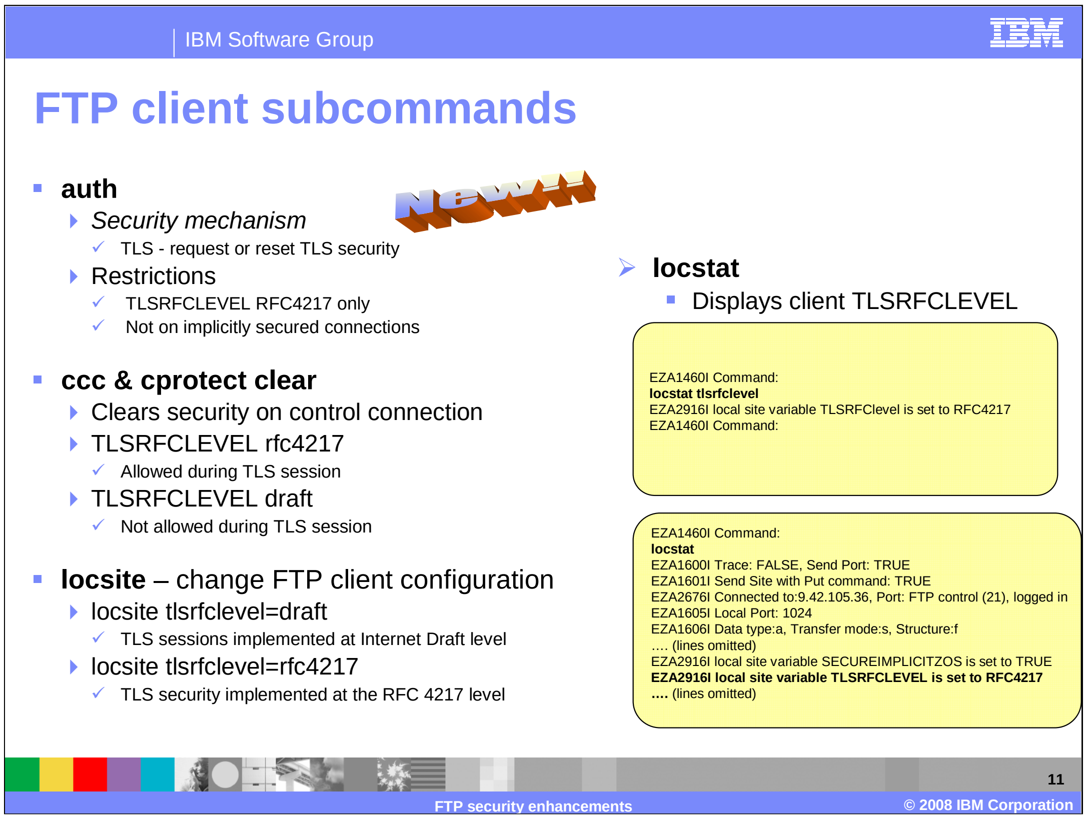
Solution - Configure level of TLS support

- Implement RFC 4217
- Configure z/OS FTP to support either internet Draft or RFC level of *On Securing FTP with TLS*
  - FTP.DATA statement – Client and Server
    - ✓ TLSRFCLEVEL {DRAFT|RFC4217}
      - TLSRFCLEVEL DRAFT
        - Internet Draft revision 5 level of Securing FTP with TLS
        - default
      - TLSRFCLEVEL RFC4217
        - RFC level of Securing FTP with TLS
    - ✓ Restriction: does not affect TLSPORT support
      - Can be disabled by coding TLSPORT 0

The solution is to implement RFC 4217 in z/OS in such a way as to support either RFC compliant sessions, or Internet Draft level sessions from earlier releases of FTP for CS for z/OS. FTP will support both levels, and provide a configuration option to select which level of TLS support you want.

The FTP.DATA statement TLSRFCLEVEL allows you to select which level of Securing FTP with TLS to use. This statement is supported for both the client and the server. TLSRFCLEVEL DRAFT is the default value, and is the value you would select if you wanted the same TLS support that z/OS FTP has offered since CSV1R2.

Even though RFC 4217 has dropped the secure port requirement, configuring TLSRFCLEVEL RFC4217 does not affect the z/OS FTP TLSPORT support. Connections to the port specified by the TLSPORT FTP.DATA statement (port 990 by default) are still implicitly secured with TLS. As in prior releases, you can disable implicit TLS security by coding TLSPORT 0 (zero) in FTP.DATA.



FTP client subcommands

New!!

- **auth**
  - *Security mechanism*
    - ✓ TLS - request or reset TLS security
  - Restrictions
    - ✓ TLSRFCLEVEL RFC4217 only
    - ✓ Not on implicitly secured connections
- **ccc & cprotect clear**
  - Clears security on control connection
  - TLSRFCLEVEL rfc4217
    - ✓ Allowed during TLS session
  - TLSRFCLEVEL draft
    - ✓ Not allowed during TLS session
- **locsite** – change FTP client configuration
  - locsite tlsrfclevel=draft
    - ✓ TLS sessions implemented at Internet Draft level
  - locsite tlsrfclevel=rfc4217
    - ✓ TLS security implemented at the RFC 4217 level

- **locstat**
  - Displays client TLSRFCLEVEL

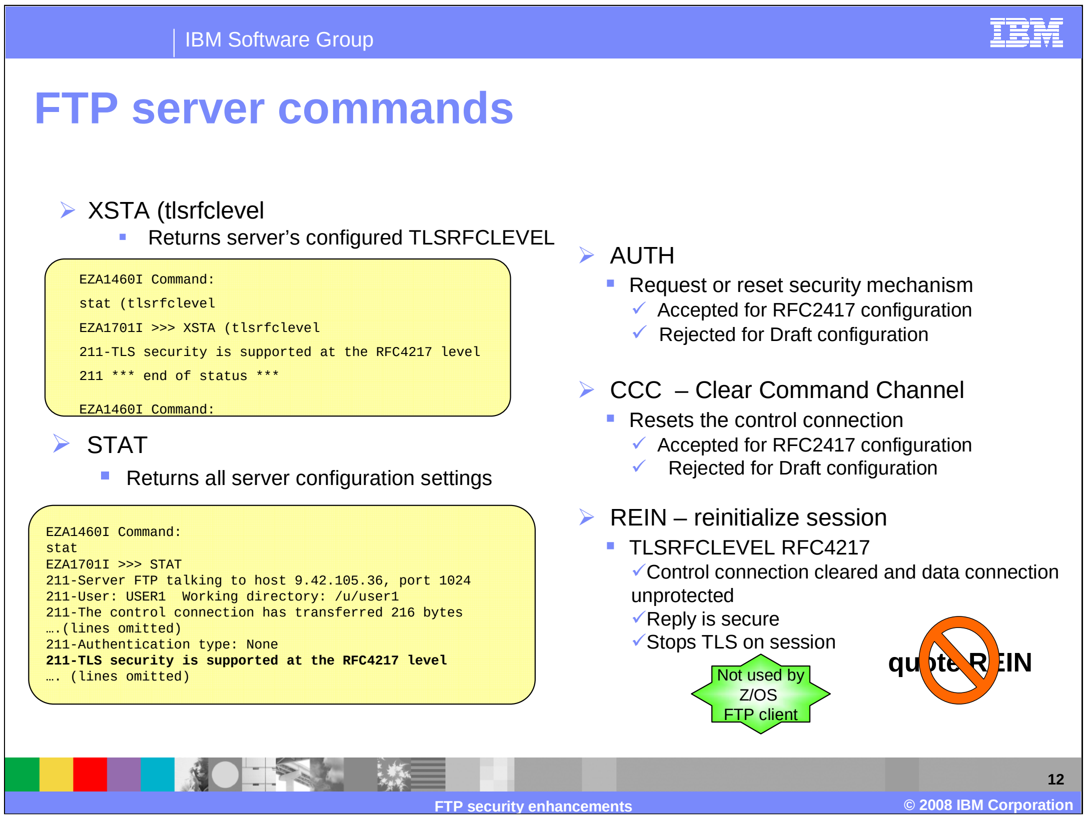
```
EZA1460I Command:
locstat tlsrfclevel
EZA2916I local site variable TLSRFClevel is set to RFC4217
EZA1460I Command:
```

```
EZA1460I Command:
locstat
EZA1600I Trace: FALSE, Send Port: TRUE
EZA1601I Send Site with Put command: TRUE
EZA2676I Connected to:9.42.105.36, Port: FTP control (21), logged in
EZA1605I Local Port: 1024
EZA1606I Data type:a, Transfer mode:s, Structure:f
.... (lines omitted)
EZA2916I local site variable SECUREIMPLICITZOS is set to TRUE
EZA2916I local site variable TLSRFCLEVEL is set to RFC4217
.... (lines omitted)
```

Since RFC 4217 allows you to reset a TLS session with a subsequent AUTH command to the server, the FTP client offers a subcommand to allow you to drive the AUTH command to the server. This subcommand is valid only when the FTP client has configured TLSRFCLEVEL RFC4217, and when the connection has not been implicitly secured with a connection to the TLSPORT. You can use the auth subcommand to reinstate TLS security after using the ccc subcommand. You could use the auth subcommand to request TLS security after logging into the server without security. In practice, however, it would be difficult to set up your security server to allow this. The FTP login userid would need access to the private cryptographic keys, an undesirable configuration (because it exposes the private keys).

The ccc and cprotect clear subcommands are part of base FTP support. These two subcommands are equivalent – they do the same thing. They each send a CCC command to the server. On an RFC 4217 compliant server, the CCC command clears security on the control connection; the data connection is left in its current state. When TLSRFCLEVEL RFC4217 is configured, the FTP client will allow you to run these subcommands while logged in on a TLS secured session. This is because RFC 4217 allows the CCC command to flow during the FTP session. When TLSRFCLEVEL DRAFT is configured, you get the current behavior which is that these subcommands are not allowed during a TLS session.

The locsite subcommand is enhanced to let you change the client’s tlsrfclevel. You have the same options as for the FTP.DATA statement. Use the locstat subcommand to display the client’s current TLSRFCLEVEL value. The locstat subcommand now supports the tlsrfclevel parameter to display only the TLSRFCLEVEL setting, as shown in this example. The mixed case TLSRFClevel shows that you could enter only TLSRFC (the capitalized portion) as an abbreviation for tlsrfclevel when entering the tlsrfclevel parameter. The comprehensive information displayed by the locstat subcommand when entered with no parameters now includes the TLSRFCLEVEL setting. The corresponding line of output appears in **bold font**.



FTP server commands

- XSTA (tlsrfclevel
  - Returns server's configured TLSRFCLEVEL

```
EZA1460I Command:
stat (tlsrfclevel
EZA1701I >>> XSTA (tlsrfclevel
211-TLS security is supported at the RFC4217 level
211 *** end of status ***
EZA1460I Command:
```

- STAT
  - Returns all server configuration settings

```
EZA1460I Command:
stat
EZA1701I >>> STAT
211-Server FTP talking to host 9.42.105.36, port 1024
211-User: USER1  Working directory: /u/user1
211-The control connection has transferred 216 bytes
….(lines omitted)
211-Authentication type: None
211-TLS security is supported at the RFC4217 level
…. (lines omitted)
```

- AUTH
  - Request or reset security mechanism
    - ✓ Accepted for RFC2417 configuration
    - ✓ Rejected for Draft configuration
- CCC – Clear Command Channel
  - Resets the control connection
    - ✓ Accepted for RFC2417 configuration
    - ✓ Rejected for Draft configuration
- REIN – reinitialize session
  - TLSRFCLEVEL RFC4217
    - ✓Control connection cleared and data connection unprotected
    - ✓Reply is secure
    - ✓Stops TLS on session

Not used by Z/OS FTP client

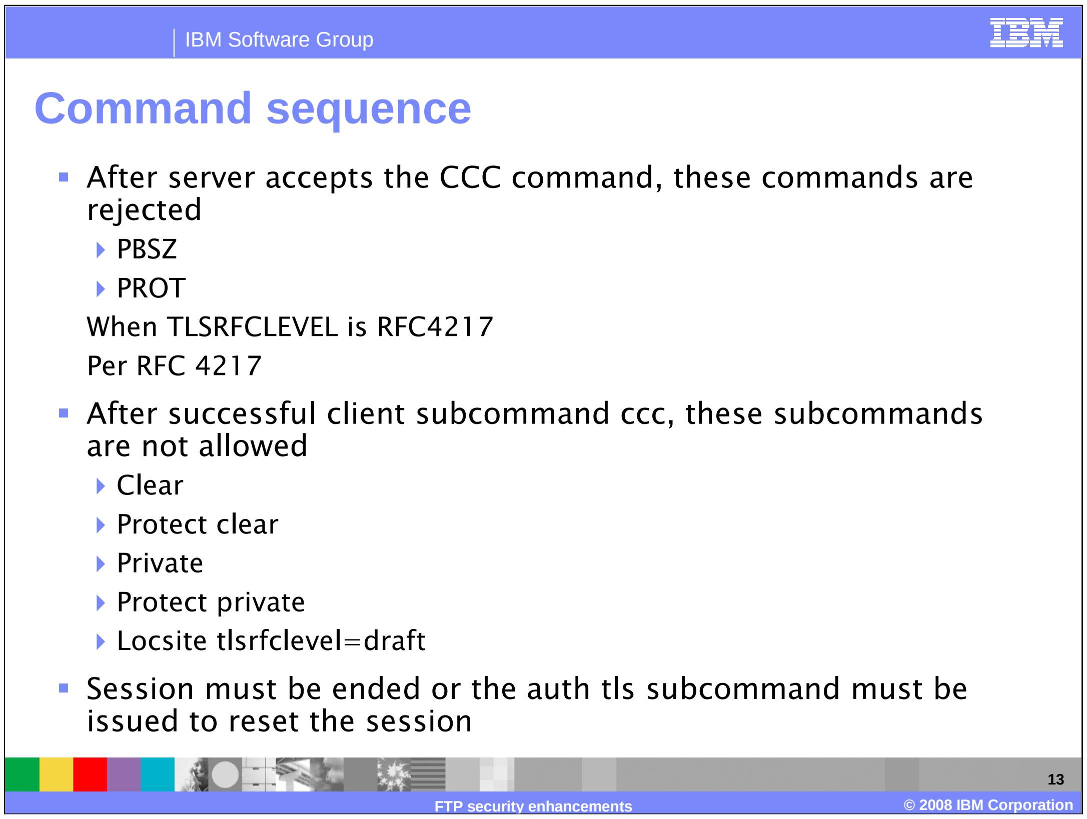
quote REIN

The server XSTA command has been enhanced to support the TLSRFCLEVEL parameter. This screen demonstrates the use of the z/OS FTP client status subcommand with the tlsrfclevel parameter to send an XSTA command to the server. From an OEM client, you could enter QUOTE XSTA (TLSRFCLEVEL to do the same thing.

The comprehensive information returned in the server STAT command reply now includes the TLSRFCLEVEL setting. The reply line that reports the TLSRFCLEVEL is in **bold font**.

When you configure TLSRFCLEVEL RFC4217, the server will now accept the AUTH and CCC commands during a TLS session. If you set TLSRFCLEVEL to draft, you get the earlier behavior which is to reject these commands during a TLS session. The AUTH command is used to request security for the current session. If your session is already secured, an AUTH command will reset security on the session. The CCC command is used to reset the control connection only. The data connection security is left in its current state, whatever that may be. You cannot use server commands to reset the data connection once you have used the CCC command. However, you can reinstate security by using the AUTH command.

RFC 4217 explicitly states that when the session is re-initialized with a REIN command, the control connection is cleared and the data connection reverts to unprotected. Therefore, the server's action changes from prior release behavior when TLSRFCLEVEL is set to RFC4217. The z/OS FTP client does not send the REIN command to the server unless you use the quote subcommand to send a REIN command to the server (the quote subcommand sends its argument to the server and waits for a server reply). Use of QUOTE REIN is not recommended from any FTP implementation regardless of whether you are using TLS security or not because it causes the client and server to lose synchronization.


IBM Software Group

Command sequence

- After server accepts the CCC command, these commands are rejected
  - PBSZ
  - PROT

  When TLSRFCLEVEL is RFC4217

  Per RFC 4217
- After successful client subcommand ccc, these subcommands are not allowed
  - Clear
  - Protect clear
  - Private
  - Protect private
  - Locsite tlsrfclevel=draft
- Session must be ended or the auth tls subcommand must be issued to reset the session

13

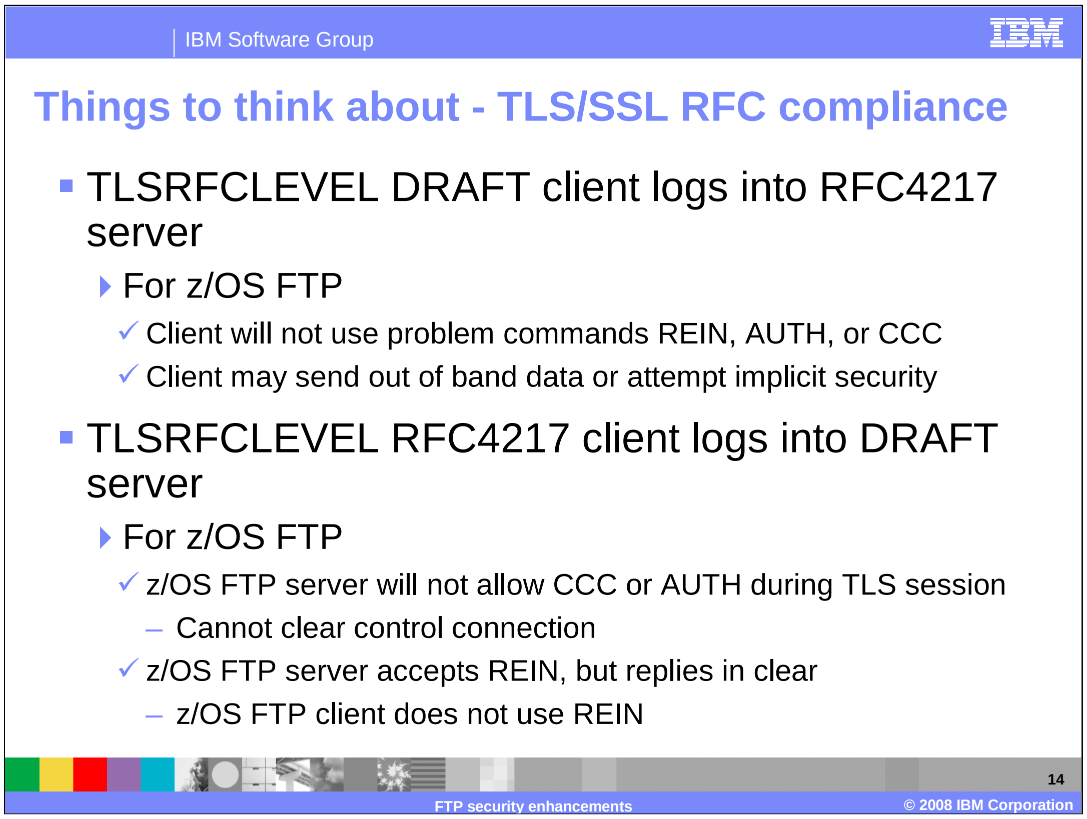
FTP security enhancements © 2008 IBM Corporation

RFC 4217 explicitly states that after the server accepts a CCC command, it must stop accepting PBSZ and PROT commands. Therefore, when TLSRFCLEVEL RFC4217 is configured, the server will reject PBSZ and PROT commands after it accepts the CCC command. You will not be able to reset the state of the data connection using these commands after CCC.

The ccc subcommand causes the FTP client to send a CCC command to the server. Since the server must reject all PBSZ and PROT commands after it accepts a CCC command, it is futile for the client to issue subcommands that require it to send PBSZ or PROT commands to the server – they would fail because the server must reject the commands. Therefore, the z/OS FTP client does not permit any subcommand which causes PBSZ or PROT to flow to the server. Such are the subcommands listed, except for **locsite**. The upshot is that you must set the data connection to the state you want before you issue the ccc subcommand.

Locsite with the tlsrfclevel=draft must be rejected because the ccc subcommand and CCC command leave the TLS session in a state which is not defined in the Internet Draft 05 of *On Securing FTP with TLS*.

You can end the session, or use the **auth tls** subcommand to reset the session, to restore use of these subcommands.


IBM Software Group

Things to think about - TLS/SSL RFC compliance

- TLSRFCLEVEL DRAFT client logs into RFC4217 server
  - For z/OS FTP
    - ✓ Client will not use problem commands REIN, AUTH, or CCC
    - ✓ Client may send out of band data or attempt implicit security
- TLSRFCLEVEL RFC4217 client logs into DRAFT server
  - For z/OS FTP
    - ✓ z/OS FTP server will not allow CCC or AUTH during TLS session
      - Cannot clear control connection
    - ✓ z/OS FTP server accepts REIN, but replies in clear
      - z/OS FTP client does not use REIN

14

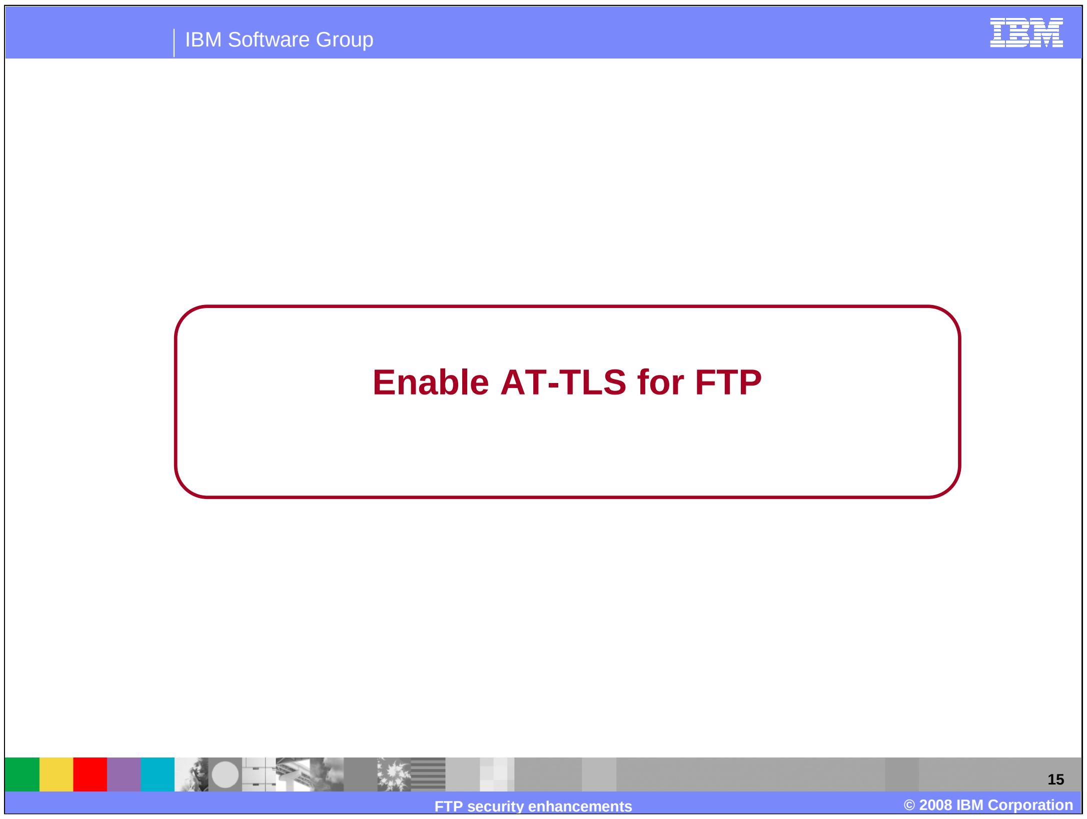
FTP security enhancements

© 2008 IBM Corporation

The primary conflicts between TLSRFCLEVEL DRAFT and TLSRFCLEVEL RFC4217 revolve around the CCC, AUTH, and REIN commands.

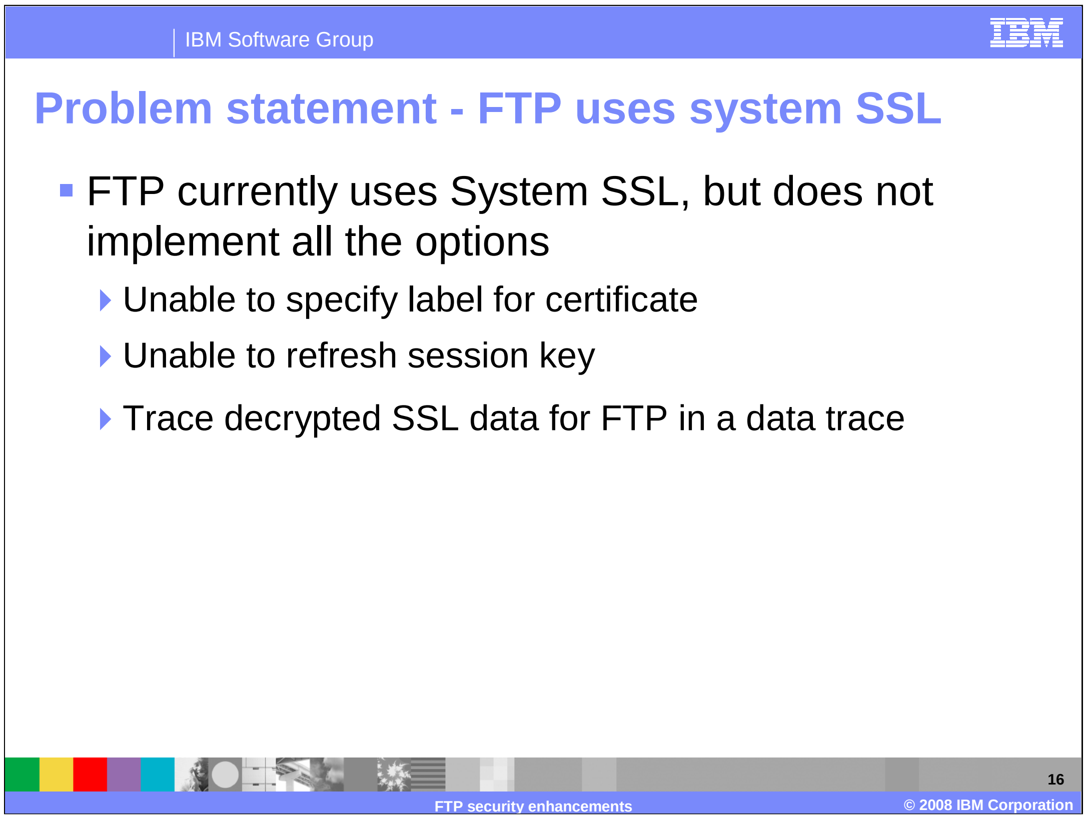
If a TLSRFCLEVEL DRAFT client logs into a TLSRFCLEVEL RFC4217 server, the session should be seamless because the draft level client won't allow the auth or ccc subcommand during a TLS login, and a z/OS FTP client will never send REIN to the server unless the user enters QUOTE REIN – this is not recommended. Nor, for that matter, is QUOTE AUTH or QUOTE CCC.   The z/OS FTP server still supports implicit connections and out of band data, although these are not RFC4217 options.   Other RFC 4217 FTP server implementations may not support implicit security or out of band data on the TLS session.

If an RFC 4217 FTP client logs into a TLSRFCLEVEL DRAFT server, the session should still be relatively seamless. The client may send AUTH or CCC to the server, but the server will reject these commands. The upshot is that you won't be able to clear the control connection of TLS security while logged in. If the client sends REIN to the server, this is a problem. The client will expect TLS to be reset on the control connection, but the server will not expect this. The z/OS FTP client never sends REIN, but other FTP client implementations might. QUOTE REIN is never recommended for any FTP implementation.



Enable AT-TLS for FTP

This section discusses the enablement of FTP to use AT-TLS.


IBM Software Group

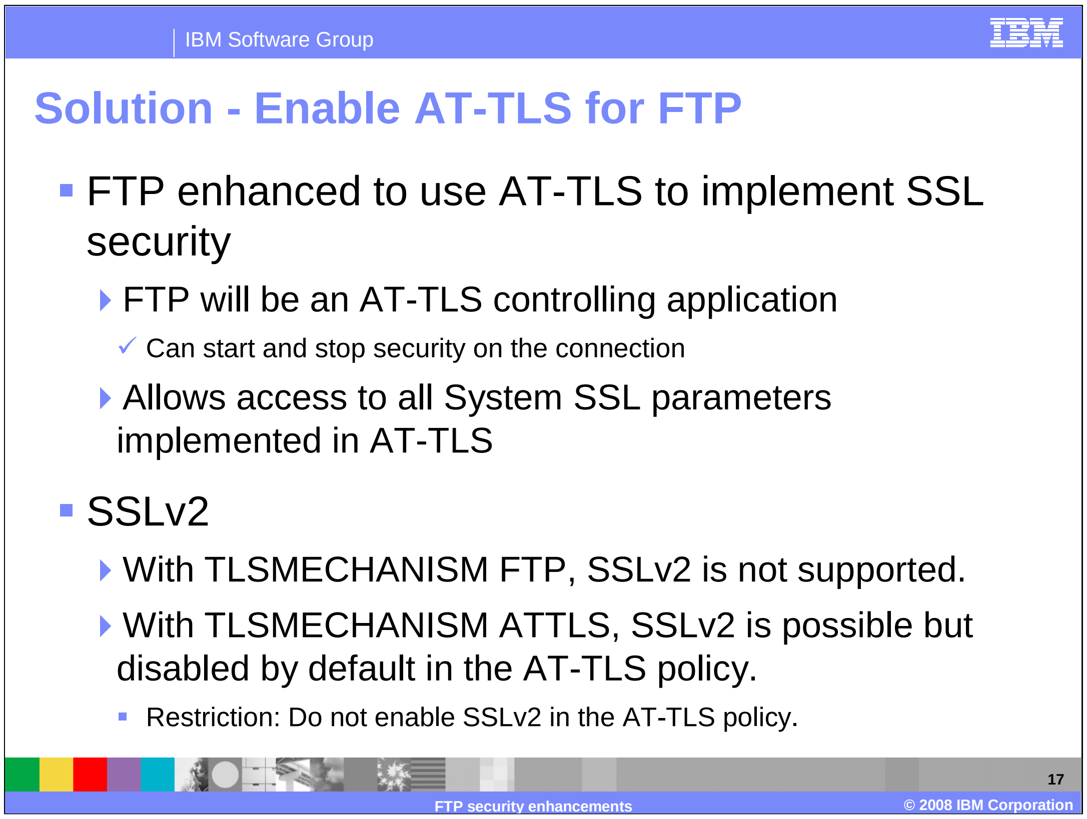
Problem statement - FTP uses system SSL

- FTP currently uses System SSL, but does not implement all the options
  - Unable to specify label for certificate
  - Unable to refresh session key
  - Trace decrypted SSL data for FTP in a data trace

16

FTP security enhancements © 2008 IBM Corporation

When FTP implemented System SSL in z/OS 1.2, all the functions of System SSL were not exploited. System SSL supports specifying a certificate label to allow certificates other than the default certificate to be used. System SSL also allows session keys to be refreshed during the lifetime of a session. Finally, the TCPIP data trace can be used to trace the decrypted data read by FTP.


IBM Software Group

Solution - Enable AT-TLS for FTP

- FTP enhanced to use AT-TLS to implement SSL security
  - FTP will be an AT-TLS controlling application
    - ✓ Can start and stop security on the connection
  - Allows access to all System SSL parameters implemented in AT-TLS
- SSLv2
  - With TLSMECHANISM FTP, SSLv2 is not supported.
  - With TLSMECHANISM ATTLS, SSLv2 is possible but disabled by default in the AT-TLS policy.
    - Restriction: Do not enable SSLv2 in the AT-TLS policy.

17

FTP security enhancements

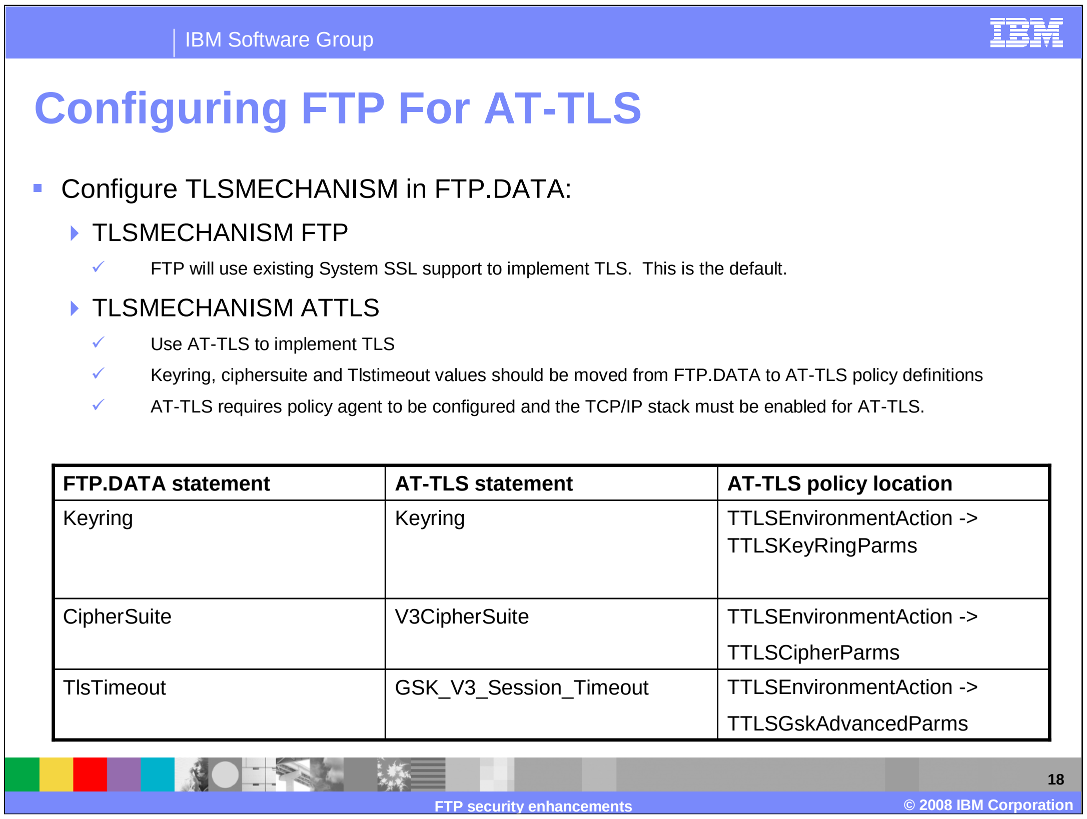
© 2008 IBM Corporation

The FTP client and server can now be configured to use AT-TLS to support SSL/TLS connections.

There are 3 types of AT-TLS applications. Those that are completely unaware they are using AT-TLS (they use AT-TLS with no code changes at all), those that have AT-TLS awareness but do not control AT-TLS (they can query the stack but not affect the choices it makes), and those that are AT-TLS controlling, meaning the application starts and stops security on the connection. FTP is a controlling AT-TLS application which requires the ApplicationControlled On statement in the AT-TLS policy.

Using AT-TLS allows all of the System SSL parameters supported by AT-TLS to be configured for FTP. For example, a certificate label can be configured instead of the default certificate. AT-TLS can also be configured to refresh the session key on a connection.

FTP does not support SSLv2 when configured with TLSMECHANISM FTP. AT-TLS does allow SSLv2 connections to be configured, but by default SSLv2 is disabled. SSLv2 should not be enabled in the AT-TLS policy for FTP.


IBM Software Group

Configuring FTP For AT-TLS

- Configure TLSMECHANISM in FTP.DATA:
  - TLSMECHANISM FTP
    - ✓ FTP will use existing System SSL support to implement TLS. This is the default.
  - TLSMECHANISM ATTLS
    - ✓ Use AT-TLS to implement TLS
    - ✓ Keyring, ciphersuite and Tlstimeout values should be moved from FTP.DATA to AT-TLS policy definitions
    - ✓ AT-TLS requires policy agent to be configured and the TCP/IP stack must be enabled for AT-TLS.

FTP.DATA statement	AT-TLS statement	AT-TLS policy location
Keyring	Keyring	TTLSEnvironmentAction -> TTLSKeyRingParms
CipherSuite	V3CipherSuite	TTLSEnvironmentAction -> TTLSCipherParms
TlsTimeout	GSK_V3_Session_Timeout	TTLSEnvironmentAction -> TTLSGskAdvancedParms

18

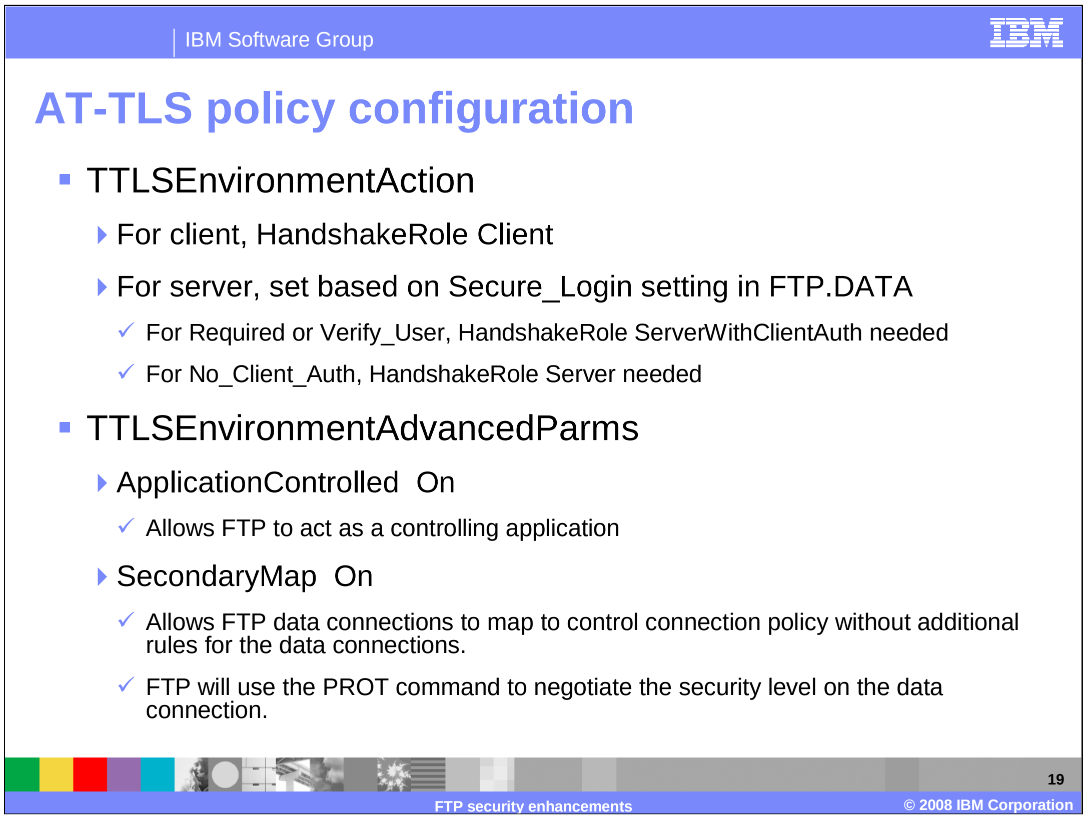
FTP security enhancements

© 2008 IBM Corporation

A new configuration statement for the FTP client and server, TLSMECHANISM has been created. The default value is FTP, which causes FTP to use the existing System SSL support.

Configuring ATTLS causes FTP to use AT-TLS. This requires AT-TLS policy updates to support AT-TLS on FTP connections. The existing FTP.DATA parameters Keyring, TlsTimeout and Ciphersuite will need to be moved to the AT-TLS policy. The TCP/IP stack must be enabled for AT-TLS and the policy agent must be configured to support AT-TLS.

The TTLSEnvironmentAction statement contains the equivalent AT-TLS parameters. The FTP.DATA Keyring statement will need to be moved to the AT-TLS policy Keyring statement, which is in the TTLSKeyRingParms statement. The CipherSuite statements should be moved to a V3CipherSuite statement in the AT-TLS policy TTLSCipherParms statement. The FTP.DATA TlsTimeout statement should be moved to the GSK_V3_Session_Timeout statement in the AT-TLS policy TTLSGskAdvancedParms statement.


IBM Software Group

AT-TLS policy configuration

- TTLSEnvironmentAction
  - For client, HandshakeRole Client
  - For server, set based on Secure_Login setting in FTP.DATA
    - ✓ For Required or Verify_User, HandshakeRole ServerWithClientAuth needed
    - ✓ For No_Client_Auth, HandshakeRole Server needed
- TTLSEnvironmentAdvancedParms
  - ApplicationControlled On
    - ✓ Allows FTP to act as a controlling application
  - SecondaryMap On
    - ✓ Allows FTP data connections to map to control connection policy without additional rules for the data connections.
    - ✓ FTP will use the PROT command to negotiate the security level on the data connection.

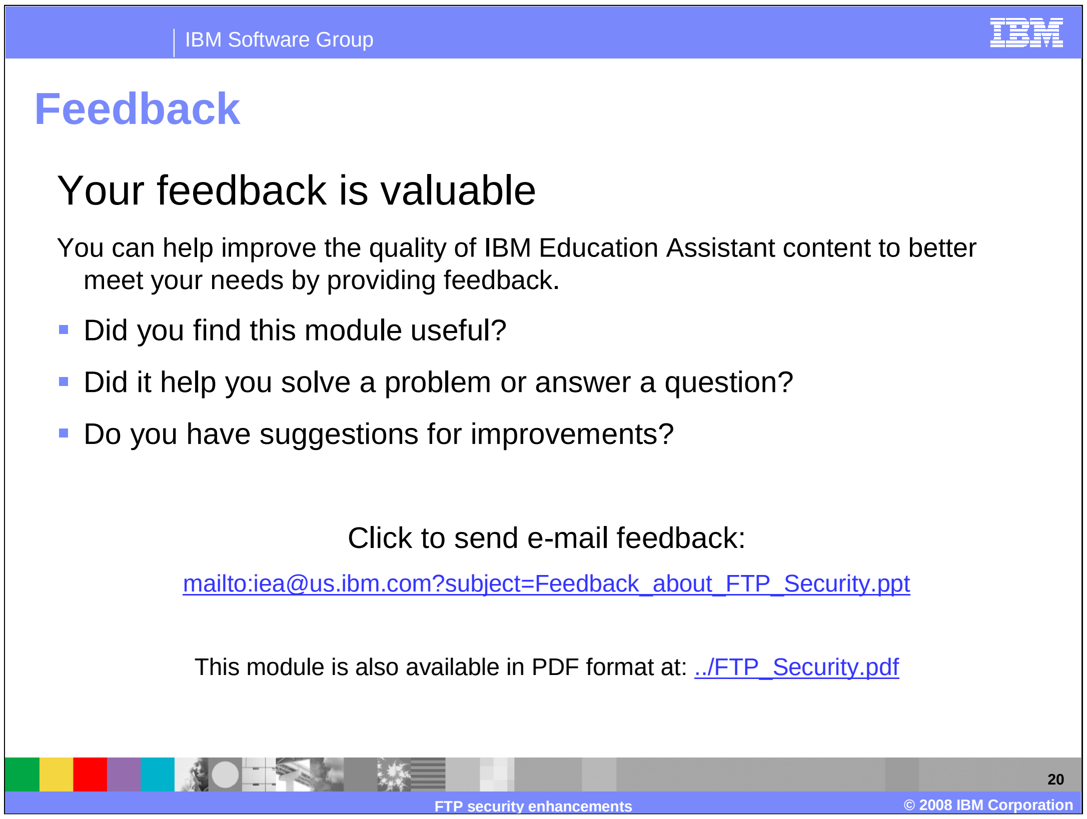
19

FTP security enhancements

© 2008 IBM Corporation

The TTLSEnvironmentAction statement also configures the HandshakeRole for the SSL connection. For the FTP client, the HandshakeRole must be set to Client. For the FTP server, the HandshakeRole is set based on the value of the Secure_Login setting in FTP.DATA. If Secure_login is configured as Required or Verify_User, the HandshakeRole must be set to ServerWithClientAuth. For the default Secure_Login statement of No_Client_Auth, the HandshakeRole must be Server.   The SSL handshake is initiated by the endpoint of the connection acting as HandshakeRole Client. The endpoint of the connection acting as Server expects to receive the initial SSL handshake request from the remote endpoint. If the HandshakeRole is not set correctly, the handshake will either timeout when both sides are set to Server or immediately end if both sides are set to Client. When using ServerWithClientAuth, a certificate will be requested from the remote partner.

A TTLSEnvironmentAdvancedParms statement is required for FTP to use AT-TLS. The ApplicationControlled statement must be set to On. This allows FTP to start and stop security on a connection.   The SecondaryMap statement should also be set to On so that data connections will have the same policy as the control connection. SecondaryMap eliminates the need to code additional TTLSRule statements for the data connection. The FTP protocol will negotiate the security level to be used on the data connection.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

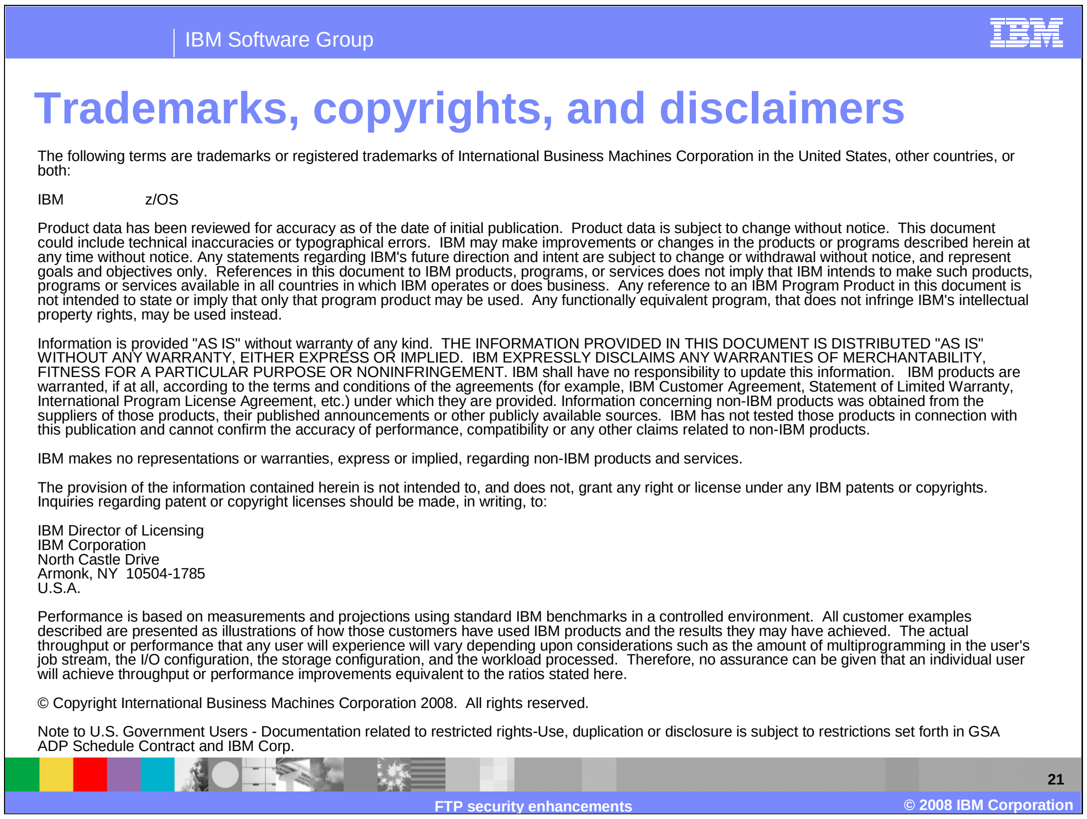
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_FTP_Security.ppt

This module is also available in PDF format at: ../FTP_Security.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.



IBM Software Group

Trademarks, copyrights, and disclaimers

21

FTP security enhancements © 2008 IBM Corporation