



CSI TCP/IP Update

Don Stoeber

CSI International



Agenda

- What's new in TCP/IP 2.3.3
 - Initially released in December 2023
 - Multiple pre-applied service packs
 - Final release was in February 2025
- What's new in 2.3.4
 - Initially released in March 2025
 - Multiple pre-applied service packs
 - Current release pre-applied service pack is June 2025



What's New in 2.3.3 ?

- Initial work on handling Syn-Flood attacks
- Needed for VSE systems on the Open Internet
- LISTENFAIL command introduced
- DEFINE ROUTE CRETRAN= DRETRAN= MINRET= MAXRET= RETRY= keywords are specified in milliseconds(1/1000ths of a second).
 - 1000 equals 1 second.



What's New in 2.3.3 ?

- OPENSLL corrections
- ID: INTERNALLOOPBACK Link ID: LOOPBACK
 - IP Address: 127.0.0.0
- ID: INTERNALHOME Link ID: LOOPBACK
 - IP Address: 192.168.1.242 SET IPADDR=
- Improved performance by removing unnecessary calls to the lock manager
- Abend exit(STXIT PC) added to message writer subtask



What's New in 2.3.3 ?

- Common Encryption Cipher Interface(CECI)
 - SHA hash of the data block
- FTP TLS\SSL explicit mode corrected
- DEFINE TELNETD with POOL=YES
 - Corrected Timer hang
- IPNOME.Z Online Messages File(OME) updated
- FTP data connection send buffer size message
 - FTP356I FTP data connection send buffer size:65520
MaxUnack:262080



What's New in 2.3.3 ?

- Automation services lock for events using Power Queue Class
- External AUTOSEND service using CLIENTDZ
 - Move define events to external partition
- TCP SYN-FLOOD attack causes excessive storage usage
 - Reduced size of statistics block(ISBLOK)
- SeeVSE dataspace collection of datagrams



What's New in 2.3.4 ?

- BLOCKIP command added
 - Part of FIREWALL feature
- Added SSL\TLS support into Email client
 - New EMLBATCH phase





OPENSSL

- **OPENSSL** is distributed and supported by 21st Century and IBM
- Your system must be z/VSE 6.2 or VSEn to use the **OPENSSL** service in conjunction with CSI SSL/TLS applications and applications using the CSI documented SSL/TLS API.
- Before attempting to implement **OPENSSL** you should also contact IBM or 21st Century technical support and verify your system has the most recent service applied for **OPENSSL**.



OpenSSL

- Cryptographic server provider(CSP) can be either CSINTSSL or OPENSSL
- Default CSP is the CSI implementation(CSINTSSL)
- OPENSSL can be activated either globally or in selected partitions using the job control statement:
 - // SETPARAM CRYSERV='OPENSSL'
 - in JCL of job running the SSL/TLS application to use OPENSSL
 - // SETPARAM CRYSERV=' CSINTSSL'
 - in JCL of job running the SSL/TLS application to use CSINTSSL
- Optionally, custom \$SOCKOPT and \$SOCDBG phases can be assembled and cataloged but it is easier and recommended to use the OPENSSL command or job control SETPARAM



OPENSSL command

- OPENSSL action
 - Global and affects all stacks and partitions
 - Only one action is allowed at a time
 - Actions are ON/OFF/DEBUGON/DEBUGOFF/SDUMPON/SDUMPOFF
- OPENSSL ON to globally activate usage of OPENSSL in all stacks and partitions
 - Once it is issued all stacks and partitions will use OPENSSL when negotiating SSL/TLS sessions therefore all SSL/TLS applications should be shutdown before issuing OPENSSL ON
- OPENSSL OFF to globally deactivate
- OPENSSL DEBUGON activates debugging messages
- OPENSSL DEBUGOFF deactivates debugging messages
- OPENSSL SDUMPON activates sdumps
 - DEBUGON must be active for SDUMPs to occur
 - // OPTION NOSYSDMP to send sdumps to SYSLST instead of the dump lib.sublib is recommended
- OPENSSL SDUMPOFF deactivate sdumps
- QUERY SET can be used to display the current OPENSSL settings
 - IPN806R OPENSSL: On DEBUGOFF SDUMPOFF



OPENSSL SETPARAM CRYSERV

- A job control SETPARAM for the variable CRYSERV can be used to set the cryptographic service provider in a single partition.
- This is probably the easiest method and overrides any other previous method.
- The variable CRYSERV can be set to either CSINTSSL or OPENSSL. Here is an example of using this method:
- `// JOB SAMPSETP`
- `...other job control statements...`
- `// SETPARAM CRYSERV='CSINTSSL'`
- `// SETPARAM CRYSERV='OPENSSL'`
- `...other job control statements...`
- When CRYSERV is equal to CSINTSSL the legacy CSI Internal SSL\TLS interface will be used.
- When CRYSERV is equal to OPENSSL the OPENSSL interface will be used.



OPENSSL Performance

- CPU usage with OPENSSL and CSINTSSL has were about the same during our internal QA testing
- Carefully select the cipher suites
 - Hardware supported algorithms can have a large impact for lowering CPU usage
 - Software implemented algorithms should be measured for CPU usage
- OPENSSL does require more virtual storage than CSINTSSL, but this is to be expected in that it supports a much larger number of algorithms and functions.
- Any partition using OPENSSL will need at least 4.5meg of additional 31-bit partition getvis.
 - This also applies when using OPENSSL in the TCP/IP stack partition.
- When converting to OPENSSL you should monitor CPU and GETVIS usage in partitions using it at your installation



CSINTSSL Cipher Suites

- CSINTSSL for SSL30, TLS10, and TLS11 supports the following:
 - Cipher Suite 35 RSA_AES256CBC_SHA1
 - Cipher Suite 2F RSA_AES128CBC_SHA1
 - Cipher Suite 0A RSA_3DESCBC_SHA1
 - Cipher Suite 09 RSA_DESCBC_SHA1
 - Cipher Suite 08 RSA_DES40CBC_SHA1
- CSINTSSL for TLS12 supports the following:
 - Cipher Suite 3D RSA_AES256CBC_SHA256
 - Cipher Suite 3C RSA_AES128CBC_SHA256
 - Cipher Suite 35 RSA_AES256CBC_SHA1
 - Cipher Suite 2F RSA_AES128CBC_SHA1



OPENSSL Cipher Suites

- OPENSSL for TLS12 supports the following:
 - Cipher Suite 2F RSA_AES128CBC_SHA160_PRF256
 - Cipher Suite 35 RSA_AES256CBC_SHA160_PRF256
 - Cipher Suite 3C RSA_AES128CBC_SHA256_PRF256
 - Cipher Suite 3D RSA_AES256CBC_SHA256_PRF256
 - Cipher Suite 9C RSA_WITH_AES_128_GCM_SHA256
 - Cipher Suite 9D RSA_WITH_AES_256_GCM_SHA384
 - Cipher Suite C013 ECDHE-RSA-AES128-SHA
 - Cipher Suite C014 ECDHE_RSA_WTH_AES_256_CBC_SHA
 - Cipher Suite C027 ECDHE_RSA_WTH_AES_128_CBC_SHA256
 - Cipher Suite C02F ECDHE_RSA_WTH_AES_128_GCM_SHA256
 - Cipher Suite C030 ECDHE_RSA_WTH_AES_256_GCM_SHA384



OPENSSL Cipher Suites

- OPENSSL for TLS13 supports the following:
 - Cipher Suite 1301 AES_128_GCM_SHA256
 - Cipher Suite 1302 AES_256_GCM_SHA384
 - Cipher Suite 1303 CHACHA20_POLY1305_SHA256

As noted in the IETF RFC8446 specification for TLS 1.3 Appendix B: *Although TLS 1.3 uses the same cipher suite space as previous versions of TLS, TLS 1.3 cipher suites are defined differently, only specifying the symmetric ciphers and cannot be used for TLS 1.2. Similarly, cipher suites for TLS 1.2 and lower cannot be used with TLS 1.3.*



OPENSSL Differences from CSINTSSL

gsk_initialize() INI@KRFN

OPENSSL requires all client applications to provide a certificate. The **CSINTSSL** service without client authentication allowed this field to contain binary zeros since the client did not send a certificate to the server unless client authentication was being used. When using OPENSSL the below must be set to the address of the key lib.sublib name even when in client mode without client authentication.

gsk_secure_soc_init()

OPENSSL provides a plethora of new cipher suites which is great. The available cipher suites will change as IBM ports over newer versions of the **OPENSSL** code. But the **CSINTSSL** service allowed applications to set the cipher specifications field to binary zeros which was then interpreted to mean ALL available cipher suites.

When using OPENSSL the following fields must be set to the address of cipher suites the application will propose when in client mode, and accept when in server mode.

Upon successful completion the CSINTSSL will contain a pointer in the SIN@PROT of a 5 character null terminated string containing the version of the protocol (SSL30, TLS10, TLS11, TLS12) that was set by the server during the SSL/TLS handshake.

OPENSSL does not provide this information. This can affect some TCP/IP messages such as:

- FTP922S Control connection using SSL TLS12 Cipher=003D
- IPA411I FTP Client connection using TLS12 Cipher=003D
- Which were produced when using CSINTSSL.
- But when using OPENSSL the protocol version will be missing and look like this:
- FTP922S Control connection using Cipher=003D
- IPA411I FTP Client connection using Cipher=C02F
- Note the blank area in the message when using OPENSSL

gsk_secure_soc_read()

To protect against possible truncation attacks the SSL/TLS protocol requires that Close_Notify alerts be exchanged on the connection before closing the TCP connection, but many TCP applications will close the connection without verifying the Close_Notify's have been exchanged and assume that a receive of zero bytes indicates the TCP connection was closed gracefully. The CSINTSSL in addition to the zero-byte length will return 4 bytes xFFFFF82E (-2002) into the callers receive buffer to allow the application to verify that a Close_Notify has been received. When using OPENSSL this value is not returned in the callers receive buffer.



SSL\TLS Sample Certificate Files

In prior releases of TCP/IP a sample RSA private key, certificate, and root certificate were created and distributed by CSI to expedite the implementation of SSL\TLS.

This is documented in the Optional Features guide.

These sample files could cause a security exposure and are no longer created and distributed by CSI.

Instead, the free Keyman utility should be used to create the files to be used by SSL/TLS applications at your site.



\$SOCKDBG SETPARAM

A job control SETPARAM for the variable SOCKDBG can also be used to activate debugging mode in a single partition.

Here is an example of using this method:

```
// JOB SAMPSETD
```

```
...other job control statements...
```

```
// SETPARAM SOCKDBG='YES'
```

```
...other job control statements...
```

When SOCKDBG is equal to YES additional diagnostic messages will be issued for the BSD and SSL\TLS CSI API's.

This replaces the need to assemble and linkedit a custom \$SOCKDBG.PHASE for debugging.



DEFINE TELNETD

- **TELNETD Deprecated Keywords**

- The DEFINE TELNETD had undocumented keywords that are no longer accepted. CAFID=, LIBRARY=, and MEMBER= and their associated settings are not used for anything. They must be removed from your DEFINE TELNETD command.
- The LUNAME= was a undocumented synonym for TARGET=. You must change LUNAME= to be TARGET=.

In previous releases a DEFINE TLSD with matching PORT= and PASSPORT= to the DEFINE TELNETD PORT= was used to contain the SSL/TLS definitions. These are now replaced with new the DEFINE TELNETD SSL keywords. The DEFINE TLSD must be removed and the following new keywords should be used on the DEFINE TELNETD.

- **SSL=NO/YES/YESCLAUTH**
 - SSL=NO is the default for no SSL\TLS secured connections.
 - SSL=YES to use SSL\TLS secured connections
- **SSLVERSION=SSL30/TLS10/TLS11/TLS12/TLS13**
 - SSLVERSION=TLS12 is the recommended version. Clients connecting into this TELNETD must support this version of the protocol.
- **SSLCIPHER= ALL/STRONG/ MEDIUM/WEAK/CUSTOM/NULL**
 - SSLCIPHER=ALL is recommended since it will automatically use all supported cipher suites.
- **SSSLIBRARY=** is the VSE library containing the certificate files.
- **SSLSUBLIBRARY=** is the VSE sub-library containing the certificate files.
- **SSLMEMBER=** is the VSE library member name of the certificate files.



DEFINE TELNETD

The default DEFINE TELNETD DRIVER= keyword is TN3270DX.

The default DRIVER= was TELNETD in some older releases of TCP/IP(1.5E-F).

The TELNETD phase is still distributed but is functionally stabilized.

Sites may have used DRIVER=TELNETD on their DEFINE TELNETD.

This should be removed to allow the default TN3270DX phase to be used.

Alternately, it could be changed to DRIVER=TN3270DX.

The functionally stabilized TELNETD phase has not been enhanced to support the new SSL\TLS keywords.



DEFINE FTPD Changes

- When defining an internal SSL/TLS server with the DEFINE FTPD command either OPENSSL or CSINTSSL uses the SSLCIPHER= keyword to set the ciphers that the server will allow.
- Prior to 2.3 this keyword allowed the values ALL, WEAK, MEDIUM, STRONG, AES, DES, NULL, or HARDWARE. The actual ciphers these corresponded to were hard coded in the FTPDAEMN and are replaced with the customizable ciphers in the \$SOCKOPT.PHASE.
- These cipher suites associated with these keywords now correspond to cipher suites in the \$SOCKOPT.PHASE. The SSLCIPHER= keyword should now contain ALL, STRONG, MEDIUM, WEAK, CUSTOM, or NULL. The HARDWARE, AES, and DES will still be accepted but an [FTP403W warning message will be issued and the SSLCIPHER will be overridden to ALL.](#)



AUTO-FTP using SSL/TLS

To use SSL/TLS with an AUTO FTP event (DEFINE EVENT command) you must do the following.

- Create an FTP server on z/VSE that support SSL/TLS similar to this:
- // JOB TCPFTSSL
- // EXEC LIBR,PARM='MSHP'
- ACCESS SUBLIB=CSILIB.ZVSE62DS
- CATALOG TCPFTSSL.L REP=YES
- *
- * * EXECUTING TCP/IP COMMANDS FROM TCPFTSSL.L
- **DEFINE FTPD,ID=FTPD0990,PORT=990,COUNT=3, -**
- **SSL=YES,SSLMODE=IMPLICIT, -**
- **SSLVERSION=TLS12,SSLCIPHER=ALL, -**
- **SSLDATACONN=PRIVATE,SSLKEY=CRYPTO.KEYRING.SAMPLE04**
- /+
- /*
- /&



AUTO-FTP using SSL/TLS

- Define the EVENT:

```
// JOB TCPELSTS
// EXEC LIBR
ACC SUB=CSILIB.ZVSE62DS
CATALOG TCPELSTS.L          REPLACE=YES
*
** EXECUTING COMMANDS FROM TCPELSTS.L
```
- **DEFINE EVENT, ID=AFTPLSTS, TYPE=POWER, CLASS=S, QUEUE=LST, ACTION=FTP**

```
/*
/*
/*
/;&
```
- Define a name for the remote FTP server like this:

```
DEFINE NAME NAME=CSIDON01, IPADDR=remote_server_ipaddress
```
- Create an AUTOFTP script like this using the defined name on the OPEN for the remote:

```
// JOB AFTPTSSL
// EXEC LIBR, PARM='MSHP'
ACCESS SUBLIB=lib.sublib
CATALOG AFTPTSSL.L  REP=YES
```
- **LOPEN 127.0.0.1 990 IMPLICIT TLS12**
- **LUSER DON**
- **LPASS ZVSE1234**
- **OPEN CSIDON01 990 IMPLICIT TLS12**

```
USER foreign-ftpserver-userid
PASS foreign-ftpserver-password
LCD /POWER/LST/S
PRIVATE
CD /foreign-directory
SETVAR &LFN = &PWRNAME + "." + &PWRNUMB + "." + &PWRSUFF
SETVAR &PCNAME = &LFN + ".LST"
LSITE CC OFF
LSITE STRIP ON
ACTIVE
PUT &LFN &PCNAME
CLOSE
LCLOSE
/*
/*
/;&
```
- That is it in a nutshell...



FTP Client

- New IPNAFTPC.L Member
- The first set of settings are the defaults and are used when
- no matching BEGIN name - END name is found from the OPEN command.
- **BEGIN DEFAULT** **Default settings must be first**
- **TLS CIPHERS ALL** **Use all supported strong cipher suites**
- **TLS LIBRARY CRYPTO**
- **TLS SUBLIB KEYRING**
- **TLS MEMBER SAMPLE04**
- **END DEFAULT** **End of default settings**
- **.***
- **BEGIN CUSTOM01** **Begin settings for this remote server**
- **SET FTPCFTIM 9000** **Foreign command receive response timeout**
- **TLS CIPHERS STRONG** **Use these SSL/TLS cipher suites from \$SOCKOPT**
- **TLS LIBRARY CRYPTO**
- **TLS SUBLIB KEYRING**
- **TLS MEMBER SAMPLE04**
- **END CUSTOM01**



FTP Client

In prior releases a single buffer was used to receive local and remote directory data. If this buffer became full a line split could occur resulting in incorrectly formatted directory output. This is now enhanced to automatically combine the buffers to avoid any line splits.

```
SET FTPCLLN5 32768  
IPNAFTPC.L
```

Default directory buffer length in



Detecting from a program that a TCP/IP stack is up

- In TCP/IP 1.5E-F the SOCKETnn SDL entry was removed from the SDL when TCP/IP was shutdown. This could cause a z/VSE system error and is no longer removed. The scanning of the SDL for the SOCKETnn to determine that TCP/IP is up and operational should be removed.
- The best way to detect if the stack is up and operational is open a control connection with the stack and send a GETVERSION request. This will also return the vendor and the version of stack that is running. See the Programmers Guide for information on opening a control connection to the stack.



Store Into Instruction Streams (SIIS) Performance Issue

A performance issue on the latest IBM processors (z13s, z14, z15) because of Store Into Instruction Streams (SIIS) has been identified and corrected. For more detailed information on this topic Google “IBM SIIS”.



CSI Licensed Sites Product Code Checking Enhancements

- Most other CSI products use a common utility (CSIPKINT) to manage their product keys. CSIPKINT is normally run once at IPL. TCP/IP can now optionally use the CSI common product code checker to validate its product key. This should simplify and centralize product keys for all CSI VSE products. There is a readme link and download available at:
- <http://www2.csi-international.com/AUTHCODES>
- The **CHECKPRD** command can also be used to activate a new product keys (PRODKEYS.PHASE) without having to shut down and restart TCP/IP.



Stack Task Priorities

TCP/IP has 6 subtasks and 1 main task. The main task is always the lowest priority and after initialization does not perform any work. The other subtasks are:

- MESSAGE – IPMSSG is the message writer. All messages written to SYSLOG, SYSLST, and any other logs defined with the DEFINE LOG command are queued to this subtask.
 - COMMAND – CMDTASK is the command processor. It processes all commands.
 - FILEIO – IPFILEIO is the file I/O processor. File open, read, write, and close operations are performed by this subtask for all files in the defined file system.
 - CSOCKET is the socket request handler. All application socket requests are queued to this task. It then spawns a pseudo task to the appropriate socket manager such as UDP, TCP, etc.
 - DRIVER – IPDRIVER is the pseudo task manager. It supports up to 65,000 pseudo tasks and handles the work for all connection sessions and other processes such as event processing, etc.
 - IPSPINCP – is always the highest priority and it monitors the other subtasks mainly for loop detection. It can now also adjust the priorities of the above other 5 subtasks.
- The default priorities of the above subtasks are established at initialization using the standard VSE change priority (CHAP) macro which makes the issuing task the lowest priority subtask in the partition.



Stack Task Priorities

By default, the **PRTYSHARE ON** command make all subtasks equal in priority. You can also use the following keywords:

- **PRTYSHARE ON DRIVER=nnn CSOCKET=nnn FILEIO=nnn**
- The nnn in the above is the weighting factor. The higher the nnn weighting factor the higher the subtasks priority.
- It is not possible to give recommendations as to what values to use since it depends on the type of workload your system is performing, but here are some examples:
- **PRTYSHARE ON DRIVER=16**
- This will cause the DRIVER subtask to have 16x higher priority over the other subtasks.
- **PRTYSHARE ON FILEIO=8**
- This will cause the FILEIO subtask to have 8x higher priority over the other subtasks.
- But only $\frac{1}{2}$ the priority over DRIVER since it is set to 16.
- **PRTYSHARE ON CSOCKET=4**
- This will cause the CSOCKET subtask to have 4x higher priority over the other subtasks.
- But only $\frac{1}{4}$ the priority over DRIVER since it is set to 16. And $\frac{1}{2}$ higher priority over FILEIO since it is set to 8.



SOCKDBG command

SOCKDBG is a new TCP/IP command. In previous releases it was required to assemble and linkedit and custom \$SOCKDBG.PHASE to activate debugging or turn off debugging. This also required that the partition the application was running in had to be recycled to pick up the \$SOCKDBG.PHASE. This technique can still be used, but now the SOCKDBG command can be used instead removing the assembly, linkedit, and recycle. This command is global and affects all active stacks.

- SOCKDBG action keyword
- The action operand can be one of these:
- DEBUGOFF/DEBUGON/SDUMPON/SDUMPOFF
- Only one action is allowed at a time, but the SOCKDBG command can be issued multiple times.
- The keyword operand can be PARTition=XX can be used but only in conjunction with DEBUGON.
- To activate debugging for all partitions issue a:
- SOCKDBG DEBUGON PART=**
- To activate debugging for one specific partition issue a:
- SOCKDBG DEBUGON PART=F2
- Replace F2 with any partition id.



New Execute Encrypted command EXENCUTE

The EXENCUTE command can be used to execute commands that were encrypted by the CIALECDC program.

- It has the following operands:
- EXENCUTE memname.memdtype KEYPHASE=CIALSEED KEYNUMBR=1 DEBUG=NO
 - memname.memdtype is the required library member name and member type that contains the encrypted commands created by the CIALECDC CATALOG utility.
 - KEYPHASE= CIALSEED is the default phase containing the key seed data to be used for generating the encryption keys.
 - KEYNUMBR=1 is the key number contained in the KEYPHASE.
- The KEYPHASE and KEYNUMBR must contain the same values used from the CIALECDC execution that created the encrypted commands member.



Bottlenecks Removed

- Unnecessary locks have been removed
- Q LOCKS to display number of locks and conflicts



Corrective Fixes

- Corrective fixes for 2.3 will be provided in new modification levels of pre-applied service packs.
- The current modification level is 2.3.4 and includes all production released zaps from prior releases.