#SHAREorg

SHARE

Problem Diagnosis for OpenSSH on z/OS

C.T. Ware
ctware@us.ibm.com
IBM Poughkeepsie, NY

February 4, 2013
Session 13024

SHARE in San Francisco 2013

Trademarks and Disclaimers

- **See http://www.ibm.com/legal/copytrade.shtml for a list of IBM trademarks.**
- **The following are trademarks or registered trademarks of other companies**
  - UNIX is a registered trademark of The Open Group in the United States and other countries
  - CERT® is a registered trademark and service mark of Carnegie Mellon University.
  - ssh® is a registered trademark of SSH Communications Security Corp
  - X Window System is a trademark of X Consortium, Inc
- **All other products may be trademarks or registered trademarks of their respective companies**

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Session Agenda

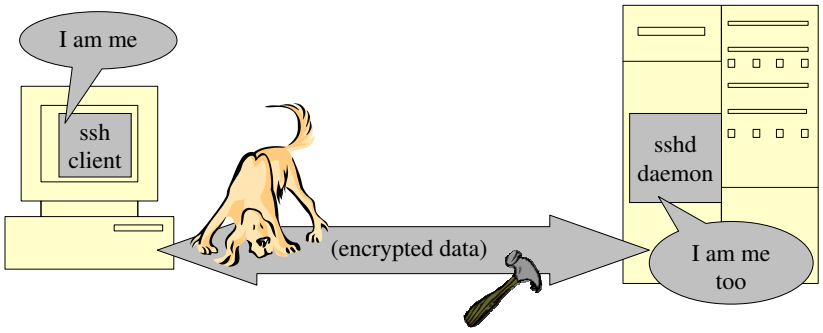
- >> OpenSSH Review <<
- Debug Facilities
- Collecting Debug Documentation
- Reading Debug Output
- Diagnosing Common Problems
- Appendix

This session is based on z/OS OpenSSH 5.0p1, unless otherwise noted [OpenSSH 1.2 HOS1120]

OpenSSH Review

- **The OpenSSH suite gives the following:**
  - Encrypted remote access, including tunneling insecure protocols.
  - Encrypted file transfer.
  - Run remote commands, programs or scripts.
  - Secure replacement for rsh, rlogin, telnet and ftp.
- **OpenSSH prevents:**
  - Eavesdropping of data transmitted over the network.
  - Manipulation of data at intermediate elements in the network (e.g. routers).
  - Address spoofing where an attack hosts pretends to be a trusted host by sending packets with the source address of the trusted host.
  - IP source routing.

OpenSSH Review



OpenSSH Review

- **OpenSSH is a very useful tool, but much of its effectiveness depends on correct use. It cannot protect from any of the following situations:**
  - Misconfiguration, misuse or abuse.
  - Compromised systems, particularly where the root account is compromised.
  - Insecure or inappropriate directory settings, particularly home directory settings.
- The z/OS OpenSSH messages use a FOTS prefix, we endeavor to keep the User's Guide as accurate as possible with explanations and solutions for each.

Functionality Review

- ssh
  - OpenSSH client (remote login program)
  - a secure alternative to rlogin, rsh, rexec
- sshd
  - OpenSSH daemon
  - daemon that listens for connections from ssh clients
  - handles key exchange, encryption, authentication, command execution, and data exchange

Functionality Review

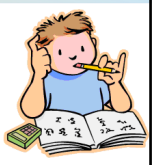
- sftp
  - Secure file transfer program
  - Similar to ftp user interface and performs all operations over encrypted ssh transport
  - Note: sftp does not use standard ftp protocols
  - Assumes files are binary. Files copied between EBCDIC and ASCII platforms are not converted by default. (See 'ascii' subcommand)
- sftp-server
  - Server side of the sftp protocol
  - sftp-server is not intended to be called directly, but from sshd using the Subsystem option.
- scp
  - Secure copy (remote file copy program)
  - Similar to rcp, but will ask for passwords/passphrases if needed and performs all operations over encrypted ssh transport
  - Assumes files are text. Files copied between EBCDIC and ASCII platforms are converted.

Session Agenda

- OpenSSH Review
- >> Debug Facilities <<
- Collecting Debug Documentation
- Reading Debug Output
- Diagnosing Common Problems
- Appendix

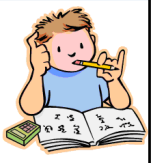
This session is based on z/OS OpenSSH 5.0p1, unless otherwise noted [OpenSSH 1.2 HOS1120]

Debug Facilities



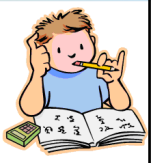
Command:	How to get debug output:
ssh	**-vvv**
scp	**-vvv**
sftp	**-vvv**
sshd	**Use USS syslogd, or in debug mode (-ddd or -De)**
ssh-keygen	**-vvv**
ssh-add	**None**
ssh-agent	**-d**
ssh-keyscan	**-v**
ssh-rand-helper	**-vvv**

Debug Facilities



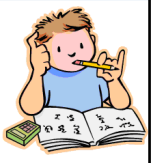
- Important note:
  - The client debug facilities may not be sufficient, for security reasons, they often will not tell the user why they were unable to establish a connection.
    - The server (when configured for debugging) typically provides more detailed information about why a particular connection attempt may have failed.
  - To collect the most amount of information, ideally you'd like to run both the client and server with the most verbose level of debugging and collect both a client and server trace from the same connection attempt.
  - You may also need to watch the MVS console for security messages from RACF (or your favorite security product).

Client Debug Facilities



- ssh
  - -v
    - Verbose mode. Causes ssh to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems. Multiple -v options increase the verbosity. You can specify up to three -v options.
  - Recommended invocation for debugging:
    - `ssh -vvv [other options] user@remote.host`
    - Debug information will be written to stderr, be prepared to capture if needed.

Client Debug Facilities

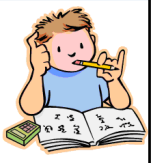


- sftp and scp
  - -v
    - Enables verbose mode. This option is also passed to ssh. Multiple -v options increase the verbosity. You can specify up to three -v options.
    - Since this is passed to ssh as well, you'll continue to receive messages regarding connection, authentication, and configuration problems, in addition to the file transfer messages.
  - Recommended invocation for debugging:
    - `sftp -vvv [other options] user@remote.host`
    - `scp -vvv [other options] user@remote.host`
    - Debug information will be written to stderr, be prepared to capture if needed.

Server Debug Facilities

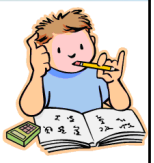
- sftp-server has a debug facility but must be enabled in the sshd_config file, and will only write to syslogd.
  - There is also the ability to cut SMF records for file transfer records
    - Specifically SMF Type 119 and subtype 96, 97, and 98 records
- Two primary methods to debug SSHD
  - Production debugging using USS syslogd
    - Preferred method – best for debugging problems encountered outside of daemon startup.
  - "Debug Mode" – special mode for sshd
    - Doesn't fork a daemon, so if problems lie in system configuration, they may not occur when running in 'debug mode'.

Server Debug Facilities



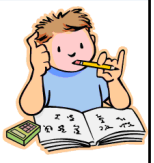
- sshd –t
  - Test mode
    - This instructs sshd to only check the validity of the sshd_config configuration file and sanity of the keys. This does not actually start sshd.
- sshd –ddd
  - Interactively debug sshd in debug mode (up to 3 for increased detail).
  - sshd will terminate after processing a single connection attempt.
- sshd –De
  - Interactively debug sshd, very similar to –ddd but:
    - sshd will remain active until terminated manually (control-c or kill <pid>).
    - Debug level based on value of sshd_config file value for LogLevel.
- sshd and syslogd
  - Best in practice, provides up to DEBUG3 level of detail while continuing to run in a production environment.
  - Can be combined with options to enable debugging in sftp-server.

Server Debug Facilities



- sshd and syslogd
  - sshd has the ability to write to the USS syslogd facility while running in production mode
    - Using these options in /etc/ssh/sshd_config:
      - *SyslogFacility*
        - *Allows you to specify which syslogd facility to write to, we recommend "DAEMON" for debugging.*
      - *LogLevel*
        - *Specify the amount of debug output to write. DEBUG3 is the most verbose and recommended value for debugging.*
    - Changes to the config file require restarting sshd.
  - Best for general debugging and/or failing connection attempts.

Server Debug Facilities



- sftp-server
  - sftp-server also has the ability to write to the USS syslogd facility:
    - In the /etc/ssh/sshd_config file, the default definition for the sftp-server is:
      - `Subsystem sftp /usr/lib/ssh/sftp-server`
    - You can place sftp-server in debug mode by specifying these options on the Subsystem entry in /etc/ssh/sshd_config:
      - *-f SyslogFacility*
        - *Allows you to specify which syslogd facility to write to, we recommend "DAEMON" for debugging.*
      - *-l LogLevel (lower case L)*
        - *Specify the amount of debug output to write. DEBUG3 is the most verbose and recommended value for debugging.*
    - Note: changes to the config file require restarting sshd.

Session Agenda

- OpenSSH Review
- Debug Facilities
- >> Collecting Debug Documentation <<
- Reading Debug Output
- Diagnosing Common Problems
- Appendix

This session is based on z/OS OpenSSH 5.0p1, unless otherwise noted [OpenSSH 1.2 HOS1120]

Collecting output from the shell

- *Typically* client debug information will be written to stderr.
  - You may need to be prepared to capture for later review or to pass along to IBM support teams.
- Shell redirection of output
  - Standard z/OS shell sample:
    - `ssh -vvv user@host 2>&1 | tee ssh-debug.log`
  - This will collect both, stderr and stdout, into a file called 'ssh-debug.log'. The '2>&1' redirects stderr to stdout, while the tee program duplicates stdout into the file (this allows capturing of data while also displaying output on the terminal).

Collecting Client Debug Output

- ssh –vvv
  - Example:
    `ssh -vvv ctware@localhost 2>&1 | tee ssh-debug.log`

```
OpenSSH_5.0p1, OpenSSL 0.9.8k 25 Mar 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug3: cipher ok: aes128-cbc [aes128-cbc,aes192-cbc,a
debug3: cipher ok: aes192-cbc [aes128-cbc,aes192-cbc,a
debug3: cipher ok: aes256-cbc [aes128-cbc,aes192-cbc,a
debug3: cipher ok: rijndael-cbc@lysator.liu.se [aes128
debug3: cipher ok: 3des-cbc [aes128-cbc,aes192-cbc,aes
debug3: cipher ok: aes128-ctr [aes128-cbc,aes192-cbc,a
debug3: cipher ok: aes192-ctr [aes128-cbc,aes192-cbc,a
debug3: cipher ok: aes256-ctr [aes128-cbc,aes192-cbc,a
debug3: cipher ok: arcfour256 [aes128-cbc,aes192-cbc,a
debug3: cipher ok: arcfour128 [aes128-cbc,aes192-cbc,a
debug3: cipher ok: blowfish-cbc [aes128-cbc,aes192-cbc
debug3: cipher ok: cast128-cbc [aes128-cbc,aes192-cbc,
debug3: cipher ok: arcfour [aes128-cbc,aes192-cbc,aes2
debug3: ciphers ok: [aes128-cbc,aes192-cbc,aes256-cbc,
...
```

Collecting Client Debug Output

- sftp –vvv
  - Example:
    `sftp -vvv ctware@localhost 2>&1 | tee ssh-debug.log`
  - Similar to ssh but also includes things like:

```
sftp> debug3: zsshSmfSetClientOpData: subcommand PUT
debug3: Looking up tmp.out
debug3: Sent message fd 6 T:17 I:2
debug3: Received stat reply T:105 I:2
debug3: Got file attribute "nlink_t_zos@us.ibm.com"
Uploading tmp.out to /u/ctware/tmp.out2
debug3: Sent message SSH2_FXP_OPEN I:3 P:/u/ctware/tmp.out2
debug3: Sent message SSH2_FXP_WRITE I:4 O:0 S:32768
debug3: SSH2_FXP_STATUS 0
debug3: In write loop, ack for 4 32768 bytes at 0
debug3: Sent message SSH2_FXP_WRITE I:5 O:32768 S:32585
debug3: SSH2_FXP_STATUS 0
debug3: In write loop, ack for 5 32585 bytes at 32768
debug3: Sent message SSH2_FXP_CLOSE I:4
debug3: SSH2_FXP_STATUS 0
debug1: zsshSmfWriteRecord: Writing SMF record type 119 subtype
sftp> debug2: channel 0: read<=0 rfd 4 len 0
```


Collecting Client Debug Output

- scp –vvv
  - Example:
    `scp -vvv tmp.out ctware@localhost:tmp.copy 2>&1 | tee ssh-debug.log`
  - Similar to ssh and sftp but also includes things like:

```
debug1: Entering interactive session.
debug2: callback start
debug2: client_session2_setup: id 0
debug1: Sending command: scp -v -t tmp.copy
debug2: channel 0: request exec confirm 0
debug2: fd 3 setting TCP_NODELAY
debug2: callback done
debug2: channel 0: open confirm rwindow 0 rmax 32768
debug2: channel 0: rcvd adjust 2097152
zsshSmfReadPipe: data length read = 198
zsshSmfReadPipe: SMF status = 84
zsshSmfTestRecord: SMF is collecting type 119, subtype 97 records
zsshSmfSetCommonData: SMF type 119 subtype 97
zsshSmfSetClientOpData: subcommand PUT
Sending file modes: C0644 65353 tmp.out
debug2: channel 0: rcvd ext data 26
Sink: C0644 65353 tmp.out
```

Collecting Server Debug Output

- sshd –De -or- sshd –ddd
  - Example:
    ```
    /usr/sbin/sshd -De 2>&1 | tee server.log
    or
    /usr/sbin/sshd -ddd  2>&1 | tee server.log
    ```

```
debug2: load_server_config: filename /etc/ssh/sshd_config
debug2: load_server_config: done config len = 244
debug2: parse_server_config: config /etc/ssh/sshd_config len 244
debug3: /etc/ssh/sshd_config:28 setting Protocol 2
debug3: /etc/ssh/sshd_config:122 setting Subsystem sftp /usr/lib/ssh/sf
debug3: /etc/ssh/sshd_config:129 setting SyslogFacility DAEMON
debug3: /etc/ssh/sshd_config:130 setting LogLevel DEBUG3
debug3: /etc/ssh/sshd_config:131 setting StrictModes no
debug3: /etc/ssh/sshd_config:132 setting Protocol 2,1
debug2: load_server_config: filename /etc/ssh/zos_sshd_config
debug2: load_server_config: done config len = 30
debug2: parse_server_config: config /etc/ssh/zos_sshd_config len 30
debug3: RNG is ready, skipping seeding
   ...
```


Collecting Server Debug Output

- sshd and syslogd
  - Ensure USS syslog daemon is available.
  - Add a configuration statement to /etc/syslog.conf
    - For our purposes we're going to utilize the 'daemon' syslog facility and the 'debug' priority code
    - `daemon.debug  /tmp/syslogd/server.logfile`
      - *ensure the directory /tmp/syslogd/ exists*
  - Update the /etc/ssh/sshd_config file to reflect our desire to run in a debug mode
    - Update/define the option "SyslogFacility" to be "DAEMON"
    - Update/define the option "LogLevel" to be "DEBUG3"
      - ```
        SyslogFacility DAEMON
        LogLevel DEBUG3
        ```
  - After updating configuration settings for the daemons, they require restarting for the changes to be picked up.
        ```

Collecting Server Debug Output

- sshd and syslogd
  - After restarting sshd and syslogd, sshd will be writing debug3 data to /tmp/syslogd/server.logfile while running a production server.
  - syslogd has the added advantage of timestamping all entries, and will show the creation of child address spaces.

```
sshd[54]: Server listening on 0.0.0.0 port 22.
sshd[54]: debug1: fd 4 clearing O_NONBLOCK
sshd[54]: debug1: Forked child 33554485.
sshd[54]: debug3: send_rexec_state: entering fd = 8 config len
sshd[54]: debug3: ssh_msg_send: type 0
sshd[54]: debug3: send_rexec_state: done
sshd[33554485]: debug1: rexec start in 4 out 4 newsock 4 pipe
sshd[33554485]: debug1: inetd sockets after dupping: 3, 3
sshd[33554485]: debug1: cipher_init: none from source OpenSSL
```


Collecting Server Debug Output

- sftp-server and syslogd
  - Ensure USS syslog daemon is available and configured for collecting sshd debug data as described previously.
  - Update the sftp subsystem configuration statement in /etc/ssh/sshd_config to reflect our desire to run in a debug mode
    - Use –f to define the value for “SyslogFacility” to be “DAEMON”
    - Use –l (lower L) to define the option “LogLevel” to be “DEBUG3”

    ```
    Subsystem sftp /usr/lib/ssh/sftp-server -f DAEMON -l DEBUG3
    ```

  - After updating configuration settings for the daemons, they require restarting for the changes to be picked up.
    - After restarting sshd and syslogd, sftp-server will be writing debug3 data to /tmp/syslogd/server.logfile while running a production server.

Collecting Server Debug Output

- Sample of the sftp-server debug output written to the syslog:

```
sftp-server[52]: session opened for local user ctware from  9.x.x.17
sftp-server[52]: received client version 3
sftp-server[52]: debug3: request 1: realpath
sftp-server[52]: realpath "."
sftp-server[52]: debug1: request 1: sent names count 1
sftp-server[52]: debug3: request 2: stat
sftp-server[52]: stat name "/tmp.copy"
sftp-server[52]: debug3: request 2: sent status 2
sftp-server[52]: sent status No such file
sftp-server[52]: debug3: request 3: open flags 26
sftp-server[52]: debug3: Got file attribute "nlink_t_zos@us.ibm.com"
sftp-server[52]: open "/tmp.copy" flags WRITE,CREATE,TRUNCATE mode 0644
sftp-server[52]: debug1: request 3: sent handle handle 0
sftp-server[52]: debug1: request 4: write "/tmp.copy" (handle 0) off 0 l
sftp-server[52]: debug3: request 4: sent status 0
sftp-server[52]: sent status Success
sftp-server[52]: debug1: request 5: write "/tmp.copy" (handle 0) off 327
sftp-server[52]: debug3: request 5: sent status 0
sftp-server[52]: sent status Success
sftp-server[52]: debug3: request 4: close handle 0
sftp-server[52]: close "/tmp.copy" bytes read 0 written 65353
sftp-server[52]: debug3: request 4: sent status 0
sftp-server[52]: sent status Success
```


Session Agenda

- OpenSSH Review
- Debug Facilities
- Collecting Debug Documentation
- >> Reading Debug Output <<
- Diagnosing Common Problems
- Appendix

This session is based on z/OS OpenSSH 5.0p1, unless otherwise noted [OpenSSH 1.2 HOS1120]

Reading Debug Output

- Recall:
  - For best effectiveness you will likely want to collect a debug trace from both the client and server for the same connection attempt.
  - Since the messages may be intimidating or confusing, remember: when in doubt, Google is your friend.
    - Note: OpenSSH runs on many platforms, keep this in mind when using searching the internet - not everything you discover will apply to the z/OS implementation.
- A great place to begin (if possible) is to also collect a set of traces from a working connection and compare with the failing results.
- Rather than reviewing one of many potential failing scenarios, let's take a look through a working one to build familiarity and comfort…

Client Debug Output – A Walkthrough

- A brief overview of the general flow of an sftp connection (note this will include ssh debug output and scp will be similar):
  - We'll review the output of:
    `sftp -vvv ctware@9.x.x.13`
  - Which is going to do:
    `put tmp.out tmp.copy`
    - Note: Some of the messages will be truncated or omitted for the sake of readability.

Client Debug Output – A Walkthrough

IP Name or Address of Remote System

```
Connecting to 9.x.x.13...
debug3: zsshSmfCreatePipe: SMF pipe created fdout=7, fdin=8
debug1: Reading configuration data /etc/ssh/zos_ssh_config
OpenSSH_5.0p1, OpenSSL 0.9.8k 25 Mar 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug3: cipher ok: aes128-cbc [aes128-cbc,aes192-cbc,aes256
…
debug2: mac_setup: found hmac-sha1
debug3: mac ok: hmac-sha1 [hmac-sha1,
…
debug3: RNG is ready, skipping seeding
…
```

RNG is ready since /dev/random exists

Client Debug Output – A Walkthrough

ICSF is installed and available

```
debug1: zsshVerifyIcsfSetup: ICSF FMID is 'HCR77A0 '
debug1: zsshVerifyIcsfSetup (163): CSFIQA successful: return
debug2: ------------------------------------
debug2: CRYPTO    SIZE      KEY       SOURCE
debug2: ------------------------------------
debug2: AES       256       SECURE    COP
…
debug1: Connecting to 9.x.x.13 [9.x.x.13] port 22.
debug1: Connection established.
…
debug3: Not a RSA1 key file /u/ctware/.ssh/id_rsa.
debug2: key_type_from_name: unknown key type '-----BEGIN'
debug3: key_read: missing keytype
debug3: key_read: missing whitespace
…
```

List of available Crypto methods and source

Reading private key(s)

Client Debug Output – A Walkthrough

Local and remote versions of OpenSSH

```
debug1: Remote protocol version 2.0, remote software version
  OpenSSH_5.0
debug1: match: OpenSSH_5.0 pat OpenSSH*
…
debug1: Local version string SSH-2.0-OpenSSH_5.0
…
debug3: RNG is ready, skipping seeding
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug2: kex_parse_kexinit: diffie-hellman-
…
debug1: mac_setup_by_id: hmac-sha1 from source ICSF
debug2: mac_setup: found hmac-sha1
debug1: kex: server->client aes128-cbc hmac-sha1 none
debug1: mac_setup_by_id: hmac-sha1 from source ICSF
debug2: mac_setup: found hmac-sha1
debug1: kex: client->server aes128-cbc hmac-sha1 none
…
```

Beginning key exchange

Using ICSF for message authentication code

Client Debug Output – A Walkthrough

```
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug3: check_host_in_hostfile: filename /u/ctware/.ssh/known_hosts
debug3: check_host_in_hostfile: filename /etc/ssh/ssh_known_hosts
…
debug2: key: /u/ctware/.ssh/id_rsa (0)
debug2: key: /u/ctware/.ssh/id_dsa (0)
debug1: Authentications that can continue: publickey,password
…
debug3: authmethod_is_enabled publickey
debug1: Next authentication method: publickey
debug1: Trying private key: /u/ctware/.ssh/id_rsa
debug1: read PEM private key done: type RSA
debug3: sign_and_send_pubkey
debug2: we sent a publickey packet, wait for reply
debug1: Authentications that can continue: publickey,password
debug1: Trying private key: /u/ctware/.ssh/id_dsa
debug1: read PEM private key done: type DSA
```

Host key authentication occurring

User authentication

Client Debug Output – A Walkthrough

```
debug3: sign_and_send_pubkey
debug2: we sent a publickey packet, wait for reply
debug1: Authentications that can continue: publickey,password
debug2: we did not send a packet, disable method
debug3: authmethod_lookup password
debug3: remaining preferred: ,password
debug3: authmethod_is_enabled password
debug1: Next authentication method: password
debug2: we sent a password packet, wait for reply
debug1: Authentication succeeded (password).
…
debug1: Entering interactive session.
debug2: callback start
debug2: client_session2_setup: id 0
debug1: Sending subsystem: sftp
…
```

User authentication occurring

Pubkey failed, trying password

Password Authentication Successful

Requesting sftp

Client Debug Output – A Walkthrough

```
debug3: SSH_FXP_REALPATH . -> /
…
sftp> debug3: zsshSmfSetClientOpData: subcommand PUT
debug3: Looking up tmp.out
…
Uploading tmp.out to /tmp.copy
debug3: Sent message SSH2_FXP_OPEN I:3 P:/tmp.copy
debug3: Sent message SSH2_FXP_WRITE I:4 O:0 S:32768
debug3: SSH2_FXP_STATUS 0
debug3: In write loop, ack for 4 32768 bytes at 0
debug3: Sent message SSH2_FXP_WRITE I:5 O:32768 S:32585
debug3: SSH2_FXP_STATUS 0
debug3: In write loop, ack for 5 32585 bytes at 32768
debug3: Sent message SSH2_FXP_CLOSE I:4
debug3: SSH2_FXP_STATUS 0
debug1: zsshSmfWriteRecord: Writing SMF record type 119 subtype 97
  length 332 with record exit IEFU84.
```

Starting remote directory

Performing a 'put'

File to put

Destination file

Open, Write, Close

32k chunks

Client Debug Output – A Walkthrough

```
sftp> debug2: channel 0: read<=0 rfd 5 len 0
…
debug2: channel 0: close_read
…
debug2: channel 0: ibuf empty
debug2: channel 0: send eof
debug2: channel 0: input drain -> closed
debug2: channel 0: rcvd eof
debug2: channel 0: rcvd close
debug3: channel 0: will not send data after close
debug2: channel 0: almost dead
debug2: channel 0: gc: notify user
debug2: channel 0: gc: user detached
debug2: channel 0: send close
debug2: channel 0: is dead
debug2: channel 0: garbage collecting
debug1: channel 0: free: client-session, nchannels 1
…
debug1: Exit status 0
```

Disconnect flow

Exit Success

Server Debug Output – A Walkthrough

- A brief overview of the general flow of an sftp connection from the server side
  - This is the server trace which corresponds to the sftp client trace we just reviewed.
    - `sftp -vvv ctware@9.x.x.13`
    - Which is going to do:
      `put tmp.out tmp.copy`
  - Note: Some of the messages will be truncated or omitted for the sake of readability and the timestamp prefix information has been removed.
  - Note: sftp-server doesn't have additional debug messages.

Server Debug Output – A Walkthrough

```
sshd[33554489]: debug1: Bind to port 22 on 0.0.0.0.
sshd[33554489]: Server listening on 0.0.0.0 port 22.
…
sshd[33554489]: debug1: Forked child 33554486.
…
sshd[33554486]: Connection from 9.x.x.17 port 43820
…
sshd[33554486]: debug1: Client protocol version 2.0; client
   software version OpenSSH_5.0
sshd[33554486]: debug1: match: OpenSSH_5.0 pat OpenSSH*
sshd[33554486]: debug1: Local version string SSH-2.0-OpenSSH_5.0
sshd[33554486]: debug2: Network child is on pid 83886136
sshd[33554486]: debug3: Current IBM Release level: 23
sshd[33554486]: debug3: MLS: seclabel of AS:          uid:0
…
sshd[33554486]: debug3: mm_answer_pwnamallow
sshd[33554486]: debug3: get_pwnamallow: input user name ctware,
sshd[33554486]: debug3: Trying to reverse map address 9.x.x.17.
…
```

Server is up

Incoming connection

Version info.

Privsep child

z/OS release 1.13

Incoming username

Server Debug Output – A Walkthrough

```
sshd[33554486]: debug3: mm_answer_keyallowed entering
sshd[33554486]: debug3: mm_answer_keyallowed: key_from_blob:
   25DE6120
sshd[33554486]: debug1: temporarily_use_uid: 0/512 (e=0/512)
sshd[33554486]: debug1: trying public key file
   /.ssh/authorized_keys
sshd[33554486]: debug3: key_read: type mismatch
sshd[33554486]: debug2: user_key_allowed: check options: 'ssh-dss
   AAAAB3NzaC…'
sshd[33554486]: debug1: restore_uid: 0/512
sshd[33554486]: debug2: key not found
…
sshd[33554486]: debug3: auth_log: authenticated 0, valid 1,
   failures 0, max 6, half 3, method publickey
sshd[33554486]: Failed publickey for ctware from 9.x.x.17 port
   43820 ssh2
sshd[33554486]: debug3: mm_answer_keyallowed: key 25DE6120 is
   disallowed
…
```

Processing user keys

User DSA public key

Not found

Server Debug Output – A Walkthrough

```
sshd[33554486]: debug3: mm_answer_keyallowed: key_from_blob:
  25DE5078
sshd[33554486]: debug1: temporarily_use_uid: 0/512 (e=0/512)
sshd[33554486]: debug1: trying public key file
  /.ssh/authorized_keys
sshd[33554486]: debug3: key_read: type mismatch
sshd[33554486]: debug2: user_key_allowed: check options: 'ssh-rsa
  AAAAB3…'
sshd[33554486]: debug1: restore_uid: 0/512
sshd[33554486]: debug2: key not found
…
sshd[33554486]: debug3: auth_log: authenticated 0, valid 1,
  failures 1, max 6, half 3, method publickey
sshd[33554486]: Failed publickey for ctware from 9.x.x.17 port
  43820 ssh2
sshd[33554486]: debug3: mm_answer_keyallowed: key 25DE5078 is
  disallowed
…
```

Processing user keys

User **RSA** public key

Not found

Server Debug Output – A Walkthrough

Now trying password

```
sshd[33554486]: debug3: monitor_read: checking request 10
sshd[33554486]: debug3: mm_answer_authpassword: sending result 1
sshd[33554486]: debug3: mm_request_send entering: type 11
sshd[33554486]: debug3: auth_log: authenticated 1, valid 1,
  failures 2, max 6, half 3, method password
sshd[33554486]: Accepted password for ctware from 9.x.x.17 port
  43820 ssh2
sshd[33554486]: debug1: monitor_child_preauth: ctware has been
  authenticated by privileged process
…
```

Successful login

Server Debug Output – A Walkthrough

Message Authentication Code from software

```
sshd[33554486]: debug1: mac_setup_by_id: hmac-sha1 from source
  OpenSSL
sshd[33554486]: debug2: mac_setup: found hmac-sha1
sshd[33554486]: debug3: mm_get_keystate: Waiting for second key
sshd[33554486]: debug3: mm_newkeys_from_blob: 25DE5180(123)
sshd[33554486]: debug1: mac_setup_by_id: hmac-sha1 from source
  OpenSSL
sshd[33554486]: debug2: mac_setup: found hmac-sha1
sshd[33554486]: debug3: mm_get_keystate: Getting compression state
sshd[33554486]: debug3: mm_get_keystate: Getting Network I/O
  buffers
sshd[33554486]: debug2: set_newkeys: mode 0
sshd[33554486]: debug1: cipher_init: aes128-cbc from source OpenSSL
sshd[33554486]: debug2: set_newkeys: mode 1
sshd[33554486]: debug1: cipher_init: aes128-cbc from source OpenSSL
sshd[33554486]: debug1: Entering interactive session for SSH2.
…
```

Ciphers from software (ICSF unavailable on this system)

Server Debug Output – A Walkthrough

Starting an sftp session

```
sshd[33554486]: subsystem request for sftp
sshd[33554486]: debug1: subsystem: exec() /usr/lib/ssh/sftp-server
sshd[33554486]: debug3: do_exec: subsystem 1
sshd[33554486]: debug3: do_exec: passwd name=ctware, uid=0,
  gid=512, dir=/, shell=/bin/sh
…
```

sftp environment

Server Debug Output – A Walkthrough

```
sftp-server[52]: session opened for local user ctware from  9.x.x.17
…
sftp-server[52]: stat name "/tmp.copy"
sftp-server[52]: debug3: request 2: sent status 2
sftp-server[52]: sent status No such file
…
sftp-server[52]: open "/tmp.copy" flags WRITE,CREATE,TRUNCATE mode 0644
sftp-server[52]: debug1: request 3: sent handle handle 0
sftp-server[52]: debug1: request 4: write "/tmp.copy" (handle 0) off 0 l
sftp-server[52]: debug3: request 4: sent status 0
sftp-server[52]: sent status Success
sftp-server[52]: debug1: request 5: write "/tmp.copy" (handle 0) off 327
sftp-server[52]: debug3: request 5: sent status 0
sftp-server[52]: sent status Success
sftp-server[52]: debug3: request 4: close handle 0
sftp-server[52]: close "/tmp.copy" bytes read 0 written 65353
sftp-server[52]: debug3: request 4: sent status 0
sftp-server[52]: sent status Success
…
```

stat() destination file

Open destination file

write destination file

close destination file

Successful write

Server Debug Output – A Walkthrough

```
sshd[33554486]: debug2: channel 0: rcvd eof
sshd[33554486]: debug2: channel 0: output open -> drain
sshd[33554486]: debug2: channel 0: obuf empty
sshd[33554486]: debug2: channel 0: close_write
sshd[33554486]: debug2: channel 0: output drain -> closed
…
sshd[33554486]: debug2: channel 0: send close
sshd[33554486]: debug3: channel 0: will not send data after close
sshd[33554486]: debug2: channel 0: rcvd close
sshd[33554486]: debug3: channel 0: will not send data after close
…
sshd[33554486]: debug2: channel 0: gc: user detached
sshd[33554486]: debug2: channel 0: garbage collecting
sshd[33554486]: Connection closed by 9.x.x.17
sshd[33554486]: debug1: do_cleanup
sshd[33554486]: debug3: zsshCloseOldDev: fd=-1
sshd[33554486]: Closing connection to 9.x.x.17
```

Disconnect flow

Closed by client (using "exit" sftp subcmd)

Server cleanup (and termination of this process)

Reading Debug Output

- Where to go from here, Messages, and Guidance
  - After walking through a successful client/server flow as we've done, when a curveball (i.e. failure) is thrown your way, you hopefully will have better understanding of what part of the connection is in question.
  - If the failure is coupled with a FOTS* message, please refer to the z/OS OpenSSH User Guide for explanations.
  - Note: OpenSSH is a complex product requiring the use of many other components (LE, CRTL, USS, TCP/IP, HFS, etc).
    - Watch for other messages in the job output and MVS console.

Session Agenda

- OpenSSH Review
- Debug Facilities
- Collecting Debug Documentation
- Reading Debug Output
- >> Diagnosing Common Problems <<
- Appendix

This session is based on z/OS OpenSSH 5.0p1, unless otherwise noted [OpenSSH 1.2 HOS1120]

Diagnosing Common Problems

- A few common issues often encountered:
  - Confusion between Host Keys and User Keys
  - StrictModes preventing connections
  - Performance issues with ssh-rand-helper

Key Confusion

- Public/Private Keypairs
  - Generated using ssh-keygen.
    - Public Key
      - *A public key is one of two keys used in public-key encryption (the other being a private key). The user releases a copy of this key to the public to allow anyone to use it for encrypting messages to be sent to the user and for decrypting messages received from the user.*
    - Private Key
      - *The user keeps the private key secret and uses it to encrypt outgoing messages and decrypt incoming messages.*
      - *The permissions for the private key should be set so that only the owner has read/write access.*

Key Confusion

- Types of authentication
  - Server authentication
    - Occurs early on during an ssh connection (prior to user authentication).
    - This prevents systems from pretending to be another system (spoofing), and assures the remote system is the desired system.
    - Authenticating the server has to be done before you send any confidential data to it. In particular, if the user authentication involves a password, the password must not be sent to an unauthenticated server.
    - The remote server provides a Public Key referred to as a Host Key (the server keeps the private key secret). This can be retrieved using ssh-keyscan or stored during an client connection.

Key Confusion

- Server authentication
  - Host Key
    - *Public portion of a public/private key pair.*
    - *This is stored in the client user's ~/.ssh/known_hosts or client system's /etc/ssh/ssh_known_hosts file.*
    - *When connecting with the corresponding server, the client offers this key and the server will match it with it's private portion of the key.*
    - *If the server changes its keypair and the stored host key no longer matches, the client will be notified (batch jobs will likely fail if this case occurs).*

Key Confusion

- Types of authentication
  - User authentication
    - Several methods including password and public key
    - Public Key authentication
      - *This is exactly the same method that is used to authenticate the server, but now the user is trying to prove its identity and the server is verifying it.*
      - *The login attempt is accepted if the user proves that he has the private key and the public key is in the remote system user's authorization list (~/.ssh/authorized_keys on the server).*
    - Typically the keypair is generated on the client system and the public portion of the key is copied (either using sftp and a password, or FTP, or possibly even copy/paste, etc) to the remote system.
      - *It is then stored in the user's authorized_keys file on the remote system.*

StrictModes Issues

- When the server has "StrictModes yes" set, user pubkey authentication may fail
  - The client will not receive any indication why the connection failed (even in debug mode).
  - The server debug trace will indicate the public key was rejected:

```
debug1: temporarily_use_uid: 0/512 (e=0/512)
debug1: trying public key file /userhome/.ssh/authorized_keys
debug1: restore_uid: 0/512
debug3: auth_log: authenticated 0, valid 1, failures 1, max 6, half 3, method publickey
Failed publickey for ctware from 9.00.00.17 port 30582 ssh2
debug3: mm_answer_keyallowed: key 25DE4078 is disallowed
```

StrictModes Issues

- If trying to establish an interactive session, this would still fall through and try to authenticate using password.
- If you're certain the pubkey pair was created and configured properly, verify the server is using StrictModes.
  - An easy test to determine if the connection is failing because of StrictModes, would be to change the server setting to 'no'
    - *Might not be feasible on a production system.*

StrictModes Issues

- If you determine the connection is failing because of StrictModes, SSHD may believe access to your files/directories may be too insecure.
  - Here are some of the files/directories SSHD tests and permissions tested and verified to work:

File	Permissions
User's home directory	755
User's authorized_keys file	644
User's known_hosts file	600
User's private key files	600
User's .ssh directory	700

Performance Issues

- The open source method to generate random numbers (ssh-rand-helper utility) uses various shell commands (listed in /etc/ssh/ssh_prng_cmds) and collects pieces of the output to be combined together to create a random number.
  - On z/OS the execution of this utility may not be as responsive as other platforms.
    - You may encounter SEC6/FF02 abends or see SIGINTs during client execution.
  - Solutions:
    - Will use /dev/random automatically if ICSF hardware is in place.
    - Check if CEE.SCEELPA is in the LPA list
    - Note: APAR OA37278 includes support to use ICSF hardware for additional purposes beyond RNG.

Session Agenda

- OpenSSH Review
- Debug Facilities
- Collecting Debug Documentation
- Reading Debug Output
- Diagnosing Common Problems
- >> Appendix <<

This session is based on z/OS OpenSSH 5.0p1, unless otherwise noted [OpenSSH 1.2 HOS1120]

Appendix

- **See the updated "IBM Ported Tools for z/OS: OpenSSH User's Guide" for more information**
  (Order Number: SA23-2246-01)
- **Website References**
  - IBM Ported Tools for z/OS:
    http://www.ibm.com/systems/z/os/zos/features/unix/ported/
  - IBM Ported Tools for z/OS: OpenSSH:
    http://www.ibm.com/systems/z/os/zos/features/unix/ported/openssh/
  - OpenSSH: http://www.openssh.org/
  - OpenSSL: http://www.openssl.org/

Appendix

- **ICSF Reference Guides:**
  - z/OS Cryptographic Services ICSF Overview
    (Order Number: SA22-7519-13)
  - z/OS Cryptographic Services ICSF Administrator's Guide
    (Order Number: SA22-7521-14)
  - z/OS Cryptographic Services ICSF System Programmer's Guide
    (Order Number: SA22-7520-14)
  - z/OS Cryptographic Services ICSF Application Programmer's Guide
    (Order Number: SA22-7522-13)
  - z/OS Cryptographic Services Writing PKCS #11 Applications
    (Order Number: SA23-2231-02)
- **Other Reference Guides:**
  - Program Directory for IBM Ported Tools for z/OS
    (Order Number: GI10-0769-06)

System z Social Media

- System z official Twitter handle:
  - @ibm_system_z
- Top Facebook pages related to System z:
  - Systemz Mainframe
  - IBM System z on Campus
  - IBM Mainframe Professionals
  - Millennial Mainframer
- Top LinkedIn Groups related to System z:
  - Mainframe Experts Network
  - Mainframe
  - IBM Mainframe
  - System z Advocates
  - Cloud Mainframe Computing
- YouTube
  - IBM System z

- Leading Blogs related to System z:
  - Evangelizing Mainframe (Destination z blog)
  - Mainframe Performance Topics
  - Common Sense
  - Enterprise Class Innovation: System z perspectives
  - Mainframe
  - MainframeZone
  - Smarter Computing Blog
  - Millennial Mainframer