Session H5
Securing Your z/OS UNIX Network Access with OpenSSH
July 10, 2006

Erin Farr
z/OS UNIX System Services Development
IBM Poughkeepsie, NY
efarr@us.ibm.com
http://www-1.ibm.com/servers/eserver/zseries/zos/unix/port_tools.html

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- z/OS®

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.
- Exceed, Exceed Web, Hummingbird CAP, and Hummingbird EIP are trademark of Hummingbird Communications Ltd.
- 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.

Table of Contents

Overview

- **Problem:**

  Unencrypted network data (including passwords)

  - Numerous customer requirements

- **Solution:**

  OpenSSH – suite of network connectivity tools that provide secure encrypted communications between two untrusted hosts over an insecure network.

  Program Product: IBM Ported Tools for z/OS

  - unpriced, runs on z/OS 1.4 and higher
  - order from ShopzSeries, under "MVS: System Mgmt. and Security."
  - GA Version info: OpenSSH 3.5p1, OpenSSL 0.9.7b, zlib 1.1.4
  - OA10315 version is: OpenSSH 3.8.1p1, OpenSSL 0.9.7d, zlib 1.1.4

What is OpenSSH – Openssh provides:

- Authentication (both client and server) through:
  - Public key cryptography
  - Existing login passwords
  - Trusted hosts authentication
- Data Privacy - through encryption
- Data Integrity - guarantees data traveling over the network is unaltered
- Authorization – regulates access control to accounts
- Forwarding (a.k.a. tunneling) – encryption of other TCP/IP-based sessions

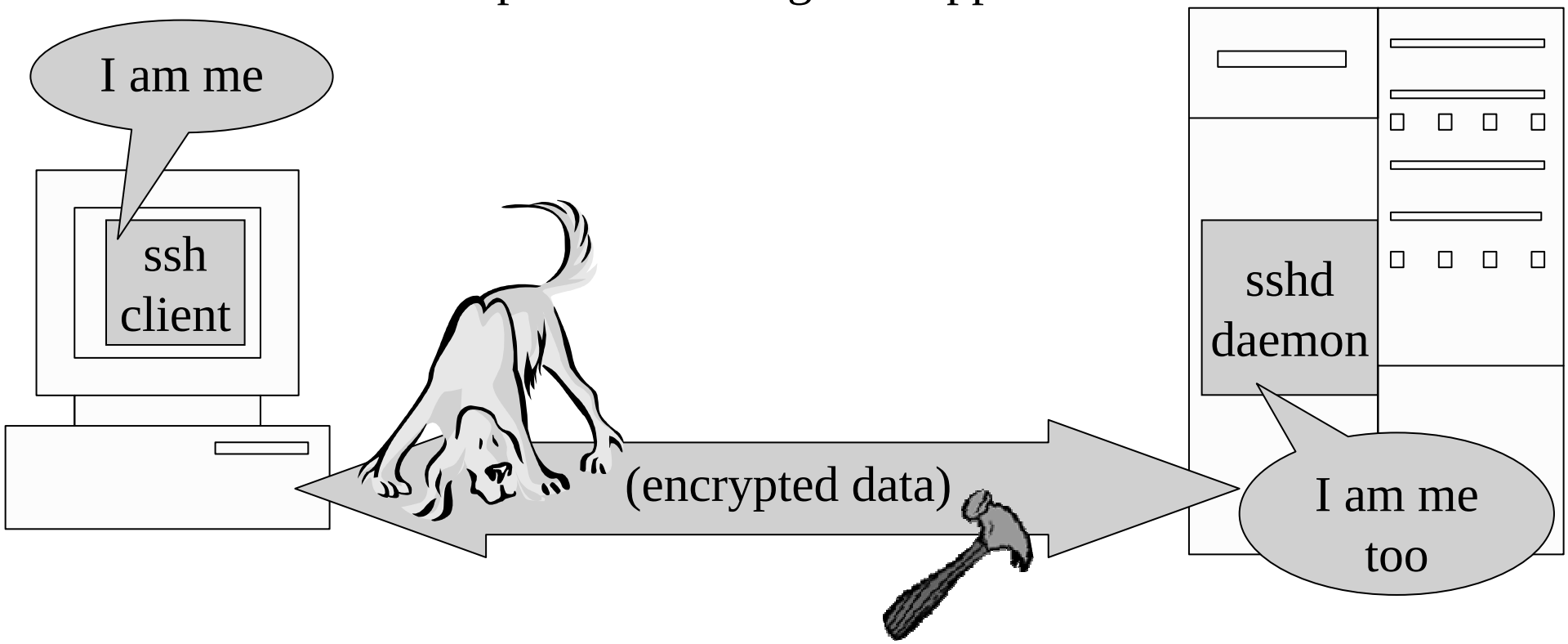
What is OpenSSH – Openssh provides:

Function	OpenSSH Utility	An alternative to…
Secure remote login	ssh, sshd	rlogin, rsh
Secure file transfer	sftp, sftp-server, scp	rcp

OpenSSH additionally provides these utilities:	
Key management	ssh-keygen, ssh-agent, ssh-add, ssh-keyscan

What is OpenSSH – an example

- Provides *secure*:
  - remote login
  - remote command execution
  - X11 and TCP/IP port forwarding
- ssh is not a shell in the "UNIX" sense.
- SSH protocols 1 and 2 supported



Why OpenSSH – Threats OpenSSH can counter

- Eavesdropping
- Name service and IP spoofing
- Connection Hijacking
- Man-in-the-Middle Attacks
- Insertion Attacks

Why OpenSSH – Threats OpenSSH doesn't prevent

- Password cracking
- IP and TCP attacks
- Traffic analysis
- Covert Channels
- Carelessness

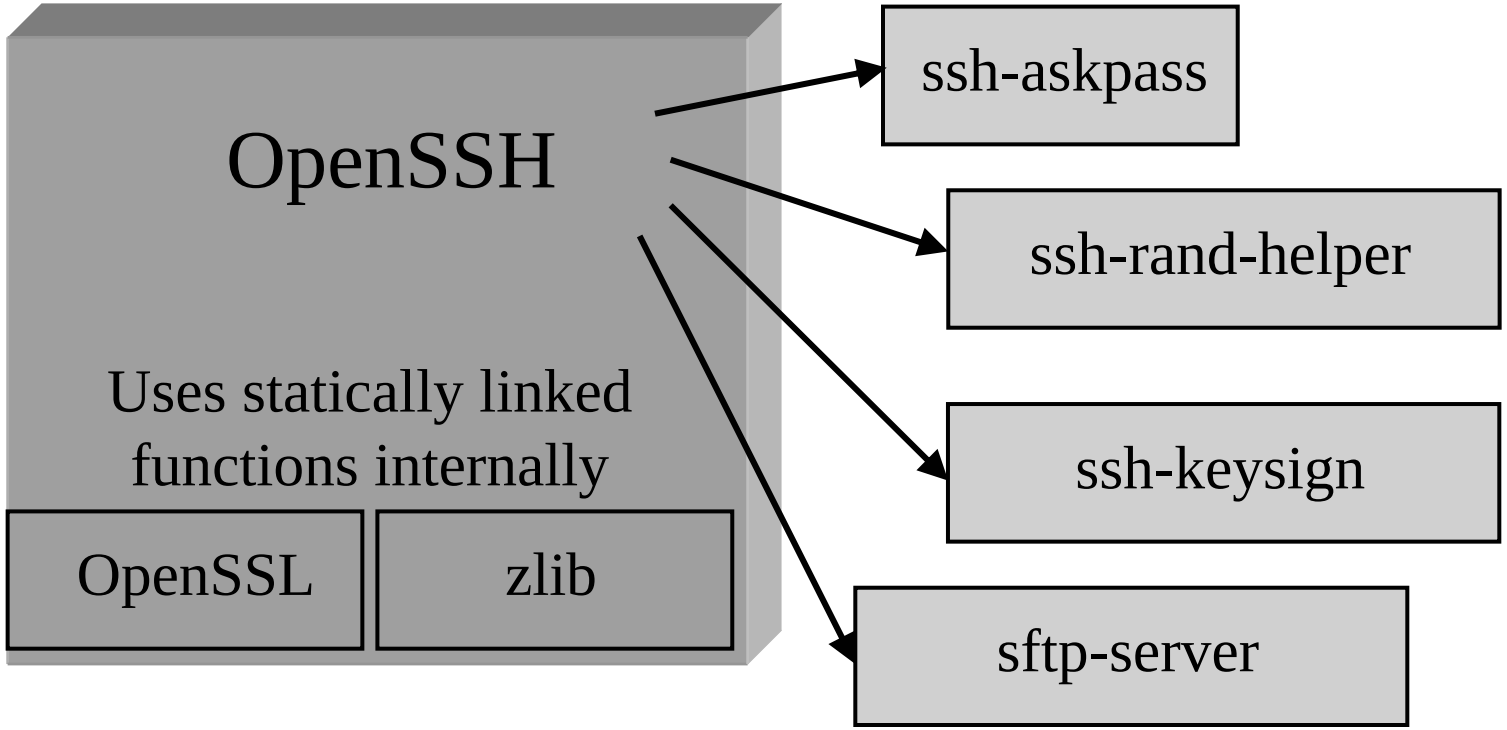
Why OpenSSH vs. other solutions?

- Why not secure FTP using System SSL?
  - While secure FTP is often the proper solution, it **cannot communicate with other platforms running OpenSSH** as their secure file transfer mechanism.
- Why not another implementation of SSH, like a commercial version?
  - OpenSSH is the **most common implementation** of Secure Shell on UNIX platforms (88% in 9/2004).

OpenSSH – functional content

- Version info: OpenSSH 3.8.1p1, OpenSSL 0.9.7d, zlib 1.1.4
- OpenSSH provides the command interface for :

  ssh, sshd, scp, sftp, ssh-agent, ssh-add, ssh-keygen, ssh-keyscan



OpenSSH – functional content – ssh and sshd

- ssh – secure remote login program
  - a secure alternative to rlogin, rsh, rexec

- sshd – secure remote login daemon
  - daemon that listens for connections from ssh clients
  - handles key exchange, encryption, authentication, command execution, and data exchange

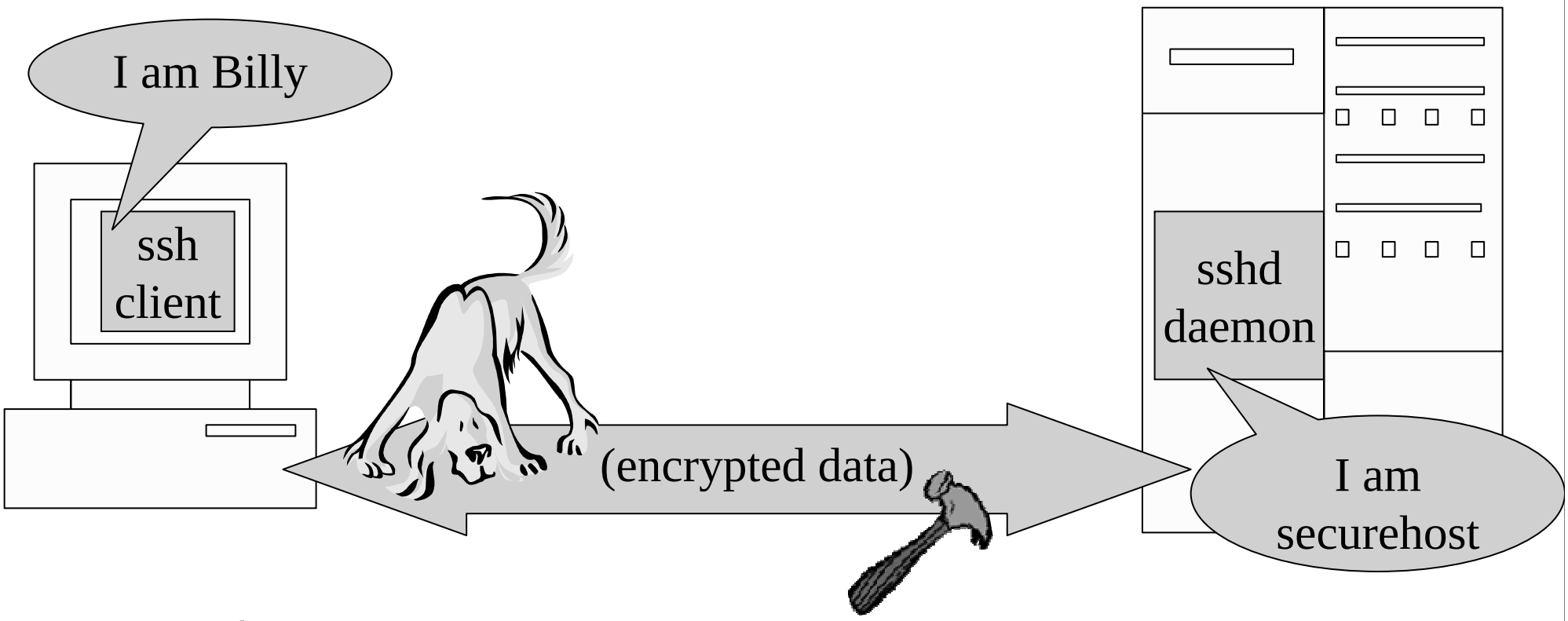
- Once an SSH session is established, other connections can be forwarded over this secure channel:
  - X11, TCP/IP, Authentication agents

- Together, ssh and sshd provide secure encrypted communications between two untrusted hosts over an insecure network.

OpenSSH – usage and invocation

S SSHD
or
_BPX_JOBNAME=SSHD /usr/sbin/sshd &

ssh Billy@securehost



OpenSSH – starting sshd

- Standalone daemon
  - Through a cataloged procedure
    - For customers who use BPX.DAEMON
  - Through /etc/rc
    - Starts automatically, but hard to restart without procedure
  - Through the UNIX shell
    - Security level not generally adequate for z/OS systems
      - User with READ access to BPX.DAEMON should not be able to log into the UNIX shell.
- Through inetd
  - Could decrease performance of ssh connection startup time

OpenSSH – Invocation through BPXBATCH

- Create a cataloged procedure using PARM=PGM to invoke a shell script

```
//SSHD    PROC
//SSHD    EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//        PARM='PGM /bin/sh -c /etc/ssh/sshd.sh'
//* STDIN and STDOUT are both defaulted to /dev/null
//STDERR  DD PATH='/tmp/sshd.stderr',PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//        PATHMODE=(SIRWXU)
```

- Useful if you have a long sshd command line
- Avoids output in /etc/profile issue, by doing the sh command with -c (does not do a login shell)
- The shell script can have logic to get around the SIGHUP issue.

```
#!/bin/sh
nohup /usr/sbin/sshd -f /etc/ssh/sshd_config &
sleep 1
```

OpenSSH and BPX.DAEMON

- sshd must be started with a user who is uid 0.
- If you are running with BPX.DAEMON, the user ID invoking sshd needs READ access to BPX.DAEMON
  - sshd needs to setuid to the (unprivileged) privilege separation user
    - unauthenticated identity change
  - Public Key authentication – no SAF call
    - “unauthenticated” identity change from standpoint of the security product
    - sshd does authentication itself

OpenSSH – secure file transfer

- sftp (secure file transfer program)
  - An interactive file transfer program, similar to the ftp user interface
  - Performs all operations over an encrypted ssh transport
    - May also use many features of ssh
  - Does not directly support MVS datasets.
- sftp-server (SFTP server subsystem)
  - Server-side of the SFTP protocol
  - Invoked from sshd

NOTE: SFTP does not talk FTP protocols

- scp – secure copy (remote file copy program)
  - Also uses ssh for data transfer
  - Similar to rcp (command syntax)
  - Unlike rcp, scp asks for passwords/passphrases if necessary

sftp (OpenSSH)	**Secure FTP (Communications Server)**
sftp (aka “SFTP”)	**Secure FTP** (aka “FTP with TLS”, “FTPS”)
IETF Secure Shell (secsh) Working Group Internet drafts	RFC 959 (FTP) RFC 2228 (FTP security extensions)
Uses OpenSSL (statically linked library)	Uses SystemSSL (hardware crypto)
Ported code – not integrated with security product	Integrated with security product
MVS dataset support not integrated	Built-in MVS dataset support

OpenSSH – Key management

•OpenSSH's key generation and management is separate from other key management provided by IBM.

- ssh-keygen
  - •creates public/private key pairs
- ssh-agent
  - •holds private keys in memory, saving you from retyping your passphrase repeatedly
- ssh-add
  - •loads private keys into the agent
- ssh-keyscan
  - •gathers SSH public host keys

OpenSSH – helper applications

- ssh-rand-helper
  - •entropy gathering mechanism for random number generation
- ssh-askpass
  - •GUI for passphrase entry, called by ssh-add.
- ssh-keysign
  - •helper program for hostbased authentication

OpenSSH – port forwarding (tunneling)

- Insecure TCP/IP protocols can be made secure by forwarding the connections through SSH

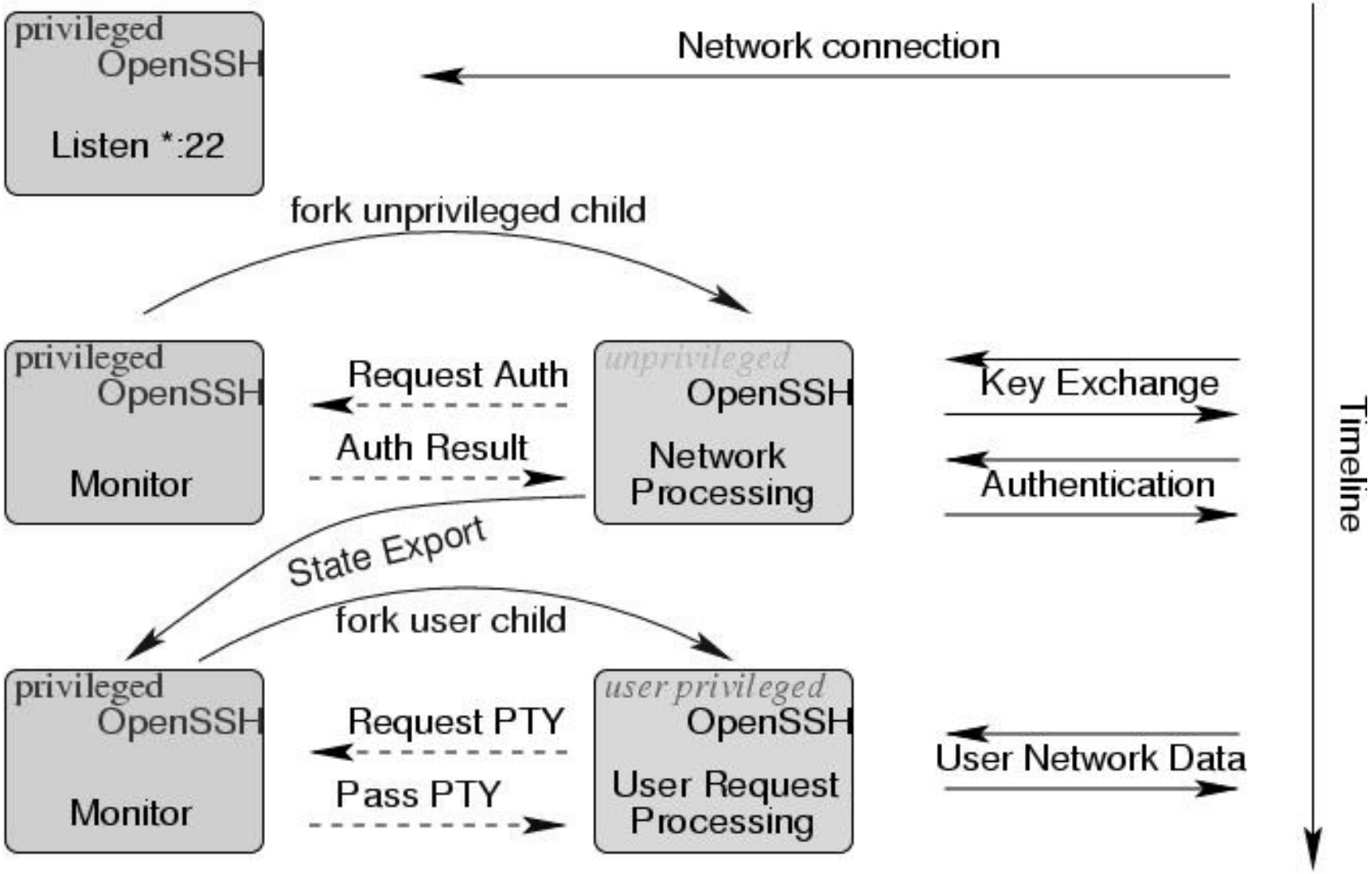
- This also means that firewalls can be bypassed.
  - The administrator configuring OpenSSH may want to consult with network and/or security administrators or policies.

- This feature is **enabled** by default.
  Users with shell access can install their own forwarders.

OpenSSH – Privilege Separation

- To reduce the effect of possible programming errors, sshd operates under the “principle of least privilege”
- The sshd daemon forks a copy of itself, switches to an unprivileged user (“SSHD”), and that process handles all the network traffic and anything not needing privileges.
- The “SSHD” privilege separation user is NOT the user who starts the daemon.

OpenSSH – Privilege Separation



OpenSSH – and syslogd

- OpenSSH daemon (sshd) can write to the UNIX syslog

- Log Facilities:
  - DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7.
  - The default is AUTH.

- Log Levels:
  - QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2 and DEBUG3.
  - The default is INFO

Interactions & Dependencies

- Hardware
  - No hardware required. Today all encryption done through software, via statically-linked OpenSSL
  - RNG through hardware possible on z/OS 1.7

- Software
  - Need z/OS 1.4 or higher.
  - Customers can now use existing PC ssh clients (e.g. PuTTY, WinSCP) to connect to z/OS UNIX.
  - Customers can now secure their existing X11 connections (e.g. Exceed)

Interactions & Dependencies – Service required

Product	z/OS 1.4	z/OS 1.5
Communications Server	APAR: PQ84183 PTF: UQ85501	APAR: PQ82209 PTF: UQ83463
Communications Server	APAR: PQ80471 PTF: UQ82236	APAR: PQ80471 PTF: UQ82237
Real Storage Manager (RSM)	APAR: OA05893 PTF: UA09169	APAR: OA05893 PTF: UA09170

OpenSSH 3.8.1p1 is the latest version on z/OS

- Provided via the **service stream – OA10315**
- OpenSSL 0.9.7d
- What's new:
  - Multilevel Security support
  - Password reset capability
  - Daemon restart capability
    - if TCP/IP is recycled, sshd will not go down, but will wait until TCP/IP returns and re-initialize itself.
    - if sshd started from /etc/rc and TCP/IP hasn't been started yet, sshd will wait for TCP/IP to come up.
    - in CINET environment, a new stack will automatically be recognized by the daemon (no SIGHUP required.)

(continued…)

OpenSSH 3.8.1p1 – What's new (continued)

- RNG through hardware possible on z/OS 1.7
  - Eliminate time-out issue with using ssh-rand-helper
  - Get random numbers faster

- Prerequisites
  - Integrated Cryptographic Service Facility (ICSF)
  - Userid must have read access to RACF CSFSERV class:

    CSFRNG - random number generate service

If ICSF is not available or userid does not have read access to the above RACF resource, SSH will revert to using ssh-rand-helper.

OpenSSH 3.8.1p1 – What's new (continued)

- Open Source function added in OpenSSH 3.8.1p1
  - New encryption algorithms, command-line options
  - Contains fixes, reduces vulnerabilities
  - New Configuration keywords:
    - ssh_config
      - AddressFamily
      - ConnectTimeout
      - EnableSSHKeysign
      - ForwardX11Trusted
      - IdentitiesOnly
      - ServerAliveInterval
      - ServerAliveCountMax
      - TCPKeepAlive
      - VerifyHostKeyDNS

    sshd_config
      - TCPKeepAlive
      - UseDNS

OpenSSH 3.8.1p1 - Changes to Consider

- The following configuration keywords were changed
  - the previous names are still supported on z/OS, but not by the OpenSSH distribution.
  - we recommend you move to new settings, once all systems sharing a config file have been upgraded.

daemon (sshd_config)

OpenSSH 3.5p1	**OpenSSH 3.8.1p1**
KeepAlive	TCPKeepAlive
VerifyReverseMapping	UseDNS

client (ssh_config)

KeepAlive	TCPKeepAlive

NOTE: The default settings for these configuration keywords have not changed.

Migration/Coexistence Considerations

- The RhostsAuthentication keyword was removed from the OpenSSH base distribution.
  - This method of authentication is not secure.
  - IBM z/OS **still maintains support** but recommends against using this setting.
  - It is both a client and daemon configuration option
  - Protocol Version 1 only, and defaults are “no” (disabled)

- In OpenSSH 3.5p1, HostbasedAuthentication automatically enabled use of ssh-keysign.
  - For OpenSSH 3.8.1p1, ssh-keysign is controlled by a separate (new) configuration keyword: EnableSSHKeysign
  - Must be set in the global ssh_config file.
  - default settings for both HostbasedAuthentication and EnableSSHKeysign are “no” (disabled.)

Migration/Coexistence Considerations

- OpenSSH is a program product built for z/OS 1.4. However….

- A version of OpenSSH has been on our z/OS Unix Tools & Toys page:
  http://www-1.ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html

  The customer should remove all executables from that version before installing the OpenSSH officially provided by IBM.

  The customer will want to keep their key files (host keys, user keys), and merge their existing configuration files with our samples.

Server Configuration

- No PARMLIB
- One new PROCLIB statement:
  - Define the privilege separation user

```
ADDGROUP SSHDG OMVS(GID(xxx))
ADDUSER SSHD DFLTGRP(SSHDG) OMVS(UID(yyy) HOME('/var/empty')
      PROGRAM('/bin/false')) NOPASSWORD
```

where *xxx* is an unused group ID, and *yyy* is an unused nonzero user ID.

- Decide how you want to start the daemon (standalone or through inetd)

Server Configuration

- Copy and modify sample configuration files

```
cp -p /samples/sshd_config /etc/ssh/sshd_config
cp -p /samples/ssh_config /etc/ssh/ssh_config
cp -p /samples/moduli /etc/ssh/moduli
cp -p /samples/ssh_prng_cmds /etc/ssh/ssh_prng_cmds
```

- Modify the **/etc/ssh/sshd_config** file to control the SSH server's:
  - Listen addresses and ports used
  - Authentication methods allowed
  - Protocols (Protocol Version 2, 1, or both)
  - Ciphers supported
  - Port forwarding
  - Session control options

Server Configuration

- Perform setup for server authentication:
  - Generate host keys for server
    - allows a client to verify the identity of the server.
    - Use ssh-keygen to create host keys:

```
ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key -N ""
ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ""
```

  - Create local and remote ssh_known_hosts files
    - Contains host public keys for all hosts you know about
    - Copy local host's public keys to the remote hosts
    - Gather public keys of remote hosts

Known Hosts File Setup

HOST1

HOST2

TIME

- Create host keys for HOST1
- Copy public host keys for HOST1 to client (HOST2)

OR

- Run ssh-keyscan against HOST1 to gather the host keys2
- Add hostkeys for HOST1 to ssh_known_hosts

Now users from HOST2 can identify HOST1 when they ssh to it.

- Run ssh-keyscan against HOST2 to gather its public host keys
- Add hostkeys for HOST2 to ssh_known_hosts

OR

- copy public keys for HOST2 to HOST1

Now users from HOST1 can identify HOST2 when they ssh to it.

User Configuration

- Copy and modify sample configuration files

```
cp -p /samples/ssh_config $HOME/.ssh/config
```

- Perform setup for user authentication:
  - Generate public/private keys – will allow a client to verify its identity to the server

```
ssh-keygen –t dsa
```

  - Create/edit local and remote authorized_keys files
    - Lists the public keys (RSA or DSA) that can be used to log into the user's account.
    - You want to:
      - Copy local user's public keys to the remote hosts
      - Gather public keys of remote user accounts

User asks… “Why can’t I log in?”

- There are various setup problems or configurations that can prevent a user from successfully logging in using ssh:

User has too open permissions for directories for files	See StrictModes in sshd_config
Sys Admin limited the number of users allowed to attempt log in	See MaxStartups in sshd_config – (default 10)
Sys Admin denied particular user, group or IP address access to the system	See AllowUsers, DenyUsers, AllowGroups, DenyGroups in sshd_config. See “from=pattern-list” in sshd doc.
User waited too long to enter password.	See LoginGraceTime in sshd_config.
Sys Admin refused users onto the system	See /etc/nologin description (sshd)
User trying to use a particular authentication method, but failing.	The Sys Admin may have disabled that authentication method in the daemon. See sshd_config.

Performance Considerations

- What could hurt OpenSSH performance?
  - If DNS is not configured properly
    - The ssh client (with –v) will sit on the following line:
      `debug1: ssh_connect: needpriv 0`
  - If the user is running with the STEPLIB environment variable set
  - If the system does not have the SCEELPA data set in LPA list

Performance Considerations

- /usr/lib/ssh/ssh-rand-helper gathers random data.
- If your OpenSSH command, when run in verbose mode, seems to be waiting on this line:

  ```
  debug3: Seeding PRNG from /usr/lib/ssh/ssh-rand-helper
  ```

  then the commands listed in /etc/ssh/ssh_prng_cmds and run by ssh-rand-helper could be timing out.
- Run ssh-rand-helper manually (from your shell prompt) to see how many and which commands are timing out.

  ```
  /usr/lib/ssh/ssh-rand-helper –vvv
  ```

- If every command is timing out, look for tuning tips in the UNIX System Services Planning Book and the MVS Initialization and Tuning Guide.
- If only a few commands are timing out, consider editing your /etc/ssh/ssh_prng_cmds file to contain different commands.

How to tell if you are using /dev/random

To verify if SSH is using the hardware support, start ssh or sshd in verbose mode.

- If you are NOT using hardware support, you will see:

$ssh **-**vvv *user@host*
OpenSSH_3.8.1p1, OpenSSL 0.9.7d 17 Mar 2004
debug1: Reading configuration data /etc/ssh/ssh_config
debug3: Seeding PRNG from /usr/lib/ssh/ssh-rand-helper
...

- If you ARE using hardware support, you will see:

$ ssh **-**vvv *user@host* # using hardware support
OpenSSH_3.8.1p1, OpenSSL 0.9.7d 17 Mar 2004
debug1: Reading configuration data /etc/ssh/ssh_config
debug3: RNG is ready, skipping seeding

Server Configuration – So many options…

How to start the daemon?

ListenAddress?

/etc/nologin?

ClientAliveInterval?

Authentication methods?

Protocol 1 or 2?

StrictModes?

Ciphers?

ClientAliveCountMax?

SyslogFacility?

MaxStartups?

TCPKeepAlive?

Hash algorithms?

LogLevel?

DenyGroups?

Port forwarding?

PermitRootLogin?

AllowUsers?

LoginGraceTime?

X11 forwarding?

UsePrivilegeSeparation?

Would you like to use SSH Protocol 1 or Protocol 2?

Protocol 2 (SSH 2)	Protocol 1 (SSH 1)
fixes the CRC data integrity problem that SSH1 has	uses a simple CRC for data integrity, which turns out to be flawed
supports many other choices for symmetric ciphers, as well as many other new features.	less ciphers, less features
follows IETF draft	no IETF draft.

Would you like to use SSH Protocol 1 or Protocol 2?

The sshd_config keyword of Protocol:	means…
2 **(default)**	Use SSH Protocol Version 2
1	Use SSH Protocol Version 1
2,1	Try Protocol 2 first, if not available then use Protocol 1

Set this to “Protocol 2”

Which encryption and hash algorithms would you like to use?

- Data privacy- Symmetric algorithms for session data
  - Advanced Encryption Standard (AES) with 128, 192, or 256 bit keys
  - arcfour, blowfish, DES, triple-DES (3DES), and CAST-128
- Data integrity – hash algorithms (MACs)
  - SSH Protocol version 2 - uses MD5, SHA-1, RIPEMD-160
  - SSH Protocol version 1 – uses weaker CRC-32
- User and Host authentication – public key algorithms
  - RSA, DSA
- Session key exchange – Diffie-Hellman

Which encryption and hash algorithms would you like to use?

The **ssh_config** and **sshd_config** keywords of:	are used to specify…
Ciphers	Protocol Version 2 ciphers
MACs	Protocol Version 2 MACs

- Default value for Ciphers: (OpenSSH 3.8.1p1)

"aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc, aes128-ctr,aes192-ctr,aes256-ctr"

- Default value for MACS:

"hmac-md5,hmac-sha1,hmac-ripemd160,hmac-sha1-96,hmac-md5-96"

Would you like to use Password or Public Key User Authentication?

Password	**Public Key**
Uses the security product	Does NOT use the security product
passwords tend to be 1-8 characters	passphrases can be long “phrases” or sentences, separated by spaces
passwords get sent across the network (encrypted)	passphrases don’t travel across the network, they decrypt the local key
one-factor authentication system (only password needs stolen)	two-factor authentication system (both passphrase and private key would need compromised.)

Would you like to use Password or Public Key User Authentication?

A **sshd_config** setting of:	means…
PasswordAuthentication yes	Server allows password authentication.
PubkeyAuthentication yes	Server allows public key authentication.

A **ssh_config** setting of:	means…
PasswordAuthentication yes	Client attempts password authentication.
PubkeyAuthentication yes	Client attempts public key authentication.
PreferredAuthentications	Specifies order in which client tries Protocol Version 2 authentications

Would you like to control the addresses where sshd will listen?

The sshd_config keyword of:	Means…
ListenAddress	Specifies the local addresses where sshd will listen. The default is to listen on all local addresses.

Format:

```
ListenAddress host|IPv4addr|IPv6_addr
ListenAddress host|IPv4_addr:port
ListenAddress [host|IPv6_addr]:port
```

Would you like to enable logging to the UNIX syslog?

The sshd_config keyword of:	can be set to…
SyslogFacility	DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The default is AUTH.
LogLevel	QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2 and DEBUG3 The default is INFO

Would you like to disconnect users if they take too long to log in?

LoginGraceTime	Means…
120 seconds **(default)**	The server disconnects after 120 seconds if the user has not successfully logged in.
0	No time limit

Some configurations recommend 60 seconds.

Set this based on your site's policies.

Would you like to limit the number of not-yet-authenticated connections?

sshd_config keyword:	Means…
MaxStartups	Specifies the maximum number of concurrent unauthenticated connections to the sshd daemon. The default is 10 (i.e. 10 people can be in the process of logging in at any given point in time.)

Set this based on your site's policies.

Would you like to allow superusers to log in using ssh?

PermitRootLogin	Means…
“yes” **(default)**	A superuser IS allowed to login
“without-password”	A superuser may not log in using password authentication
“forced-commands-only”	public key authentication is allowed, but only if Authorized Keys File "command=" is used.
“no”	A superuser is NOT allowed to login

To avoid brute-force attacks against root password from untrusted machines, set to “no”, login as regular user, and su to root.

Would you like to control access for specific users or groups?

The sshd_config keyword of:	Means…
AllowUsers *Username_patterns*	Login allowed only for USER names which match one of the patterns
AllowGroups *Groupname_patterns*	Login allowed only for GROUP names which match one of the patterns
DenyUsers *Username_patterns*	Login disallowed only for USER names which match one of the patterns
DenyGroups *Groupname_patterns*	Login disallowed only for GROUP names which match one of the patterns

Would you like to control access for specific users or groups?

Examples:

to allow only users *user1* and *user2*

```
AllowUsers user1 user2
```

to allow only usernames starting with pattern “user”

```
AllowUsers user*
```

to deny all users coming from host *badhost*

```
DenyUsers *@badhost
```

Do you want to prevent a user from logging in using empty passwords?

PermitEmptyPasswords	Means…
“no” (default)	Empty passwords are not allowed.
“yes”	Server allows login to accounts with empty password strings.*

- WARNING: Empty passwords may be INADVERTENTLY be allowed through a SURROGAT class.
- The MVS identity running sshd SHOULD NEVER BE GIVEN read access to a SURROGAT class profile, BPX.SRV.uuuuuuuu (where uuuuuuuu is the MVS userid for each user who is permitted to log in with an empty password.)
- This would allow ANY user to login to userid uuuuuuuu without a password.

Would you like to enable privilege separation?

The sshd_config setting of:	Means…
UsePrivilegeSeparation yes	privilege separation is enabled. **(default)**
UsePrivilegeSeparation no	privilege separation is disabled.

- NOTE: Privilege Separation cannot be used with compression.
- Set this to “yes”.

Would you like to allow port forwarding (tunneling)?

The sshd_config keywords:	Means…
AllowTcpForwarding	Specifies whether TCP forwarding is permitted. **(default is "yes")**
X11Forwarding	Specifies whether X11 forwarding is permitted. (default is "no".)
GatewayPorts	Specifies whether remote hosts are allowed to connect to ports forwarded by the client. Set to "no" ("no" is default.)

Would you like to prevent users from inadvertently exposing their private keys?

The sshd_config setting of:	Means…
StrictModes yes **(default)**	sshd checks file modes and ownership of the user's files and home directory before accepting login.
StrictModes no	the check is not performed.

If you allow public key authentication, this should be set this to “yes”.

Other configuration options of interest…

- If /etc/nologin file exists…
  - Only superusers (UID 0) can log in. Non-UID 0 users (including those permitted to BPX.SUPERUSER) cannot log in, and get displayed contents of the /etc/nologin file.
  - Consider the current setting of PermitRootLogin before creating this file.
- OpenSSH supports hostbased, but better methods exist.
  - Disabled by default
- Use of login program (UseLogin keyword)
  - disabled by default
  - not compatible with privilege separation

Would you like ssh to detect if a host key changed due to DNS spoofing?

The ssh_config setting of…	Means…
CheckHostIP yes **(default)**	ssh checks for host's IP address in the known_hosts file. (i.e. both hostname and IP address must be in known_hosts file.) If IP address missing, it is added and a warning is displayed.
CheckHostIP no	check is not performed.

How strict would you like the ssh client to be when checking host keys?

StrictHostKeyChecking	Means…
“yes”	ssh will never automatically add host keys to the $HOME/.ssh/known_hosts file ssh will refuse to connect to a host whose host key has changed.
“ask” **(default)**	new host keys will be added to the user known host files only after the user has confirmed the action. ssh will refuse to connect to hosts whose host key has changed.
“no”	ssh will automatically add new host keys to the user known hosts files.

Session Summary

- OpenSSH enhances security of z/OS platforms
- OpenSSH adds function to help enhance security of existing applications

Appendix

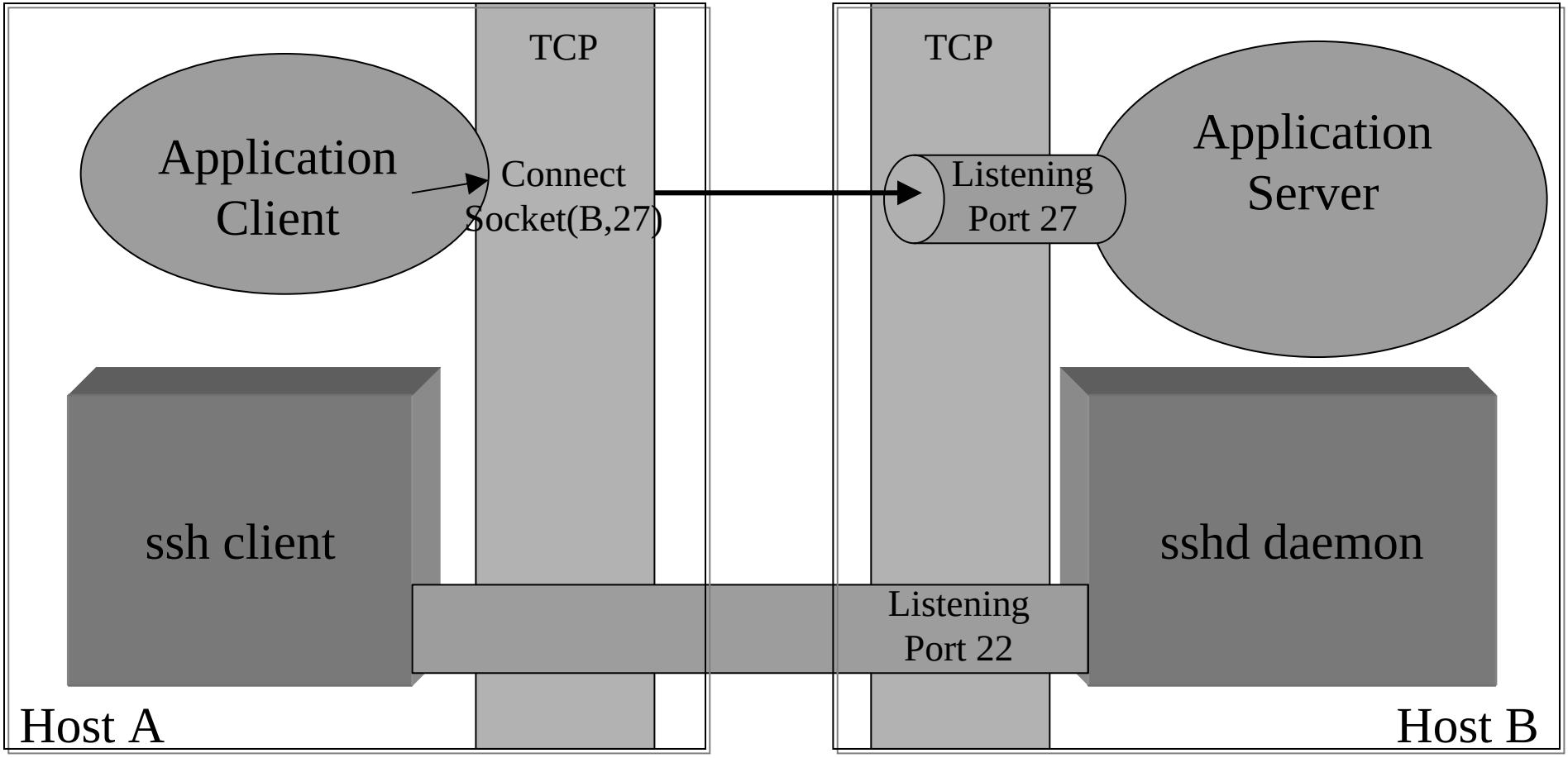
- OpenSSH home page: www.openssh.org
- IETF **Secure Shell (secsh) Working Group**
  - **http://www.ietf.org/html.charters/secsh-charter.html**
- SSH The Secure Shell, The Definitive Guide. Barret & Silverman. 2001 O'Reilly & Associates, Inc.
- z/OS Publication. IBM Ported Tools for z/OS

Appendix – Supporting Slides

- Local Port Forwarding

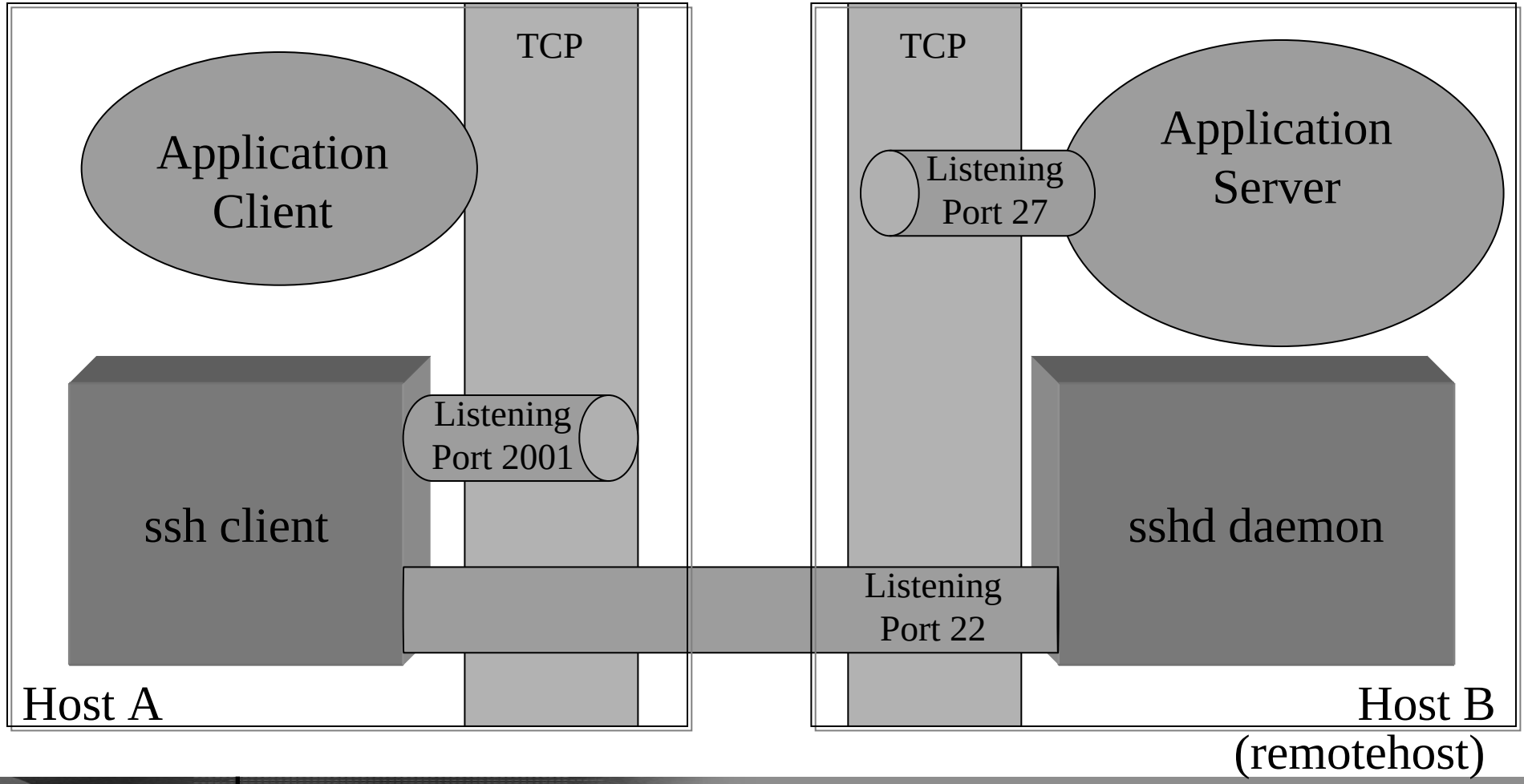
OpenSSH – Without TCP/IP Port Forwarding

Direct client/server connection (no forwarding)



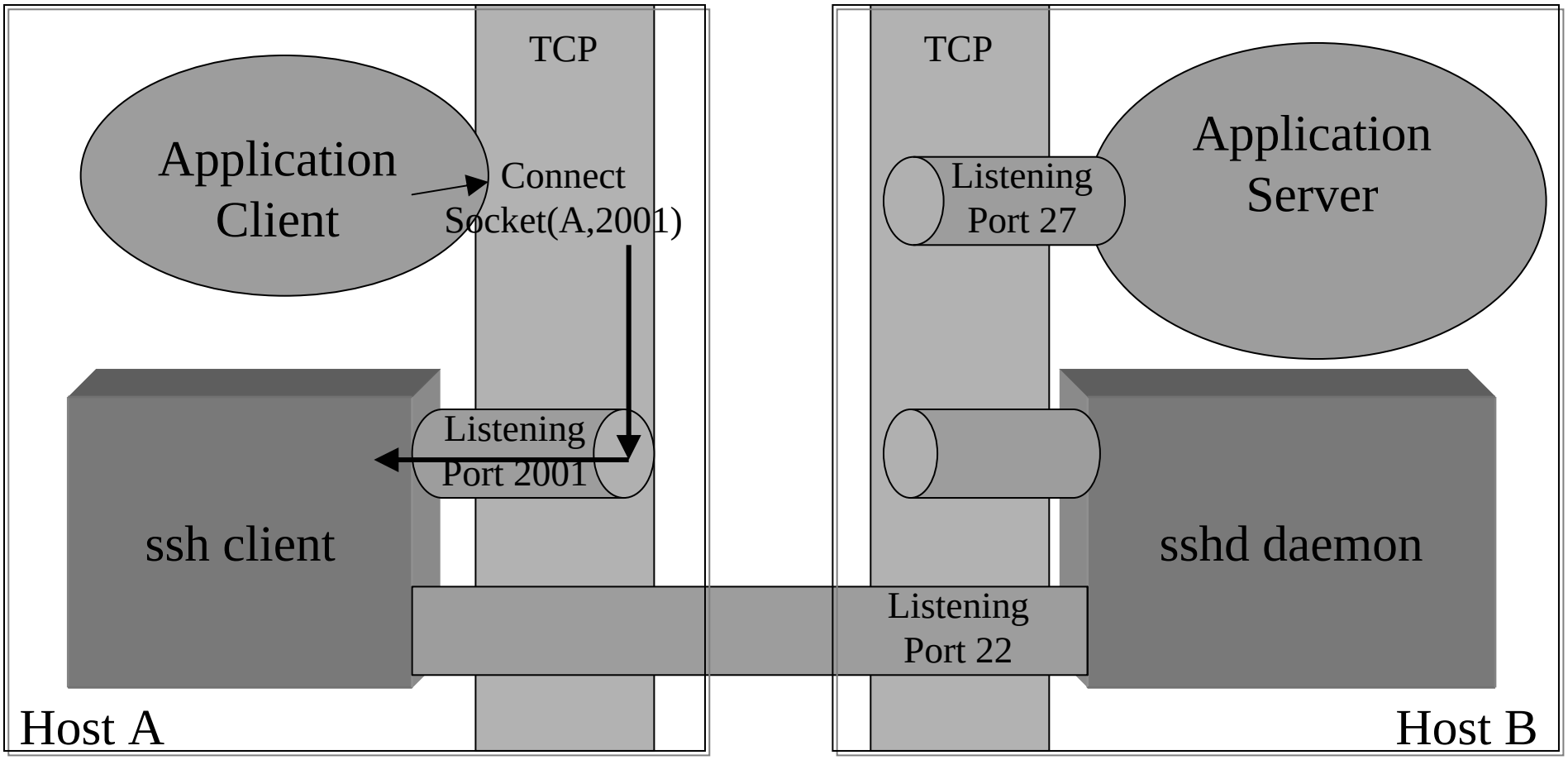
OpenSSH – TCP/IP Port Forwarding

ssh –L 2001:remotehost:27 billy@remotehost



OpenSSH – TCP/IP Port Forwarding

TCP/IP application wants to contact server, through SSH connection



OpenSSH – TCP/IP Port Forwarding

ssh forwards the data through an SSH tunnel, sshd delivers to server