Exploiting Crypto Express & CPACF Hardware with Linux

Richard Young

IBM STG Lab Services

9:30am Monday March 10th, 2014

www.SHARE.org

Agenda

1 zEnterprise Crypto Hardware Background

2 Making the Cryptographic Hardware Available to Linux

3 Enabling Linux to use the Hardware

4 Enabling Java and WebSphere to Exploit the Crypto Hardware

5 Configuring the IBM HTTP Server to use the Crypto Hardware

6 Enabling the WAS Plugin to Use the Crypto Hardware

7 openSSL and openSSH

8 In Kernel Crypto and DM-Crypt

Agenda

1   **zEnterprise Crypto Hardware Background**

2   Making the Cryptographic Hardware Available to Linux

3   Enabling Linux to use the Hardware

4   Enabling Java and WebSphere to Exploit the Crypto Hardware

5   Configuring the IBM HTTP Server to use the Crypto Hardware

6   Enabling the WAS Plugin to Use the Crypto Hardware

7   openSSL and openSSH

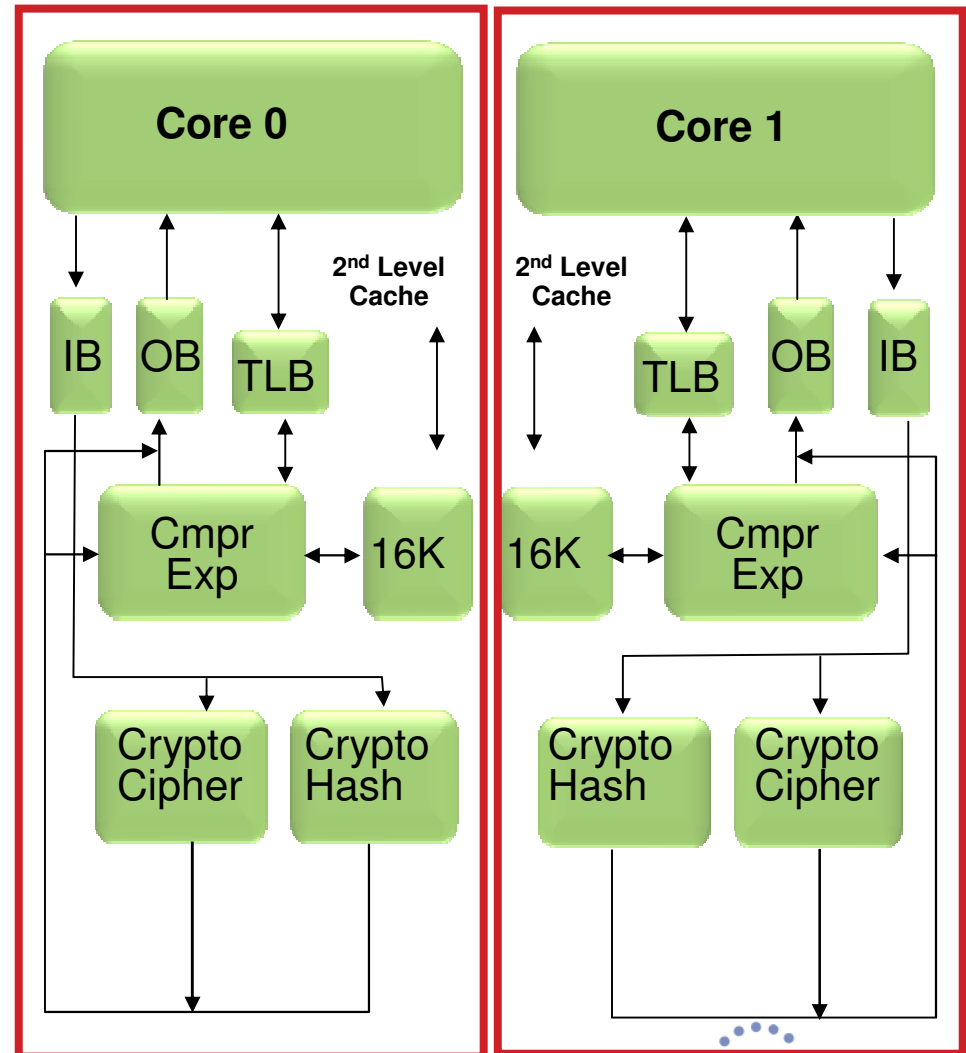**8**   In Kernel Crypto and DM-Crypt

zEnterprise Crypto Background

- **System z has two categories of crypto hardware**
  - **CPACF** – Provides support for symmetric ciphers and secure hash algorithms (SHA) on every central processor. The potential encryption/decryption throughput scales with the number of CPs.
  - **CEX** – **The Crypto Express feature can be configured in three ways:** Either as cryptographic Coprocessor (CEXC) for secure key encrypted transactions, or as cryptographic Accelerator (CEXA) for Secure Sockets Layer (SSL) acceleration. A CEXA works in clear key mode. The Crypto Express 4S allows for a third mode as a Secure IBM CCA Coprocessor
- **The solutions in this presentation make use of clear key acceleration**

SHARE
in Anaheim

zEC12 Compression and Cryptography Accelerator

- Coprocessor dedicated to each core (Was shared by two cores on z196)
  - Independent compression engine
  - Independent cryptographic engine
  - Available to any processor type
  - *Owning processor is busy when its coprocessor is busy*
- Data compression/expansion engine
  - Static dictionary compression and expansion
- CP Assist for Cryptographic Function
  - **290-960 MB/sec** bulk encryption rate
  - DES (DEA, TDEA2, TDEA3)
  - SHA-1 (160 bit)
  - SHA-2 (244, 256, 384, 512 bit)
  - AES (128, 192, 256 bit)
  - **CPACF FC 3863 (No Charge)** is required to enable some functions and is also required to support Crypto Express4S or Crypto Express3 features

Core 0 | **Core 1**

2nd Level Cache | 2nd Level Cache

IB | OB | TLB | TLB | OB | IB

Cmpr Exp | 16K | 16K | Cmpr Exp

Crypto Cipher | Crypto Hash | Crypto Hash | Crypto Cipher

Crypto Express4S

- One PCIe adapter per feature
  - Initial order – two features
- FIPS 140-2 Level 4
- Installed in the PCIe I/O drawer
- Up to 16 features per server
- Prerequisite: CPACF (FC 3863)

- Accelerator
  - Clear Key (RSA <= 4k)
- Coprocessor
  - Clear Key (RSA <= 4k and RNG)
  - Secure Key (ECC via CCA)

- Three configuration options for the PCIe adapter
  - Only one configuration option can be chosen at any given time
  - Switching between configuration modes will erase all card secrets
    - Exception: Switching from CCA to accelerator or vice versa
- **Accelerator**
  - For SSL acceleration
  - Clear key RSA operations
- **Enhanced: Secure IBM CCA coprocessor** (default)
  - Optional: TKE workstation (FC 0841) for security-rich, flexible key entry or remote key management
- **New: IBM Enterprise PKCS #11 (EP11) coprocessor**
  - Designed for extended evaluations to meet public sector requirements
    - Both FIPS and Common Criteria certifications
  - Required: TKE workstation (FC 0841) for management of the Crypto Express4S when defined as an EP11 coprocessor
  - Supported on Crypto Express4S only
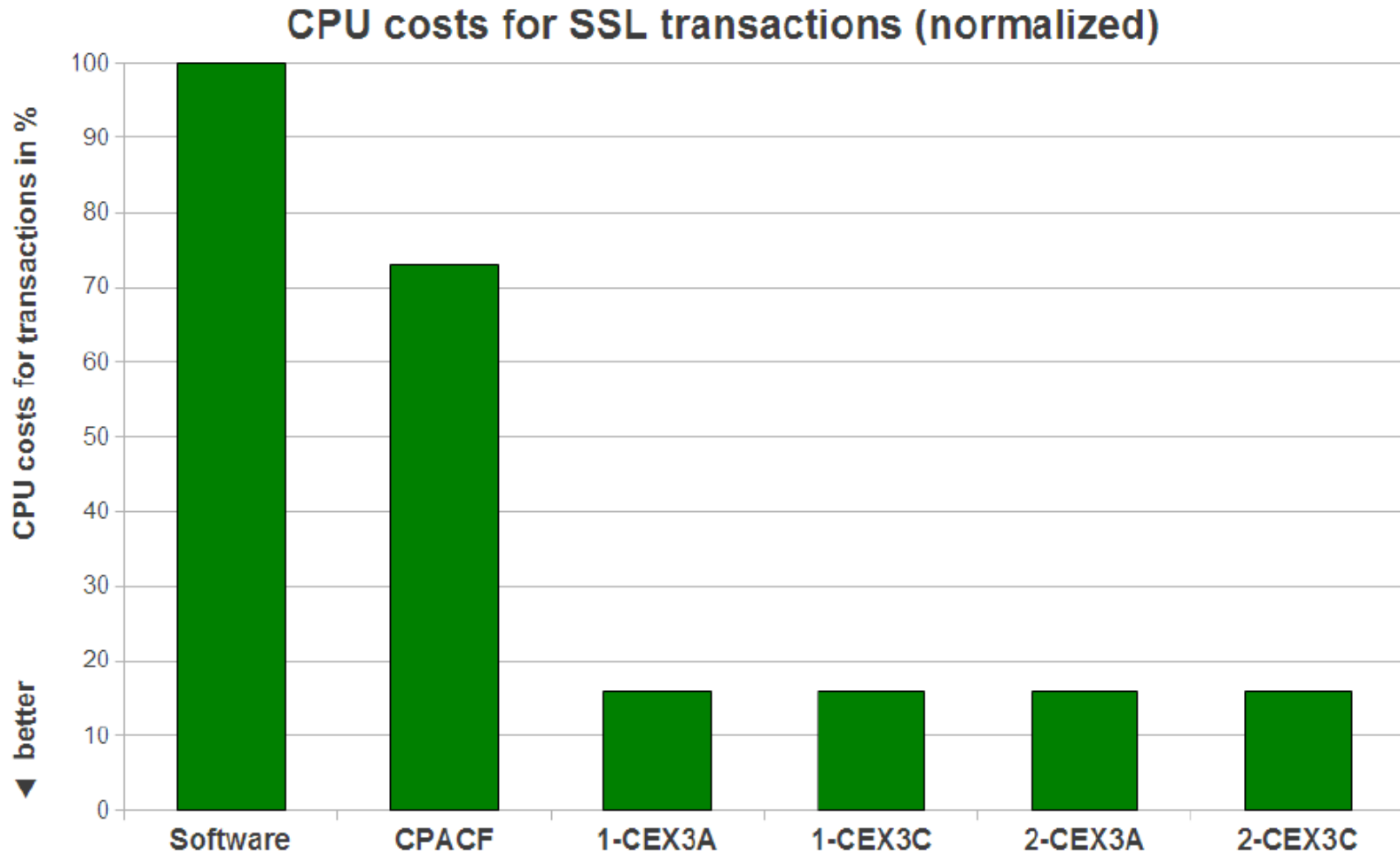
Value of Cryptographic Hardware

SSL transaction throughput (normalized)
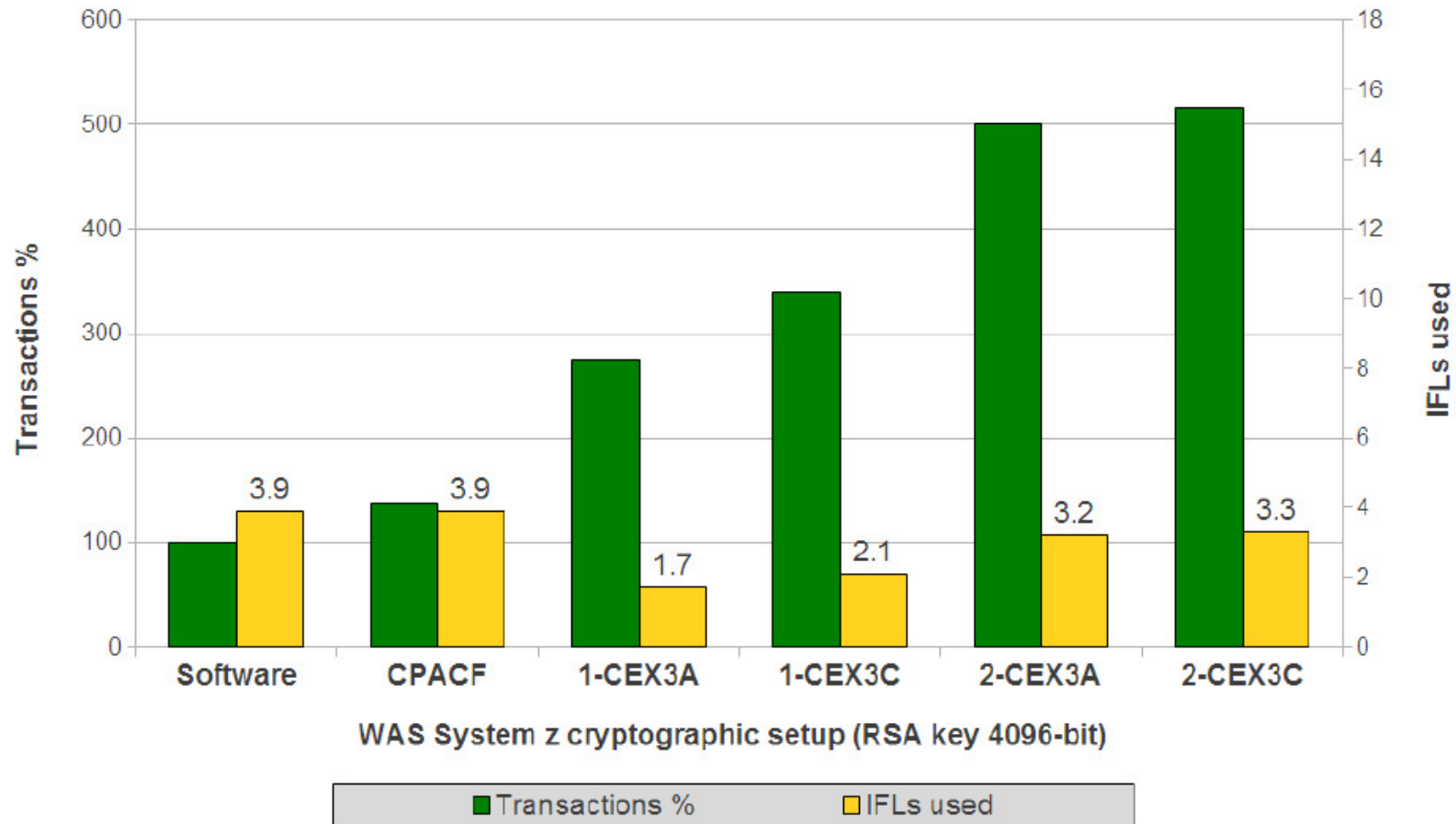
WAS System z cryptographic setup (RSA key 4096-bit)

Legend: Transactions % | IFLs used

For additional details see: ZSW03250-USEN-00.pdf

Value of Cryptographic Hardware

CPU costs for SSL transactions (normalized)

For additional details see: ZSW03250-USEN-00.pdf

Value of Cryptographic Hardware

Average response times for the SSL transactions
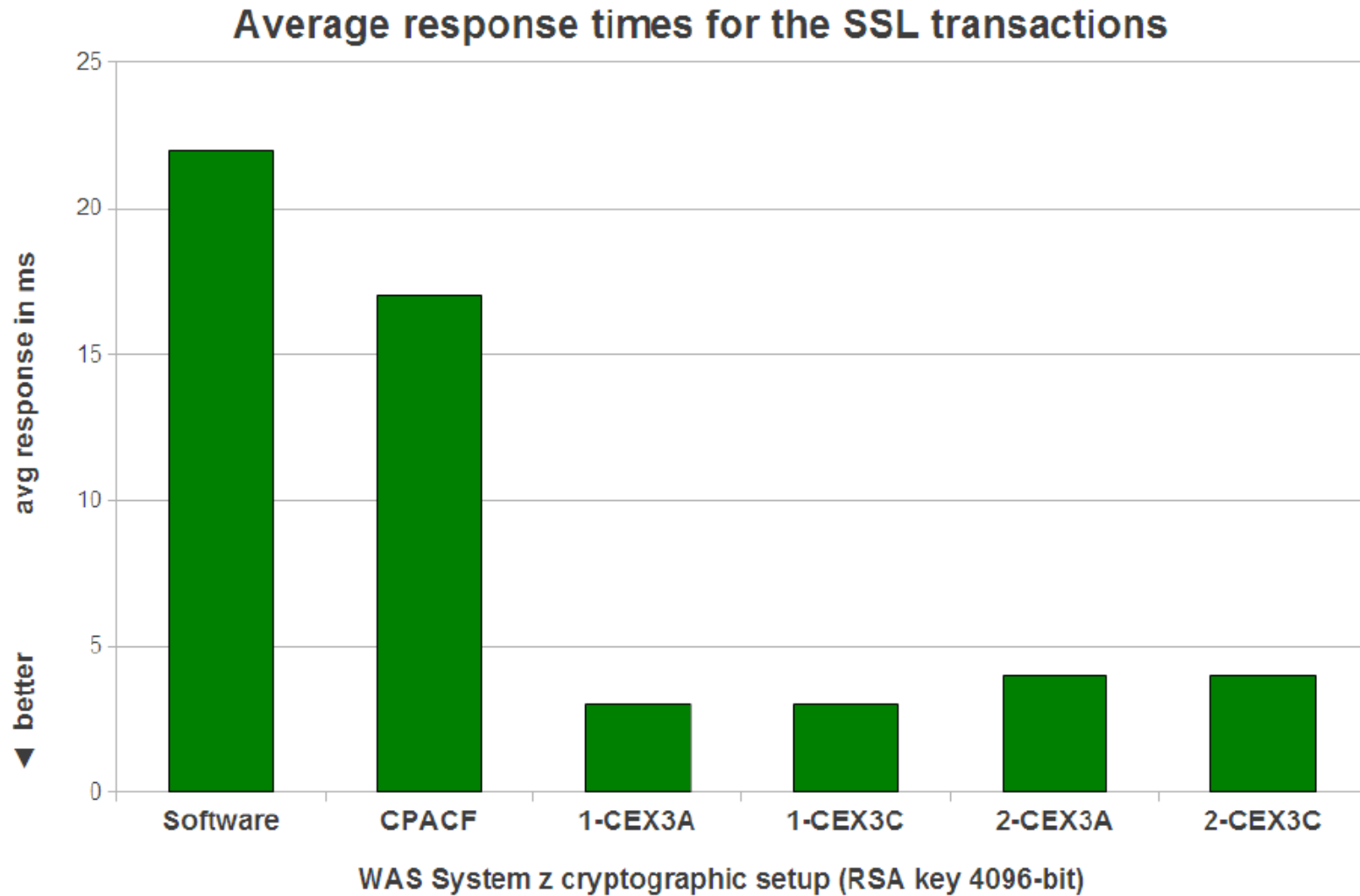
For additional details see: ZSW03250-USEN-00.pdf

zEnterprise Crypto Background

- OpenSSL needs the engine ibmca to communicate with the interface library (libICA). The libICA library then communicates with CPACF or via the Linux generic device driver z90crypt with a CEX (if available). The device driver z90crypt must be loaded in order to use CEX features.

- **Many potential exploiters**
  - WebSphere Application Server/Portal
  - Java Applications
  - IBM HTTP Server
  - Apache
  - WebSphere Plugin
  - Linux SSH, SFTP, SCP
  - In Kernel Crypto Exploiters
    - DM-Crypt
    - IPSec
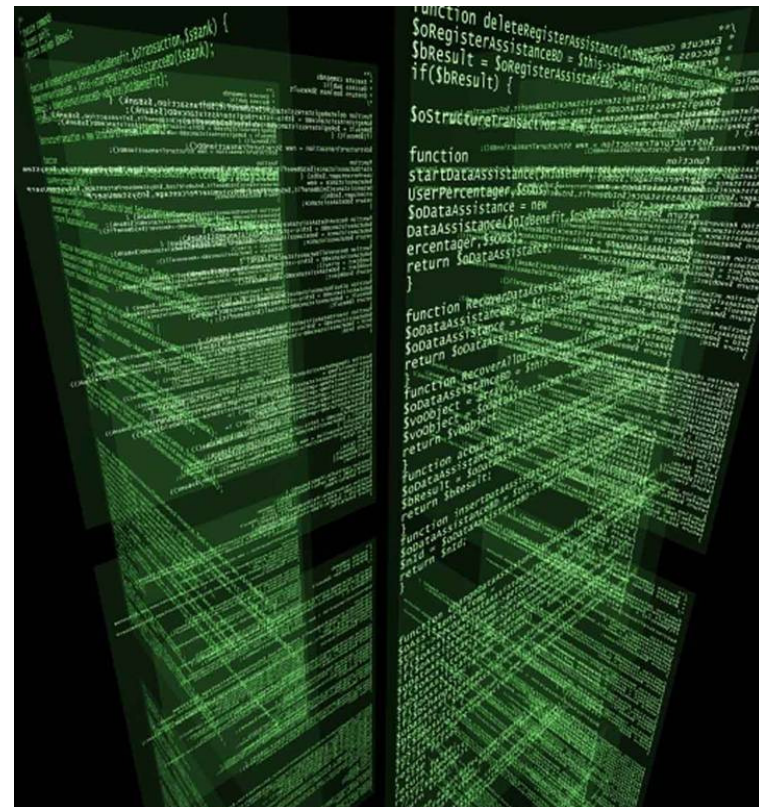
Linux on System z Crypto Stack
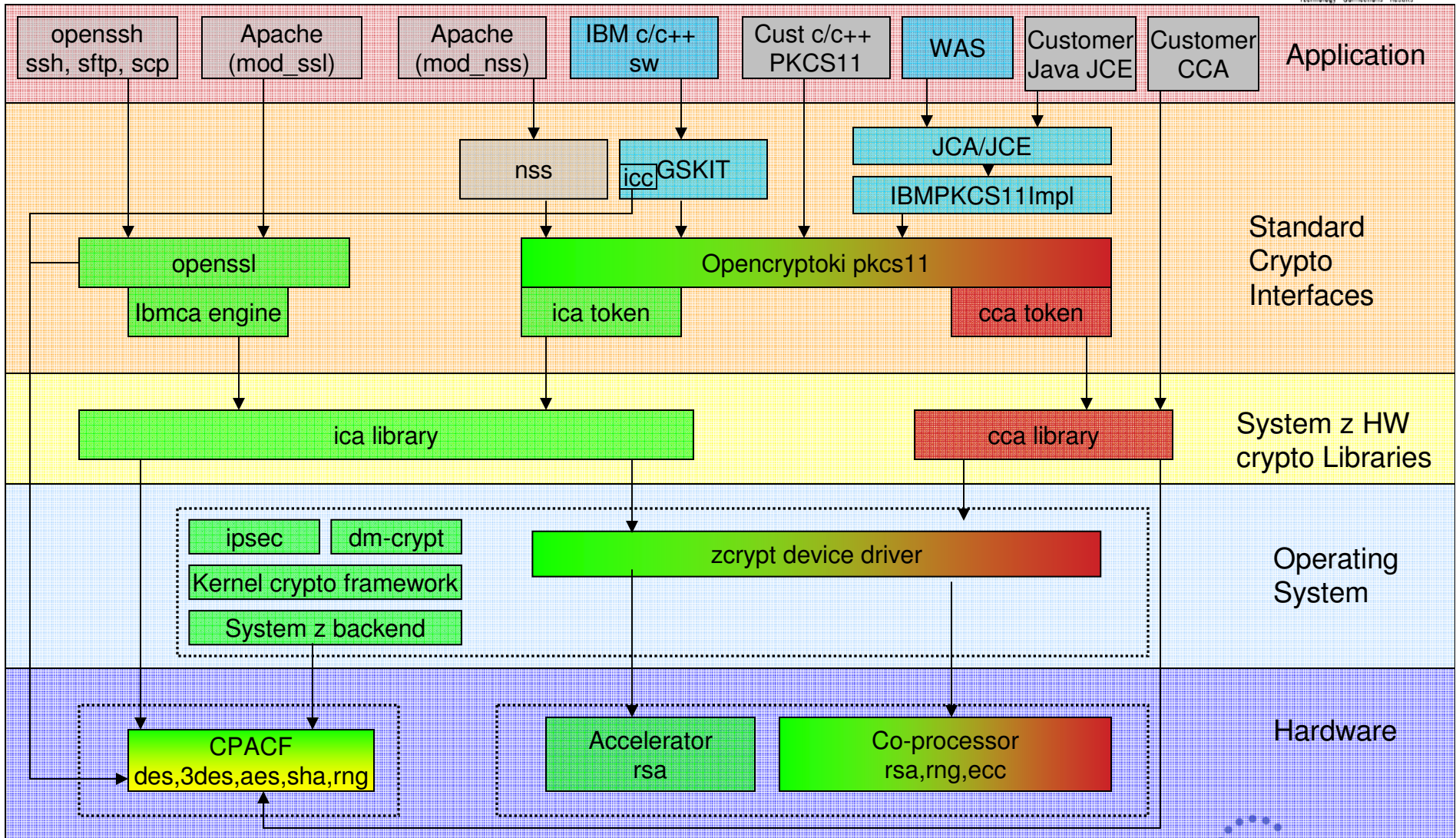
openssh ssh, sftp, scp	Apache (mod_ssl)	Apache (mod_nss)	IBM c/c++ sw	Cust c/c++ PKCS11	WAS	Customer Java JCE	Customer CCA	**Application**

Standard Crypto Interfaces

- nss
- icc GSKIT
- JCA/JCE
- IBMPKCS11Impl
- openssl
- Ibmca engine
- Opencryptoki pkcs11
- ica token
- cca token

System z HW crypto Libraries

- ica library
- cca library

Operating System

- ipsec
- dm-crypt
- Kernel crypto framework
- System z backend
- zcrypt device driver

Hardware

- CPACF des,3des,aes,sha,rng
- Accelerator rsa
- Co-processor rsa,rng,ecc

Legend: Clear | Protected | Secure

SHARE in Anaheim

Agenda

1. zEnterprise Crypto Hardware Background

2. **Making the Cryptographic Hardware Available to Linux**

3. Enabling Linux to use the Hardware

4. Enabling Java and WebSphere to Exploit the Crypto Hardware

5. Configuring the IBM HTTP Server to use the Crypto Hardware

6. Enabling the WAS Plugin to Use the Crypto Hardware

7. **openSSL and openSSH**

8. **In Kernel Crypto and DM-Crypt**

Making the Crypto Hardware available to Linux

- Crypto Express hardware needs to be configured at the HMC to the LPAR
- Need to make the Crypto Express Accelerator available to the guest when running under z/VM
- This can be configured as dedicated, shared, or to a specific crypto domain
- The CPACF is automatically made available with the PU, you just need to ensure the enabling microcode is on. (Feature code 3863)
- Supported algorithms vary by processor model and Express card

Making the Crypto Hardware available to Linux

Making the Crypto Hardware available to Linux

```
USER RGYLXWS8 4Z2M6UN2 2G 5G G
    INCLUDE LINDFLT
    COMMAND DEFINE STORAGE 2G STANDBY 3G
    CPU 00
    CPU 01
    CRYPTO APVIRTUAL   ◄─────────────────
    MACHINE ESA 4
```

- `DIRM FOR RGYLXWS8 CRYPTO APVIRT`

  - Provides access to the CEX

  - Alternatively it could be dedicated, which is typically used for secure key operations

- Or add to your Linux directory "profile" (LINDFLT above)

- CPACF always available from a hardware virtualization perspective

Agenda

1 zEnterprise Crypto Hardware Background

2 Making the Cryptographic Hardware Available to Linux

3 **Enabling Linux to use the Hardware**

4 Enabling Java and WebSphere to Exploit the Crypto Hardware

5 Configuring the IBM HTTP Server to use the Crypto Hardware

6 Enabling the WAS Plugin to Use the Crypto Hardware

Preparing Linux – Required packages

- **Required Packages**
  - libica (qty 2)
  - openCryptoki (qty 3)
  - Given continued enhancements, the more current the better
- **Examples shown in the presentation are with:**
  - WebSphere V8
  - SLES 11 SP2

```
RGYLXWS8:/ # rpm -qa | grep libica-
libica-2_1_0-2.1.0-0.9.1
libica-2_1_0-32bit-2.1.0-0.9.1
RGYLXWS8:/ # rpm -qa | grep openCrypto
openCryptoki-32bit-2.4-0.9.1
openCryptoki-2.4-0.9.1
openCryptoki-64bit-2.4-0.9.1
RGYLXWS8:/ #
```

About the Linux Packages

- openCryptoki provides the pkcs11 interfaces
- PKCS #11 is one of the family of standards for **Public Key Cryptography Standards.** It provides a platform independent API to cryptographic tokens (such as HSMs).

- libica - The Library for IBM® Cryptographic Architecture provides the header files and libica library to write cryptographic programs both with and without cryptographic hardware. It also provides programs such as icainfo and icastats

Preparing Linux - Validating CPACF

```
RGYLXWS8:/ # icainfo
The following CP Assist for Cryptographic Function (CPACF) operations are
supported by libica on this system:
SHA-1:          yes
SHA-256:        yes
SHA-512:        yes
DES:            yes
TDES-128:       yes
TDES-192:       yes
AES-128:        yes
AES-192:        yes
AES-256:        yes
PRNG:           yes
CCM-AES-128:    no
CMAC-AES-128:   no
CMAC-AES-192:   no
CMAC-AES-256:   no
```

- "icainfo" will show the cryptographic operations supported by libica on your system
- Influenced by processor model and microcode enablement feature

Preparing Linux

```
RGYLXWS8:/ # rcz90crypt start
Loading z90crypt module
RGYLXWS8:/ # rcpkcsslotd start
Starting pkcsslotd daemon:usermod: `root' is primary group name.
```

- **The z90crypt module and pkcsslot daemon must be loaded and started. Do this dynamically with**
  - `rcz90crypt start`
  - `rcpkcsslotd start`
- **Don't forget to permanently enable**
  - `chkconfig z90crypt on`
  - `chkconfig pkcsslotd on`

Preparing Linux - Confirming CEX Adapter

```
RGYLXWS8:/ # cat /proc/driver/z90crypt

zcrypt version: 2.1.1
Cryptographic domain: 15
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 0
CEX3C count: 0
CEX3A count: 1                    <----------------------------
requestq count: 0
pendingq count: 0
Total open handles: 0


Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A 7=CEX3C 8=CEX3A
              0000000000000000 0000000000000000 0000800000000000 0000000000000000
```

- /proc/driver/z90crypt will shows the number and type of Crypto Express devices enabled to your system.

Preparing Linux

- Before the crypto hardware can be used the PKCS11 token must be initialized.

- Initializing the PKCS11 token/hardware requires a security officer and user PIN to be set

- BOTH must be changed after they are set before crypto operations can occur on the hardware

- A token label must also be set

- These setting are unique to the individual Linux guest, however they could be set on a Linux master image you clone from

Preparing Linux – Initialize Token and Change SO PIN

```
RGYLXWS8:/ # pkcsconf -c 0 -I
Enter the SO PIN:
Enter a unique token label: rgylxws8
RGYLXWS8:/ #
```

- The PKCS11 token is initialized with pkcsconf, a Security Officer PIN set, and a token label applied
- You will need to use this token label later
- The PINS must be changed after the initial setting

```
RGYLXWS8:/ # pkcsconf -c 0 -P
Enter the SO PIN:
Enter the new SO PIN:
Re-enter the new SO PIN:
RGYLXWS8:/ #
```

- The Security Officer PIN must be changed before proceeding further
- pkcsconf –c 0 –P

Preparing Linux – Status After Initialization

```
RGYLXWS8:/ # pkcsconf -t
Token #0 Info:
     Label: rgylxws8
     Manufacturer: IBM Corp.
     Model: IBM ICA
     Serial Number: 123
     Flags: 0x880445 (RNG|LOGIN_REQUIRED|CLOCK_ON_TOKEN|TOKEN_INITIALIZED|USER_PIN_TO_BE_CHANGED|S
O_PIN_TO_BE_CHANGED)
     Sessions: 0/-2
     R/W Sessions: -1/-2
     PIN Length: 4-8
     Public Memory: 0xFFFFFFFF/0xFFFFFFFF
     Private Memory: 0xFFFFFFFF/0xFFFFFFFF
     Hardware Version: 1.0
     Firmware Version: 1.0
     Time: 13:14:15
Token #1 Info:
     Label: IBM OS PKCS#11
     Manufacturer: IBM Corp.
     Model: IBM SoftTok
     Serial Number: 123
```

- After initializing the token hardware is still not ready
- pkcsconf -t
- When ready the flags will be 0x44D
- Ensure you are checking the correct token/label

SHARE
in Anaheim

Preparing Linux – Set and Change the User PIN

```
RGYLXWS8:/ # pkcsconf -c 0 -u
Enter the SO PIN:
Enter the new user PIN:
Re-enter the new user PIN:
```

- The User PIN is set, the SO PIN is required for this operation
- pkcsconf –c 0 –u

```
RGYLXWS8:/ # pkcsconf -c 0 -p
Enter user PIN:
Enter the new user PIN:
Re-enter the new user PIN:
```

- The User Pin must be changed before use also.
- pkcsconf –c 0 –p

Preparing Linux –> 0x44D = Ready

```
RGYLXWS8:/ # pkcsconf -c 0 -p
Enter user PIN:
Enter the new user PIN:
Re-enter the new user PIN:
RGYLXWS8:/ # pkcsconf -t
Token #0 Info:
        Label: rgylxws8
        Manufacturer: IBM Corp.
        Model: IBM ICA
        Serial Number: 123
        Flags: 0x44D (RNG|LOGIN_REQUIRED|USER_PIN_INITIALIZED|CLOCK_ON_TOK
        Sessions: 0/-2
        R/W Sessions: -1/-2
        PIN Length: 4-8
        Public Memory: 0xFFFFFFFF/0xFFFFFFFF
        Private Memory: 0xFFFFFFFF/0xFFFFFFFF
        Hardware Version: 1.0
        Firmware Version: 1.0
```

- When ready for use, the Flags value is 0x44D, anything else does not work

Preparing Linux Summary

- **You can configure the Security Officer and User PINs on your master image**
- **PINS must be changed after they are initially set**
- **Crypto APVIRT (or variation) required for a virtualized CEX**
- **Get the most current libica and openCryptoki from your distributor for your version / release**
- **Monitor with**
  - /proc/driver/z90crypt
  - icainfo/icastats
  - pkcsconf –t
  - lszcrypt

Agenda

1 zEnterprise Crypto Hardware Background

2 Making the Cryptographic Hardware Available to Linux

3 Enabling Linux to use the Hardware

4 **Enabling Java and WebSphere to Exploit the Crypto Hardware**

5 Configuring the IBM HTTP Server to use the Crypto Hardware

6 Enabling the WAS Plugin to Use the Crypto Hardware

7 openSSL and openSSH

8 In Kernel Crypto and DM-Crypt

Enabling Java and WebSphere

- **Steps**
  - Update the Java policy files to be unrestricted
  - Set the WebSphere JVM Custom property
  - Create the hardware token file
  - Update the Java security file
  - Customize the WebSphere Cipher Suite
  - Make userid(s) part of the PKCS11 group
  - Validate use of the hardware

Enabling Java and WebSphere

- IBM SDKs ship with a strong but limited set of policy files.
- To use the strongest encryption you need to update the policy files with the unrestricted version.
- The link for SDK 6 is:

  `https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk`

- The updated files go in:

  `/opt/IBM/WebSphere/AppServer/java/jre/lib/security`

Enabling Java and WebSphere

```
RGYLXWS8:~ # cd /opt/IBM/WebSphere/AppServer/java/jre/lib/security/
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # ls
US_export_policy.jar  cacerts      java.security      trusted.libraries
blacklist             java.policy  local_policy.jar
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # cp local_policy.jar local_policy.jar.or
iginal
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # cp US_export_policy.jar US_export_polic
y.jar.orignal
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # cp /root/local_policy.jar .
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # cp /root/US_export_policy.jar .
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # cp local_policy.jar local_policy.jar-un
restricted
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # cp US_export_policy.jar US_export_polic
y.jar-unrestricted
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security #
```

- Unzip the unrestricted.zip in a temporary work directory
- Copy the local_policy.jar and US_export_policy.jar in to the WebSphere java/jre/lib/security directory
- Backup copies are handy, as maintenance to the SDK will overlay your unrestricted file with the restricted one.
- Set permissions and ownership as desired

Enabling Java and WebSphere

Application servers > **server1** > **Process definition** > **Java Virtual Machine** > **Custom properties**

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

[+] Preferences

	Select	Name ◇	Value ◇	Description ◇
		You can administer the following resources:		
	☐	com.ibm.security.jgss.debug	off	
	☐	com.ibm.security.krb5.Krb5Debug	off	
	☐	com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA	true	

Total 3

- JVM Custom property needed for every JVM in Cell (Dmgr, Node Agents, App Servers)
- Per APAR PK45677, Add JVM custom property:

  `com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA`
  with value of `true`

Enabling Java and WebSphere

- **Need to create hwcrypto.cfg file**
- **Suggested location is /opt/IBM/WebSphere/**
- **Customize contents with**
  - Unique token label (the one you specified on the pkcsconf –c 0 –l initialization)
  - Token slot number (the zero above, in this case)
- **Example on the next slide**

Contents sample hwcrypto.cfg

```
name = rgylxws8
library=/usr/lib/pkcs11/PKCS11_API.so64
description=custom
slotListIndex = 0
disabledMechanisms = {
CKM_MD5
CKM_SHA_1
CKM_MD5_HMAC
CKM_SHA_1_HMAC
CKM_SSL3_MASTER_KEY_DERIVE
CKM_SSL3_KEY_AND_MAC_DERIVE
CKM_SSL3_PRE_MASTER_KEY_GEN
}
```

- Token Label

- Slot number

Enabling Java and WebSphere

```
RGYLXWS8:/ # cd /opt/IBM/WebSphere/AppServer/java/jre/lib/security/
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # ls
US_export_policy.jar                 cacerts                 local_policy.jar-unrestricted
US_export_policy.jar-unrestricted    java.policy             local_policy.jar.original
US_export_policy.jar.orignal         java.security           trusted.libraries
blacklist                            local_policy.jar
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # cp java.security java.securi
RGYLXWS8:/opt/IBM/WebSphere/AppServer/java/jre/lib/security # vi java.security
```

- Java.security file must be customized
- Resides in /opt/IBM/WebSphere/AppServer/java/jre/lib/security
- IBMPKCS11Impl moved to the top of the list and the hwcrypto.cfg file referenced
- Its good to have backup copies just in case…

Enabling Java and WebSphere

- Original java.security

```
# List of providers and their preference orders (see above):
#
#security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.1=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.jsse.IBMJSSEProvider
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.security.cmskeystore.CMSProvider
security.provider.8=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
security.provider.9=com.ibm.security.sasl.IBMSASL
security.provider.10=com.ibm.xml.crypto.IBMXMLCryptoProvider
security.provider.11=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.12=org.apache.harmony.security.provider.PolicyProvider
```

Enabling Java and WebSphere

- Customized java.security
- hwcrypto.cfg line is wrapped as shown, but is a single line

```
#
#security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.1=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl    /opt/IBM/Web
Sphere/hwcrypto.cfg
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.jsse.IBMJSSEProvider
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.security.cmskeystore.CMSProvider
security.provider.8=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
security.provider.9=com.ibm.security.sasl.IBMSASL
security.provider.10=com.ibm.xml.crypto.IBMXMLCryptoProvider
security.provider.11=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.12=org.apache.harmony.security.provider.PolicyProvider
#
```

Enabling Java and WebSphere

- **The WebSphere cipher suite needs to be adjusted to include those which your hardware and software will service**
- **Older configurations might use AES 128 and/or Triple DES**
  - SSL_RSA_WITH_AES_128_CBC_SHA
  - SSL_RSA_WITH_3DES_EDE_CBC_SHA
- **Newer Configurations**
  - SSL_RSA_WITH_AES_256_CBC_SHA

- See ZSW03250-USEN-00 for a discussion of cipher support
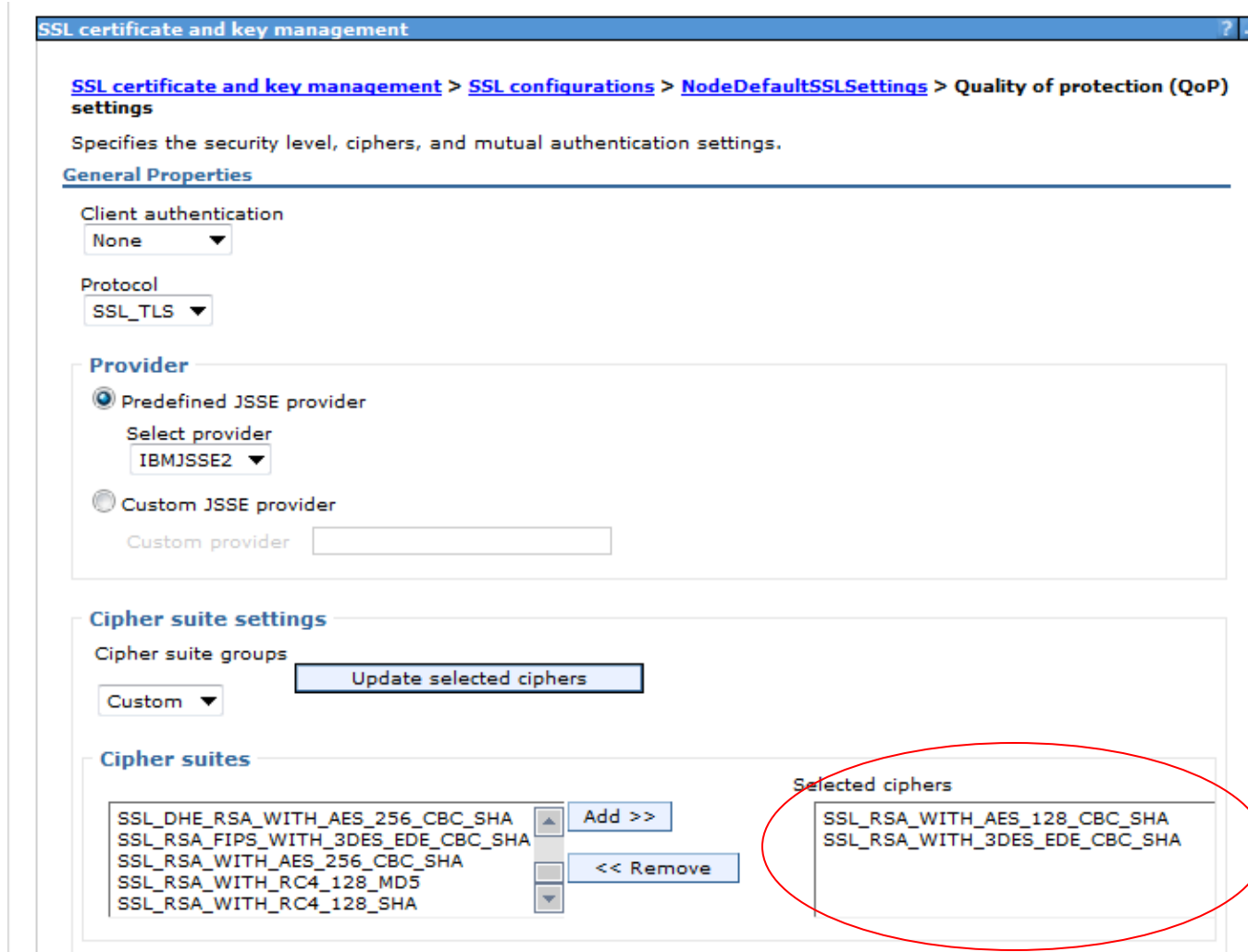
Enabling Java and WebSphere

- Original default ciphers

Enabling Java and WebSphere

- Customized high strength ciphers eligible for offload
- Repeat for other "SSL Configurations" as needed

Enabling Java and WebSphere

- The userid that will run the software using the pkcs11 cryptographic hardware must be added to the pkcs11 group
- In this case wasadmin is added to the pkcs11 group
- root is automatically added to this group when the pkcsslot daemon is started

```
rgylx001:/ # id
uid=0(root) gid=0(root) groups=0(root),64(pkcs11)
rgylx001:/ # usermod -G wasgroup,pkcs11 wasadmin
```

Enabling Java and WebSphere
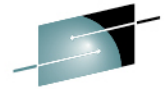
```
RGYLXWS8:/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin # ./startServer.sh ser
ver1
ADMU0116I: Tool information is being logged in file
           /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/startServer
.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 1374
RGYLXWS8:/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin # cat /proc/driver/z90
crypt

zcrypt version: 2.1.1
Cryptographic domain: 15
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 0
CEX3C count: 0
CEX3A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 1

Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A 7=CE
X3C 8=CEX3A
          0000000000000000 0000000000000000 0000800000000000 0000000000000000
```

Enabling Java and WebSphere

```
RGYLXWS8:/ # cat /proc/driver/z90crypt

zcrypt version: 2.1.1
Cryptographic domain: 15
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 0
CEX3C count: 0
CEX3A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 1


Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A 7=CE
X3C 8=CEX3A
          0000000000000000 0000000000000000 0000800000000000 0000000000000000


Waiting work element counts
          0000000000000000 0000000000000000 0000000000000000 0000000000000000


Per-device successfully completed request counts
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 0000001B 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Enabling Java and WebSphere

```
RGYLXWS8:/ # icastats
 function | # hardware | # software
----------+------------+-----------
    SHA-1 |         12 |          0
  SHA-224 |          0 |          0
  SHA-256 |          0 |          0
  SHA-384 |          0 |          0
  SHA-512 |          0 |          0
   RANDOM |        133 |          0
 MOD EXPO |          6 |          1
  RSA CRT |         20 |          0
  DES ENC |          0 |          0
  DES DEC |          0 |          0
 3DES ENC |          0 |          0
 3DES DEC |          0 |          0
  AES ENC |         37 |          0
  AES DEC |         22 |          0
 CMAC GEN |          0 |          0
 CMAC VER |          0 |          0
```

icastats – Part of libica V2 package. Tracks hardware and software requests through the libica package and allows you to understand how many request are performed in hardware vs software

SHARE
in Anaheim

Enabling Java and WebSphere

- **Don't forget to enable your deployment manager and node agents**

- **Remember to reapply customizations after apply maintenance**

- **Minimum levels:**
  - SLES 10 SP3
  - RHEL 5.5
  - WAS 7.0.0.9
  - SDK 1.6 SR7
  - Shared CEX2C device, z/VM APAR VM64727

Sample Java JCE Application

- Modified to encrypt text 1000 times to clearly show in icastats
- Utilizes previously defined hwcrypto.cfg and modifed java.security file from previous WebSphere example
- Sample runs with and without crypto hardware WITHOUT modification

Sample Java JCE Application

```
class JCEtestz {
public static void main (String[] args)
{
  SecretKey aesKey = null;
  try { // create random AES key
     KeyGenerator keygen =
     KeyGenerator.getInstance("AES");
     aesKey = keygen.generateKey();
  } catch (Exception e){e.printStackTrace(); }
  Cipher aesCipher;
  try { // Create the cipher
      aesCipher =
   Cipher.getInstance("AES/ECB/NoPadding");
     // Initialize the cipher for encryption
     aesCipher.init(Cipher.ENCRYPT_MODE,
   aesKey);
     // Our cleartext
   String str = "Can you read me now?        ";
      byte[] cleartext = str.getBytes();
```

```
   new String(cleartext);
   System.out.println(new String(cleartext) );
   byte[] ciphertext = null;
   //Encrypt the cleartext
   for(int i=0; i<1000; i++){
      ciphertext = aesCipher.doFinal(cleartext);
   }

   System.out.println(new String(ciphertext) );
   //Initialize the same cipher for
   //decryption
   aesCipher.init(Cipher.DECRYPT_MODE,
    aesKey);
   //Decrypt the ciphertext
   byte[] cleartext1 =
    aesCipher.doFinal(ciphertext);
   //Print cleartext1
   System.out.println(new String(cleartext1) );
} catch (Exception e) { e.printStackTrace(); }
System.out.println("Done!");
 }
}
```

Sample Java JCE Application

```
  DES ENC |           0 |           0
  DES DEC |           0 |           0
 3DES ENC |           0 |           0
 3DES DEC |           0 |           0
  AES ENC |        1082 |           0
  AES DEC |         339 |           0
 CMAC GEN |           0 |           0
 CMAC VER |           0 |           0
RGYLXWS8:~ # java -cp ./JCEtestz.jar com.ibm.lbs.JCEtestz
Can you read me now?
ï¿½ï¿½8ï¿½wï¿½ï¿½ï¿½Tï¿½6Ù°ï¿½  ï¿½ï¿½ï¿½
Can you read me now?
Done!
RGYLXWS8:~ # icastats
 function | # hardware | # software
----------+------------+------------
    SHA-1 |      756704 |           0
  SHA-224 |           0 |           0
  SHA-256 |         320 |           0
  SHA-384 |           0 |           0
  SHA-512 |           0 |           0
   RANDOM |         193 |           0
 MOD EXPO |          93 |          37
  RSA CRT |          24 |           0
  DES ENC |           0 |           0
  DES DEC |           0 |           0
 3DES ENC |           0 |           0
 3DES DEC |           0 |           0
  AES ENC |        2082 |           0
  AES DEC |         340 |           0
```

Agenda

1   zEnterprise Crypto Hardware Background

2   Making the Cryptographic Hardware Available to Linux

3   Enabling Linux to use the Hardware

4   Enabling Java and WebSphere to Exploit the Crypto Hardware

5   **Configuring the IBM HTTP Server to use the Crypto Hardware**

6   Enabling the WAS Plugin to Use the Crypto Hardware

7   openSSL and openSSH

8   In Kernel Crypto and DM-Crypt

Enabling the IBM HTTP Server

- **Other HTTP servers can be enabled but the steps are different**
- **Required steps:**
  - Install HTTP Server and current fixpack
  - Generate the a certificate and key pair
  - Create a Stash file for the User PIN
  - Add IHS user to the PKCS11 groups
  - Update the httpd.conf
  - Restart the server

Enabling the IBM HTTP Server

- **My example environment consisted of**
  - SLES 11 SP2+
  - IHS 8.0
  - IHS Java 8.0
- **IHS 6.1 will perform Asymmetric key encryption with the Crypto Express (CEX)**
  - IHS V6.1 has added support for using the CPACF hardware in APAR PK93112
- **IHS 7.0 and 8.0 can perform Asymmetric encryption via CEX as well as Symmetric key encryption via the CPACF**
- **Remove gskikm.jar from /opt/IBM/HTTPServer/java/jre/lib/ext** (Only for V7, NOT V8)

Enabling IHS –Certificate Generation

- Be sure to apply IHS fixpack
- Apply WAS SDK fixpack to IHS also.
- For V7 use ikeyman or gsk7cmd/gsk7cmd_64 to generate certificates, used gskcmd for V8
- For V7 use the 64 bit gsk7cmd_64 command with 64 bit crypto libraries, gskcmd is always 64 bit with V8

```
gskcmd –cert  –create –crypto
  /usr/lib/pkcs11/PKCS11_API.so64 –tokenlabel
  rgylxws8  –pw 88888888 –size 1024 –dn
  "CN=rgylxws8.pdl.pok.ibm.com, O=IBM, OU=LBS,
  ST=New York, C=US" –label lbstest –expire 7300
```

Enabling the IHS – Sample Script V7

```
#/bin/sh
exportPATH=/opt/IBM/HTTPServer/gsk7_64/bin:/opt/IBM/HTTPServer/bi
   n:/opt/IBM/HTTPServer/java/jre/bin:$PATH
export JAVA_HOME=/opt/IBM/HTTPServer/java/jre/

echo "!!!! gskikm.jar must be removed from the java path !!!"
gsk7cmd_64 -version
gsk7cmd_64  -keydb -create -db /opt/certs/dummy.kdb -pw zlinux -
   type cms -expire 7300  -stash
echo "Listings certs in the pkcs11 crypto"
gsk7cmd_64  -cert -list all  -crypto
   /usr/lib/pkcs11/PKCS11_API.so64 -tokenlabel rgylx001 -pw
   11111111
mkdir /opt/certs
/opt/IBM/HTTPServer/bin/sslstash -c /opt/certs/pkcs11.sth crypto
   11111111
chmod 700 /opt/certs/pkcs11.sth
echo "Createing new self signed certifcate"
gsk7cmd_64 -cert  -create -crypto /usr/lib/pkcs11/PKCS11_API.so64
   -tokenlabel rgylx001  -pw 11111111 -size 1024 -dn
   "CN=rgylx001.ibm.com, O=IBM, OU=LBS, ST=New York, C=US" -label
   lbstest -expire 7300
```

Enabling the IHS – Sample Script V8

```
#/bin/sh
export
  PATH=/opt/IBM/HTTPServer/gsk8/bin:/opt/IBM/HTTPServer/bin:/opt/I
  BM/HTTPServer/java/jre/bin:$PATH
export JAVA_HOME=/opt/IBM/HTTPServer/java/jre/
gsk8cmd -version
gsk8cmd  -keydb -create -db /opt/certs/dummy.kdb -pw zlinux -type
  cms -expire 7300  -stash
echo "Listings certs in the pkcs11 crypto"
gsk8cmd   -cert -list all  -crypto /usr/lib/pkcs11/PKCS11_API.so64
  -tokenlabel rgylxws8 -pw 88888888
mkdir /opt/certs
/opt/IBM/HTTPServer/bin/sslstash -c /opt/certs/pkcs11.sth crypto
  88888888
chmod 700 /opt/certs/pkcs11.sth
echo "Createing new self signed certifcate"
gsk8cmd -cert  -create -crypto /usr/lib/pkcs11/PKCS11_API.so64 -
  tokenlabel rgylxws8  -pw 88888888 -size 1024 -dn
  "CN=rgylxws8.pdl.pok.ibm.com, O=IBM, OU=LBS, ST=New York, C=US"
  -label lbstest -expire 7300
```

- No jar removal

- New gsk command

Enabling IHS –Certificate Generation

- Utilizing a script for certificate generation can simplify and automate the process
- Allows an easy way to test certificate management after every fix/fixpack is applied
- No gui required
- Very repeatable

Enabling IHS –Stash file for User PIN

- Crypto user PIN required and provided via "stash file"
- Can imbed in your certificate generation script
- Below 11111111 is the "user PIN" in the example

```
# mkdir /opt/certs
/opt/IBM/HTTPServer/bin/sslstash -c
  /opt/certs/pkcs11.sth crypto  11111111
# chmod 700 /opt/certs/pkcs11.sth
```

Enabling the IHS – SSLVirtual Host

```
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
Listen 443
<VirtualHost *:443>
    SSLEnable
    SSLProtocolDisable SSLv2
    ServerName rgylxws8.pdl.pok.ibm.com
    SSLCipherSpec 3A
    DocumentRoot /opt/IBM/HTTPServer/htdocs
    KeyFile /opt/certs/dummy.kdb                       ◄─────────────  Required
    SSLServerCert rgylxws8:lbstest
    SSLStashfile /opt/certs/pkcs11.sth
    SSLPKCSDriver /usr/lib/pkcs11/PKCS11_API.so
############################
# Symmetric offload  (required with older gskit)
    SSLAttributeSet 417 549
##########################
</VirtualHost>
```

Enabling the IBM HTTP Server

```
# usermod -G nobody,nogroup,pkcs11 nobody
usermod: `nobody' is primary group name.
```

Add appropriate libcrypto.so to the bottom of the httpd.conf
- `LoadFile /usr/lib64/libcrypto.so.0.9.8`

- **Restart Apache**
  - `/opt/IBM/HTTPServer/bin/apachectl restart`
- **Test https:// with your favorite browser**

Enabling the IBM HTTP Server

- `"Cryptographic token initialization failed. Cryptographic token support will not be available."`
- **Several possible causes**
  - pkcsconf –t does not show flag 0x44D
  - For V7, using 32 bit gsk7cmd with 64bit PKCS11_API.so64 ?
  - Pointing to a token label other than the one you initialized (the examples here use rgylxws8)
- `[crit] Error 430 initializing SSL environment, aborting startup`

- `[error] SSL0153E: Initialization error, The PKCS#11 driver failed to find the token specified by the caller. Configuration Failed`

  - Incorrect token/label in httpd.conf
  - Missing Loadfile for libcrypto.so.xxx in httpd.conf

Enabling the IBM HTTP Server

```
# tail ../logs/error_log
[Tue Sep 13 10:48:45 2011] [error] [client 172.110.101.6]
   [5e0440] [23788] SSL0209E: SSL Handshake Failed, ERROR
   processing cryptography. [172.110.101.6:50480 ->
   172.110.100.15:443] [10:48:45.000019752]
[Tue Sep 13 10:48:45 2011] [error] [client 172.110.101.6]
   [5e0440] [23788] SSL0209E: SSL Handshake Failed, ERROR
   processing cryptography. [172.110.101.6:50481 ->
   172.110.100.15:443] [10:48:45.000674329]
```

- Could mean the userid IHS is running under is not part of the PKCS11 group

Enabling the IHS - Success
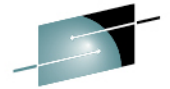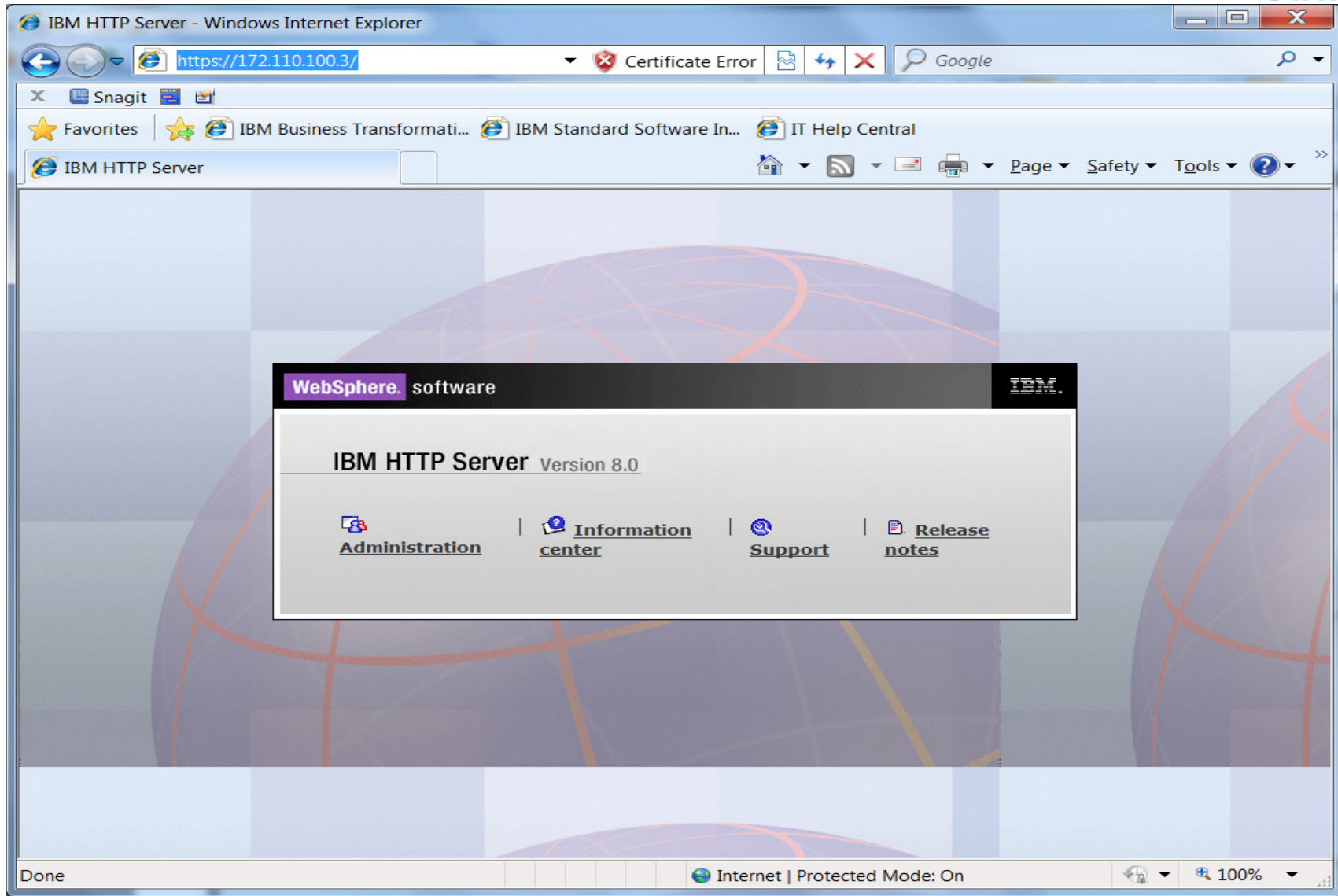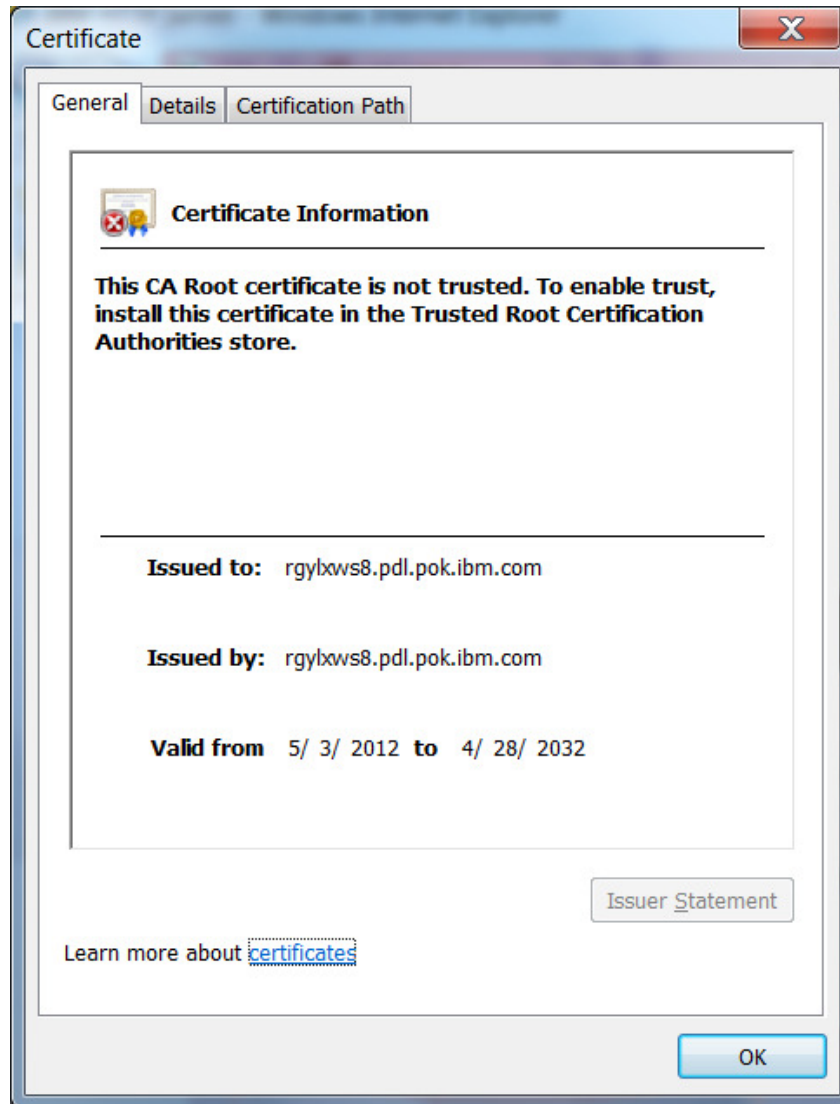
Enabling the IHS

- Confirming the usage of the certificate generated from gskcmd

Enabling the IBM HTTP Server

```
RGYLXWS8:/opt/IBM/HTTPServer/conf # icastats
 function | # hardware | # software
----------+------------+------------
    SHA-1 |         59 |          0
  SHA-224 |          0 |          0
  SHA-256 |          0 |          0
  SHA-384 |          0 |          0
  SHA-512 |          0 |          0
   RANDOM |        155 |          0
 MOD EXPO |         12 |          1
  RSA CRT |         32 |          0
  DES ENC |          0 |          0
  DES DEC |          0 |          0
 3DES ENC |        141 |          0
 3DES DEC |         63 |          0
  AES ENC |         37 |          0
  AES DEC |         22 |          0
 CMAC GEN |          0 |          0
 CMAC VER |          0 |          0
```

- icastats reporting crypto operations in hardware and no new software operations

Enabling the IBM HTTP Server

```
RGYLXWS8:/opt/IBM/HTTPServer/conf # cat /proc/driver/z90crypt
zcrypt version: 2.1.1
Cryptographic domain: 15
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 0
CEX3C count: 0
CEX3A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 4

Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A 7=CEX3C 8=CEX3A
          0000000000000000 0000000000000000 0000800000000000 0000000000000000
```

Enabling the IBM HTTP Server

```
Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A 7=CEX3C 8=CEX3A
           0000000000000000 0000000000000000 0000800000000000 0000000000000000
Waiting work element counts
           0000000000000000 0000000000000000 0000000000000000 0000000000000000


Per-device successfully completed request counts
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 0000002D 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Enabling Apache for Crypto

```
RGYLXWS8:/etc/apache2 # cat /proc/driver/z90crypt

zcrypt version: 2.1.1
Cryptographic domain: 3
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 0
CEX3C count: 0
CEX3A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 2


Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A 7=CEX3C 8=CEX3A
        0800000000000000 0000000000000000 0000000000000000 0000000000000000


Waiting work element counts
        0000000000000000 0000000000000000 0000000000000000 0000000000000000


Per-device successfully completed request counts
    00000000 0000005D 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Enabling Apache for Crypto

```
RGYLXWS8:/etc/apache2 # curl -k https://127.0.0.1
<html>
  <head>
    <title>Hello</title>
  </head>
  <body bgcolor=white>
        <h1>Hello</h1>

    <p>Hellow </p>

  </body>
</html>
RGYLXWS8:/etc/apache2 # cat /proc/driver/z90crypt
```

Enabling Apache for Crypto

```
RGYLXWS8:/etc/apache2 # cat /proc/driver/z90crypt

zcrypt version: 2.1.1
Cryptographic domain: 3
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 0
CEX3C count: 0
CEX3A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 2


Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A 7=CEX3C 8=CEX3A
          0800000000000000 0000000000000000 0000000000000000 0000000000000000


Waiting work element counts
          0000000000000000 0000000000000000 0000000000000000 0000000000000000


Per-device successfully completed request counts
    00000000 0000005D 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Enabling Apache for Crypto

- /etc/apache2/ssl-global.conf
- Add the SSLCryptoDevice ibmca directive
- Enables crypto express exploitation

```
# This global SSL configuration is ignored if
# "SSL" is not defined, or if "NOSSL" is defined.
<IfDefine SSL>
<IfDefine !NOSSL>
<IfModule mod_ssl.c>
    SSLCryptoDevice ibmca

    #
    #   Some MIME-types for downloading Certificates and CRLs
    #
    AddType application/x-x509-ca-cert .crt
    AddType application/x-pkcs7-crl    .crl

    #   Pass Phrase Dialog:
```

Enabling Apache for Crypto


```
RGYLXWS8:/etc/apache2 # rcapache2 restart
[Tue Aug 13 15:23:19 2013] [warn] NameVirtualHost RGYLXWS8:80 has no VirtualHosts
Syntax OK
Shutting down httpd2 (waiting for all children to terminate)                          done
Starting httpd2 (prefork) [Tue Aug 13 15:23:19 2013] [warn] NameVirtualHost RGYLXWS8:80 has no VirtualHosts
                                                                                     done

RGYLXWS8:/etc/apache2 # cat /proc/driver/z90crypt

zcrypt version: 2.1.1
Cryptographic domain: 3
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 0
CEX2A count: 0
CEX3C count: 0
CEX3A count: 1
requestq count: 0
pendingq count: 0
Total open handles: 3
```

Enabling Apache for Crypto

```
RGYLXWS8:/etc/apache2 # curl -k https://127.0.0.1
<html>
  <head>
    <title>Hello</title>
  </head>
  <body bgcolor=white>
```

```
pendingq count: 0
Total open handles: 3


Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A 7=CEX3C 8=CEX3A
          0800000000000000 0000000000000000 0000000000000000 0000000000000000


Waiting work element counts
          0000000000000000 0000000000000000 0000000000000000 0000000000000000


Per-device successfully completed request counts
    00000000 00000060 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Agenda

Enabling the IHS Plugin

- **PK96110 enables use of calls for CPACF**
- **Requires two custom properties in the plugin**
  - `SSLPKCSDriver`
  - `SSLPKCSPassword`

- **Web Servers > xxxxx > Plug- in Properties > Custom Properties**
- **Ensure PK82147 is applied**

Enabling the IHS Plugin

Cell=RGYLXWS8Cell01, Profile=Dmgr01

Web servers

Web servers > **webserver1** > **Plug-in properties** > **Custom properties**

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

⊞ Preferences

| New... | Delete |

Select	Name ⌄	Value ⌄	Description ⌄
You can administer the following resources:			
☐	CertLabel	rgylxws8	
☐	SSLConsolidate	true	
☐	SSLPKCSDriver	/usr/lib64/pkcs11/PKCS11_API.so	
☐	SSLPKCSPassword	/opt/certs/pkcs11.sth	
Total 4			

Enabling the IHS Plugin


```
<?xml version="1.0" encoding="ISO-8859-1"?><!--HTTP server plugin config file for the webserver RGYLXWS8Cell01.RG
YLXWS8Node01.webserver1 generated on 2012.05.03 at 11:30:00 PM GST-->
<Config ASDisableNagle="false" AcceptAllContent="true" AppServerPortPreference="HostHeader" ChunkedResponse="fals
e" FIPSEnable="false" FailoverToNext="false" HTTPMaxHeaders="300" IISDisableNagle="false" IISPluginPriority="High
" IgnoreDNSFailures="false" OS400ConvertQueryStringToJobCCSID="false" RefreshInterval="60" ResponseChunkSize="64"
 SSLConsolidate="true" SSLPKCSDriver="/usr/lib64/pkcs11/PKCS11_API.so" SSLPKCSPassword="/opt/certs/pkcs11.sth" Tr
ustedProxyEnable="false" VHostMatchingCompat="false">
    <Log LogLevel="Error" Name="/opt/IBM/WebSphere/Plugins/logs/webserver1/http_plugin.log"/>
    <Property Name="ESIEnable" Value="true"/>
    <Property Name="ESIMaxCacheSize" Value="1024"/>
    <Property Name="ESIInvalidationMonitor" Value="false"/>
    <Property Name="ESIEnableToPassCookies" Value="false"/>
    <Property Name="PluginInstallRoot" Value="/opt/IBM/WebSphere/Plugins/"/>
```

- plugin-cfg.xml in /opt/IBM/WebSphere/Plugins/config/webserver1
- Properties reside in the "Config" section of plugin-cfg.xml

Enabling the IHS Plugin

```
    ReplyInterval="60">
    <Server ConnectTimeout="5" ExtendedHandshake="false" MaxConnections="-1" Name="RGYLXWS8No
erIOTimeout="900" WaitForContinue="false">
        <Transport Hostname="RGYLXWS8." Port="9080" Protocol="http"/>
        <Transport Hostname="RGYLXWS8." Port="9443" Protocol="https">
            <Property Name="keyring" Value="/opt/IBM/WebSphere/Plugins/config/webserver1/plugin
            <Property Name="stashfile" Value="/opt/IBM/WebSphere/Plugins/config/webserver1/plug
            <Property Name="certLabel" Value="rgylxws8"/>
        </Transport>
    </Server>
```

Enabling the IHS and the Plugin

- **Summary**

  ➤ Try gskit commands instead of ikeyman. Scripting provides a more repeatable consistent process.

  ➤ IHS 6 can utilize CEX
     IHS V6.1 has added support for using the CPACF hardware in APAR PK93112

  ➤ With GSKIT 8, you do NOT remove the JAR like previous releases

Agenda

1	zEnterprise Crypto Hardware Background
2	Making the Cryptographic Hardware Available to Linux
3	Enabling Linux to use the Hardware
4	Enabling Java and WebSphere to Exploit the Crypto Hardware
5	Configuring the IBM HTTP Server to use the Crypto Hardware
6	Enabling the WAS Plugin to Use the Crypto Hardware
7	**openSSL and openSSH**
8	In Kernel Crypto and DM-Crypt

Linux on System z Crypto Stack

openssh ssh, sftp, scp	Apache (mod_ssl)	Apache (mod_nssl)	IBM c/c++ sw	Cust c/c++ PKCS11	WAS	Customer Java JCE	Customer CCA	Application

Standard Crypto Interfaces

- nss
- icc GSKIT
- JCA/JCE
- IBMPKCS11Impl
- openssl
  - Ibmca engine
- Opencryptoki pkcs11
  - ica token
  - cca token

System z HW crypto Libraries

- ica library
- cca library

Operating System

- ipsec
- dm-crypt
- Kernel crypto framework
- System z backend
- zcrypt device driver

Hardware

- CPACF des,3des,aes,sha,rng
- Accelerator rsa
- Co-processor rsa,rng,ecc

Clear	Protected	Securey

Complete your session evaluations online at www.SHARE.org/AnaheimEval

SHARE in Anaheim

The value of Open SSL and Hardware Crypto

Improvement due to avoided compression

CPU consumption
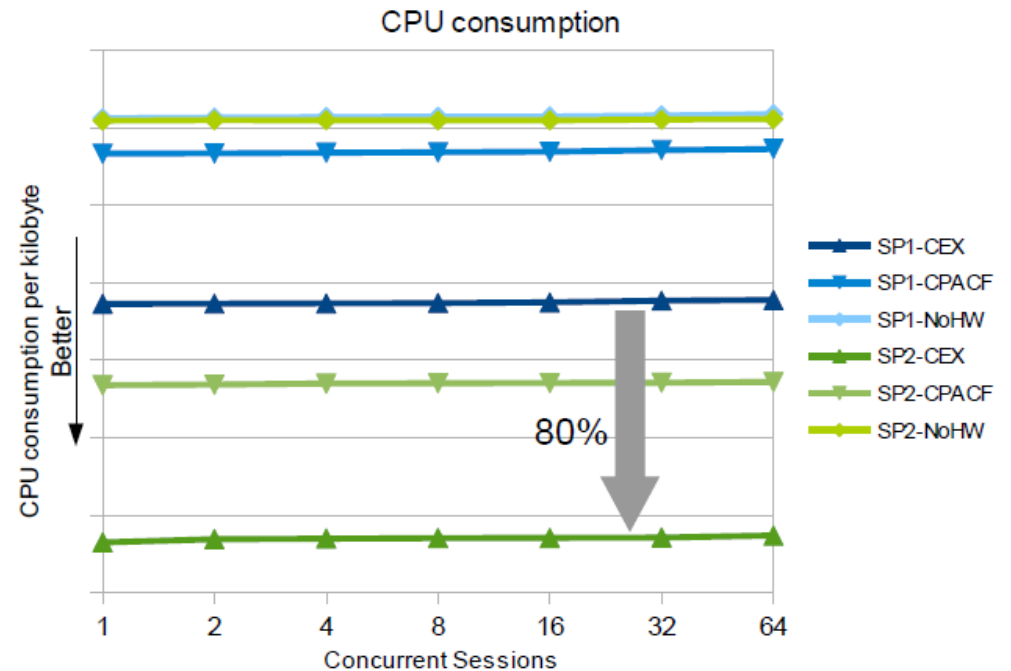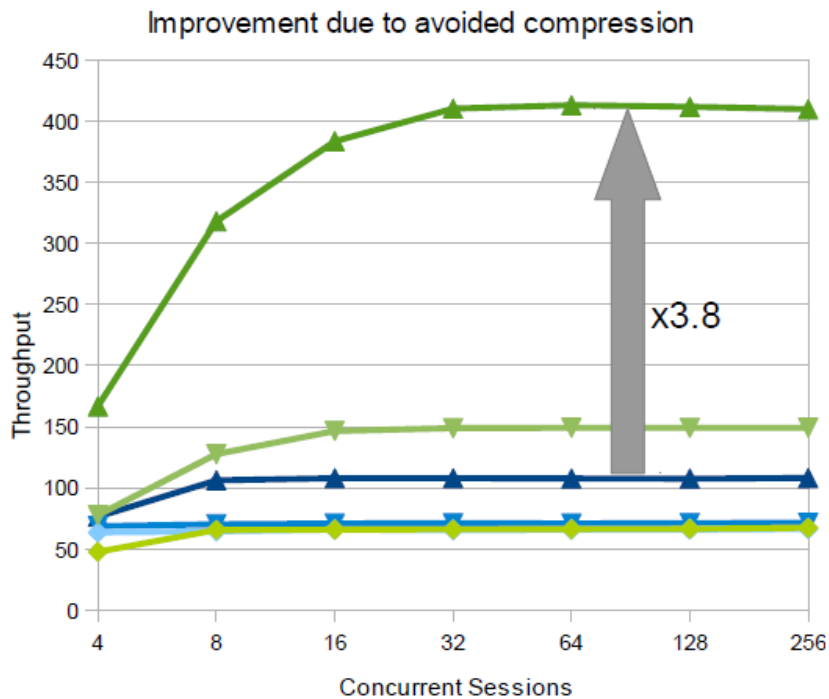
- Compressing the data to save cryptographic effort was the default for a while
  - Counter-productive as CPACF/CEX is so fast (and CEX account as off-loaded)
- Now it is possible to deactivate compression via an Environment variable

```
OPENSSL_NO_DEFAULT_ZLIB=Y
```

- 1000k payload cases w/CPACF and cards x3.8 faster now, still x2.3 without CEX cards
- Even 40b payload cases still show 15% throughput improvement
- Additionally depending on the setup 50% to 80% less cpu per transferred kilobyte

Enabling openssl & openssh use of Hardware Crypto

- **ssh, sftp, and scp can benefit from the acceleration**
    - Increased speed and reduced CPU consumption

- **Consider configuring on your Linux master images for all guests**
- **Implementation steps are simple**

    Step 1 – Ensure machine has the CPACF feature code enabled

    Step 2 – Install openssl-ibmca
    ```
    # zypper in openssl-ibmca
    ```

    Step 3 – Update /etc/ssl/openssl.cnf

(next page)

Enabling openssl & openssh use of Hardware Crypto

```
RGYLXWS8:/etc/ssl # rpm -ql openssl-ibmca
/usr/lib64/engines/libibmca.so
/usr/share/doc/packages/openssl-ibmca
/usr/share/doc/packages/openssl-ibmca/README
/usr/share/doc/packages/openssl-ibmca/openssl.cnf.sample
```

- The openssl.cnf.sample gets appended to the /etc/ssl/openssl.cnf
- The first line of the sample file is added to the top of openssl.cnf
- The rest is added to the bottom of openssl.cnf
- When completed, validate it is now active

```
RGYLXWS8:/etc/ssl # openssl engine
(dynamic) Dynamic engine loading support
(ibmca) Ibmca hardware engine support          ⟵
```

Immediately start exploiting the crypto hardware

```
RGYLXWS8:~ # sftp ryoung1@172.110.101.22
Connecting to 172.110.101.22...
Password:
sftp> put testfile


RGYLXWS8:~ # icastats
 function | # hardware | # software
----------+------------+------------
    SHA-1 |       3618 |          0
  SHA-224 |          0 |          0
  SHA-256 |         60 |          0
  SHA-384 |          0 |          0
  SHA-512 |          0 |          0
   RANDOM |          9 |          0
 MOD EXPO |          3 |          6
  RSA CRT |          2 |          0
  DES ENC |          0 |          0
  DES DEC |          0 |          0
 3DES ENC |          0 |          0
 3DES DEC |          0 |          0
  AES ENC |     163452 |          0
  AES DEC |     399796 |          0
 CMAC GEN |          0 |          0
 CMAC VER |          0 |          0
```

Agenda

1 zEnterprise Crypto Hardware Background

2 Making the Cryptographic Hardware Available to Linux

3 Enabling Linux to use the Hardware

4 Enabling Java and WebSphere to Exploit the Crypto Hardware

5 Configuring the IBM HTTP Server to use the Crypto Hardware

6 Enabling the WAS Plugin to Use the Crypto Hardware

7 openSSL and openSSH

8 **In Kernel Crypto and DM-Crypt**

SHARE
in Anaheim

Linux on System z Crypto Stack

openssh ssh, sftp, scp	Apache (mod_ssl)	Apache (mod_nssl)	IBM c/c++ sw	Cust c/c++ PKCS11	WAS	Customer Java JCE	Customer CCA	**Application**

nss

icc GSKIT

JCA/JCE

IBMPKCS11Impl

Standard Crypto Interfaces

openssl

Opencryptoki pkcs11

lbmca engine

ica token

cca token

System z HW crypto Libraries

ica library

cca library

Operating System

ipsec	dm-crypt

Kernel crypto framework

System z backend

zcrypt device driver

Hardware

CPACF
des,3des,aes,sha,rng

Accelerator rsa

Co-processor rsa,rng,ecc

Clear	Protected	Securey

Complete your session evaluations online at www.SHARE.org/AnaheimEval

Encrypting Filesystems

- Utilizes "In Kernel Crypto"
- Load required kernel modules for hardware based encryption

```
# modprobe des_s390
# modprobe sha1_s390
# modprobe sha256_s390
# modprobe aes_s390
# modprobe sha512_s390
```

- Setup dmcrypt as normal

Encrypting Filesystems

```
RGYLXWS8:~ # cryptsetup luksFormat /dev/SYSTEM/LVCRYPT
WARNING!
========
This will overwrite data on /dev/SYSTEM/LVCRYPT irrevocably.
Are you sure? (Type uppercase yes): YES
Note: make sure keyboard layout and encoding here matches
the intended environment for unlocking the volume
Enter LUKS passphrase:
Verify passphrase:
Command successful.

RGYLXWS8:~ # cryptsetup luksOpen /dev/SYSTEM/LVCRYPT lvcryptfs
Enter LUKS passphrase:
key slot 0 unlocked.
Command successful.

RGYLXWS8:~ # mkfs -t ext3 /dev/mapper/lvcryptfs
RGYLXWS8:~ # mount /dev/mapper/lvcryptfs  /mnt
RGYLXWS8:~ # cryptsetup status lvcryptfs
/dev/mapper/lvcryptfs is active:
  cipher:  aes-cbc-essiv:sha256
  keysize: 128 bits
  device:  /dev/dm-4
  offset:  1032 sectors
  size:    2096120 sectors
  mode:    read/write
```

Encrypting Filesystems – Enable Automatic Mount

```
RGYLXWS8:~ # cat /etc/crypttab
<target device>   <source device>   <key file>
lvcryptfs   /dev/SYSTEM/LVCRYPT   /etc/keyfile.key   luks

RGYLXWS8:~ # cat /etc/fstab
/dev/disk/by-path/ccw-0.0.0200-part1 /   ext3   acl,user_xattr   1 1
proc   /proc   proc   defaults   0 0
sysfs   /sys   sysfs   noauto   0 0
debugfs   /sys/kernel/debug   debugfs   noauto   0 0
devpts   /dev/pts   devpts   mode=0620,gid=5   0 0
/dev/SYSTEM/LVOPT   /opt   ext3   acl,user_xattr   1 2
/dev/SYSTEM/LVVAR   /var   ext3   acl,user_xattr   1 2
/dev/mapper/lvcryptfs /opt/crypt   ext3   auto   1 2

RGYLXWS8:~ # echo -n 'topsecret' > /etc/keyfile.key
RGYLXWS8:~ # chkconfig boot.crypto-early on
RGYLXWS8:~ # chkconfig boot.crypto on
```

# References
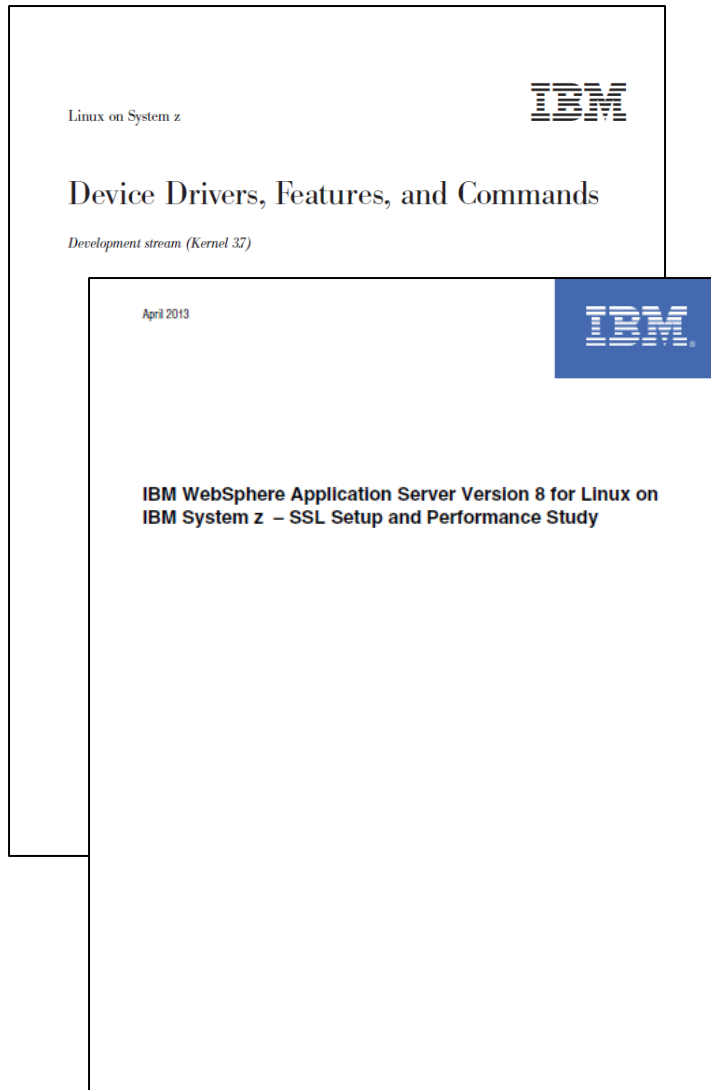
- **Linux on System z Device Drivers, Features, and Commands**
  SC33-8411-18
  http://www.ibm.com/developerworks/linux/linux390/documentation_dev.html

- **IBM WebSphere Application Server Version 8 for Linux IBM System z - SSL Setup and Performance Study**
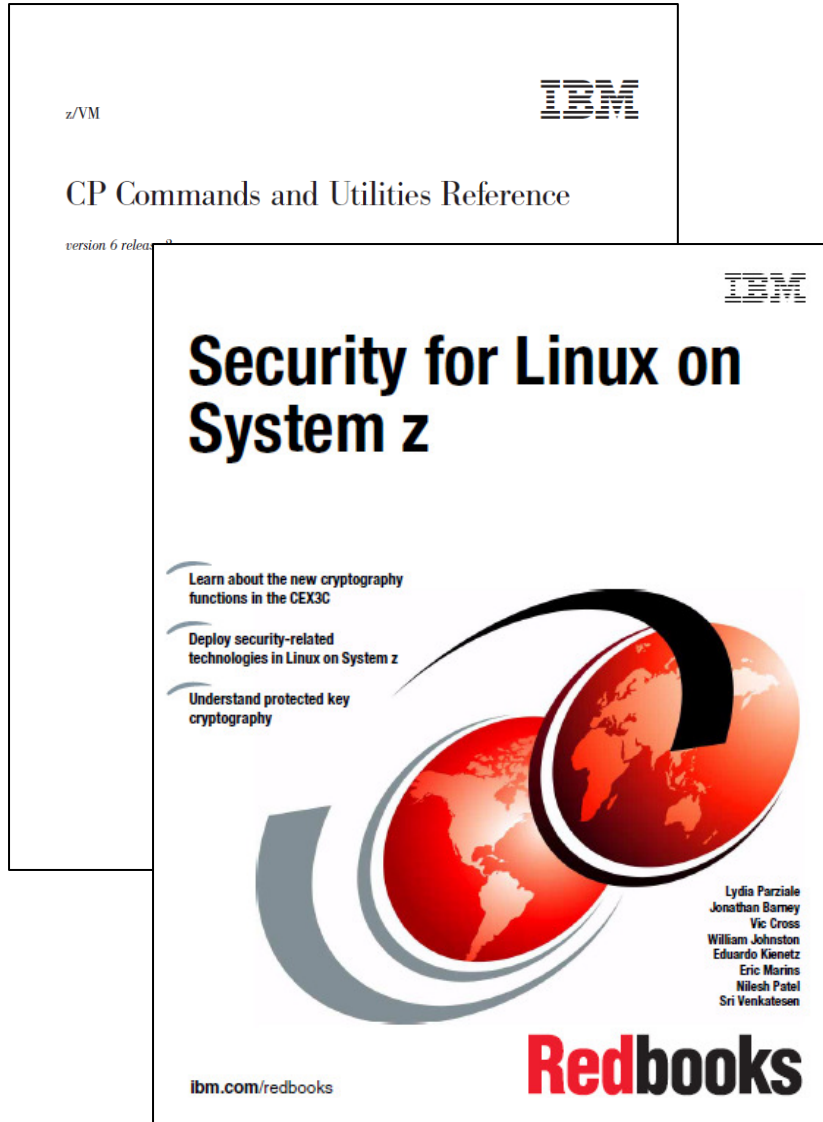  ZSW03250-USEN-00
  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102274

# References

z/VM

IBM

CP Commands and Utilities Reference

version 6 release

IBM

## Security for Linux on System z

- Learn about the new cryptography functions in the CEX3C
- Deploy security-related technologies in Linux on System z
- Understand protected key cryptography

Lydia Parziale
Jonathan Barney
Vic Cross
William Johnston
Eduardo Kienetz
Eric Marins
Nilesh Patel
Sri Venkatesen

**Redbooks**

ibm.com/redbooks

- • z/VM CP Commands and Utilities Reference

  *SC24-6175-01*

- • Security for Linux on System z

  SG24-7728

SHARE
in Anaheim

# References

- **Developerworks - Linux on System z Cryptographic Support**
- **ikeyman & gsk7cmd "Must Gather"**
- **IHSDIAG Crypto Hardware FAQ**
- **WAS Techdoc - Enabling and Configuring Cryptographic Technology**

SHARE
in Anaheim

# Thank you for attending

## Please remember to fill out your session evaluations

**Richard G. Young**

*Executive I.T. Specialist*

*IBM STG Lab Services*

*Virtualization & Linux on
zEnterprise  Team Lead*

IBM

*777 East Wisconsin Ave*

*Milwaukee, WI  53202*

*Tel 262 893 8662*

*Email: ryoung1@us.ibm.com*

SHARE
in Anaheim