Db2 Night Show

IDUG

Db2 Security Best Practices Volume III+

Dave Beulke,
Dave Beulke @ Associates,
Division of Pragmatic Solutions, Inc.
Twitter (@DBeulke)
LinkedIn (www.linkedin.com/in/davebeulke)

Cross Platform

Dave@davebeulke.com

- Lifetime IBM Db2 Information Champions
- One of 40 IBM Db2 Gold Consultant Worldwide
- President of DAMA-NCR
- Past President of International Db2 Users Group - IDUG
- Best speaker at CMG conference & former TDWI instructor
- Former Co-Author of certification tests
  - Db2 DBA Certification tests
  - IBM Business Intelligence certification test
- Former Columnist for IBM Data Management Magazine
- Extensive experience in Big Data systems, DW design and performance
  - Working with Db2 on z/OS since V1.2
  - Working with Db2 on LUW since OS/2 Extended Edition
  - Designed/implemented first data warehouse in 1988 for E.F. Hutton
  - *Syspedia* for data lineage and data dependencies since 2001 –
    - Find, understand and integrate your data faster!

Proven Performance Tips:
www.DaveBeulke.com

- Consulting
  - Security Audit & Compliance
  - Db2 Performance Review
  - CPU MLC Demand Reduction
  - Analytics & Database Design Review
  - Db2 12 Migration Assistance
  - Java Application Development assistance

- Educational Seminars
  - Java Security for Application Developers
  - Db2 Performance for Java Developers
  - Db2 Version 12
  - Analytic Designs for Performance
  - How to Do a Performance Review

I am honored to have been a presenter at 30+ years of Db2 conferences

2021 – Java Db2 Performance Best Practices
Security Best Practices Volume III
2020 - SQL Performance for Big Data
2019 - Best Design and Performance Practices for Analytics
2018 – Philadelphia - Security Best Practices Volume II
-Best Design and Performance Practices for Analytics
2017 – Anaheim -Understand IDAA Performance and Justify an IDAA Appliance
2016 – Austin Performance Enterprise Architectures for Analytic Design Patterns
How to do your own Db2 Security Audit
2015 - Valley Forge Db2 Security Practices
Big Data Performance Analytics Insights
2014 – Phoenix Big Data SQL Considerations
2013 – Orlando Big Data Disaster Recovery Performance
2012 – Denver Agile Big Data Analytics
2011 – Anaheim Db2 Temporal Tables Performance Designs
2010 - Tampa - Java Db2 Developer Performance Best Practices
2009 – Denver -Java Db2 Perf with pureQuery and Data Studio
Improve Performance with Db2 Version 9 for z/OS
2008 – Dallas - Java pureQuery and Data Studio Performance
2007 - San Jose - Developing High Performance SOA Java Db2 Apps
Why I want Db2 Version 9
2006 - Tampa - Class - How to do a Db2 Performance Review
Db2 Data Sharing
Data Warehouse Designs for Performance
2005 – Denver - High Performance Data Warehousing
2004 – Orlando – Db2 V8 Performance
President of IDUG
2003 - Las Vegas - Db2 UDB Server for z/OS V8 Breaking all the Limits
Co-author IBM Business Intelligence Certification Exam

2002 - San Diego - Db2 UDB for LUW 8 - What is new in Db2 Version 8
Data Warehouse Performance
2001 – Orlando -Data Sharing Recovery Cookbook
Designing a Data Warehouse for High Performance
Co-authored the first IBM Db2 z/OS Certification Exam
2000 – Dallas - Db2 Data Warehouse Performance Part II
1999 – Orlando - Store Procedures & Multi-Tier Performance
Developing your Business Intelligence Strategy
Evaluating OLAP Tools
1998 - San Francisco - Db2 Version 6 Universal Solutions
Db2 Data Warehouse Performance
Db2 & the Internet Part II
1997 – Chicago - Db2 & the Internet
1996 – Dallas- Sysplex & Db2 Data Sharing
Best Speaker Award at CMG Conference Mullen Award
1995 – Orlando - Practical Performance Tips
Improving Application Development Efficiency
1994 - San Diego - Database Design for Time Sensitive Data &
Guidelines for Db2 Column Function Usage
1993 – Dallas - High Availability Systems: A Case Study &
Db2 V3: A First-Cut Analysis
1992 - New York -Db2 –CICS Interface Tuning
1991 - San Francisco - Pragmatic Db2 Capacity Planning for DBAs
1990 – Chicago - Performance Implication of Db2 Design Decisions
1989 – Chicago - Db2 Performance Considerations

In my *Security Volume I & II* speeches (found on my blog)

- 4 Aspects of security
- Physical infrastructure
- System infrastructure
- Database infrastructure connections
- Application infrastructure
- Understand your connections
- Secure the network connections
- Secure servers
- Encryption protocols, practices and options
- FIPS-140 Compliance
- Achieving FIPS 140 standards
- Protecting data at rest
- SQL for discovery of user permissions

- IDs used through your services
- Connection governance
- Standard and exception protocols
- Risk Assessment and Access Realization
- Security encryption long term commitment
- Defining a secure database
- Using database steganography
- Table steganography
- Column steganography
- Audit discovery for Db2 z/OS and Db2 LUW
- Audit SQL Db2 LUW
- Audit Research – SQL for z/OS

- User Ids research
- Understand the extend of the Ids in your system
- Determine risk of each user id
- Begin list to eliminate obsolete ids
- Ids by database and application cross reference
- Ideas for auditing your environment security
- Tighten up your infrastructure system definitions
- Governance Risk assessments are integrated into lifecycle
- **Old z/OS and LUW Security Audit SQL queries and scripts**

Security attacks aren’t going away

- From the DBIR Report- https://www.verizon.com/about/news/verizon-2021-data-breach-investigations-report
  - Data -83 contributors, 12 industries, and 3 world regions
    - Victims spanning 88 countries
  - The average data breach costs in 2021 is $4.24 million,
    - a 10% rise from 2020 findings.
- 95% of data breaches have human error as factor
  - IBM Report - https://www.ibm.com/security/data-breach

  Report analyzes 29,207 quality incidents, of which 5,258 were confirmed breaches
  - The top 5 most expensive data breach attack vectors in 2021 are:
  - Business email compromise - $5.01 million
  - Phishing - $4.65 million
  - Malicious insiders - $4.61 million
  - Social engineering criminal attacks - $4.47 million
  - Vulnerabilities in third-party software - $4.33 million
- Fines, penalties and losses associated with breaches skyrocket

Security is only as good as the weakest link in the chain

- ***Database security depends on many supporting technologies:***
  - The **host operating system(s)** – provides protection of the database, its configuration and data.
  - The **networks** – provides protections via network devices and applications.
  - **Cloud, Web and application servers** – provide the security framework for all the cloud interfaces, hosted web applications;
    - Connected world-These servers control access to other servers and applications that control others etc..
  - The **applications** – provides access to the data. If the application does not contribute to the security model, it can provide fully-privileged, un-audited access to the database and any data it connects to.

DBAs responsible for actions of many people

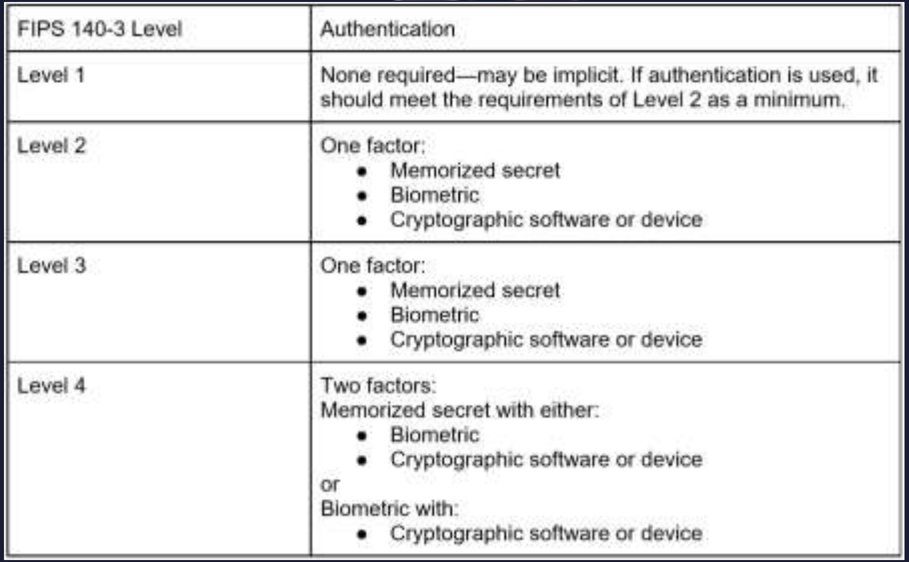
- Goes beyond a single operating system
  - System administrators – SYSADMs, SECADMs
  - DBAs – DBADMs
  - Operators – SYSOPR, job scheduling system
- Goes beyond a single application
  - So many processes, web services and RESTful interfaces
  - Variety of programming languages from many different decades
- How many levels of authorized application does your company have?
  - How many legacy levels of different programming languages does your infrastructure execute
- Welcome to the matrix of security complexity
  - Thousands/millions of security situations every minute!

What you can do as an Administrator

- Verify Authentication
  - REVOKE PUBLIC access against all tables!! ***Everywhere!!!***
  - Only have ROLE definitions – no explicit user ids anywhere in your system!

- Lock down Admin functionality
  - Understand your System and database level authorizations and configuration
  - Understand how ids and processes can expose or reference your data

- Encrypt your data!
  - All database table data is encrypted – all data at rest is encrypted
  - All system communications are encrypted
  - All interfaces use FIPS 140 – What Level do you use?



FIPS 140-3 Level	Authentication
Level 1	None required—may be implicit. If authentication is used, it should meet the requirements of Level 2 as a minimum.
Level 2	One factor: • Memorized secret • Biometric • Cryptographic software or device
Level 3	One factor: • Memorized secret • Biometric • Cryptographic software or device
Level 4	Two factors: Memorized secret with either: • Biometric • Cryptographic software or device or Biometric with: • Cryptographic software or device

What you can do as an Administrator

- Partner with your RACF & security administrators
  - Understand how your ROLE groups are defined
    - What criteria is used to separate ROLE definitions
  - Understand ALL Service Accounts or Job schedulers or defaults
    - Understand their boundaries and if/how they are logged
    - Know them across all your test, QA and production environments

- Security within your designs, definitions and application access
  - Consider Row and Column Access Control (RCAC) for PII data
  - Label-Based Access Control (LBAC) for different vendors access to same table but separate data
  - Confirm application connection configuration, usage and application id type used for access
  - Monitor SQL within the environment!! – Any rogue application SQL?
  - Use Steganography within your database and applications

Steganography - Hide data in plain sight!

- Steganography adds another layer to security
- Makes data useless for the hacker
  - Padding data
  - Data shift
  - Formulas - value added, subtracted and calculated
  - Scrambled, reversed or data combination of fields
- Legacy database tables/columns easily enhanced
  - No changes to database
  - Example: No changes to database or application
    - Social Security Number - 9 digits
    - Credit Card numbers
    - Any and all types of PII data can be hid through steganography!

Steganography – Obscure to be secured

- Introduce to the “SALT and PEPPER” your data concepts
- No changes to database - SSN 9 digits
- Example SSN is ‘123456789’

```
CREATE TABLE BEULKE.CUST1 (
FIRST_NAME    CHAR (25)        NOT NULL DEFAULT ' ',
LAST_NAME     CHAR (25)        NOT NULL DEFAULT ' ',
ADDRESS1      CHAR (25)        NOT NULL DEFAULT ' ',
ADDRESS2      CHAR (25)        NOT NULL DEFAULT ' ',
CITY          CHAR (25)        NOT NULL DEFAULT ' ',
STATE_CD      CHAR (2)         NOT NULL DEFAULT ' ',
ZIPCODE       CHAR (25)        NOT NULL DEFAULT ' ',
PHONE         DECIMAL (10,0)   NOT NULL DEFAULT 9999999999,
SSN          DECIMAL (9,0)  NOT NULL DEFAULT 99999999999,
CC_NBR        DECIMAL (16,0)   NOT NULL DEFAULT 9999999999999999,
CC_EXP        DECIMAL (4,0)    NOT NULL DEFAULT 9999,
CC_SCD        DECIMAL (3,0)    NOT NULL DEFAULT 999
REAL SSN = 123456789 and customer from STATE_CD ‘CA’ - California
```

- SALT and Pepper is data shift!

Steganography

- So INSERT ‘**456789123**’ into SSN since “CA”
- Application know the data shift based on State code
- EDITPROC, FIELDPROC, Store Procedure, Java Class

```
CREATE TABLE BEULKE.CUST1 (
FIRST_NAME     CHAR (25)         NOT NULL DEFAULT ' ',
LAST_NAME      CHAR (25)         NOT NULL DEFAULT ' ',
ADDRESS1       CHAR (25)         NOT NULL DEFAULT ' ',
ADDRESS2       CHAR (25)         NOT NULL DEFAULT ' ',
CITY           CHAR (25)         NOT NULL DEFAULT ' ',
STATE_CD       CHAR (2)          NOT NULL DEFAULT ' ',
ZIPCODE        CHAR (25)         NOT NULL DEFAULT ' ',
PHONE          DECIMAL (10,0)    NOT NULL DEFAULT 9999999999,
SSN           DECIMAL (9,0)   NOT NULL DEFAULT 99999999999,
CC_NBR         DECIMAL (16,0)    NOT NULL DEFAULT 9999999999999999,
CC_EXP         DECIMAL (4,0)     NOT NULL DEFAULT 9999,
CC_SCD         DECIMAL (3,0)     NOT NULL DEFAULT 999
REAL SSN = 123456789 and customer from STATE_CD ‘CA’ - California
```

- SALT and Pepper is data shift!

Steganography – New system secured

- Example #2 - Add more factors to hide your data
- Hide in plain sight
- Hashing, encoding, adding digit or values

```
CREATE TABLE BEULKE.CUST2 (
FIRST_NAME     CHAR (25)        NOT NULL DEFAULT ' ',
LAST_NAME      CHAR (25)        NOT NULL DEFAULT ' ',
ADDRESS1       CHAR (25)        NOT NULL DEFAULT ' ',
ADDRESS2       CHAR (25)        NOT NULL DEFAULT ' ',
CITY           CHAR (25)        NOT NULL DEFAULT ' ',
STATE_CD       CHAR (2)         NOT NULL DEFAULT ' ',
ZIPCODE        CHAR (25)        NOT NULL DEFAULT ' ',
PHONE          DECIMAL (10,0)   NOT NULL DEFAULT 9999999999,
SSN              DECIMAL (11,0)   NOT NULL DEFAULT 99999999999,
CC_NBR         DECIMAL (16,0)   NOT NULL DEFAULT 9999999999999999,
CC_EXP         DECIMAL (4,0)    NOT NULL DEFAULT 9999,
CC_SCD         DECIMAL (3,0)    NOT NULL DEFAULT 999
REAL SSN = 123456789
```

- Two bytes of overhead - SSN defined as 11 bytes

Steganography

- "SALT and PEPPER" your data
- Padded with an extra 4th & 6th digits before and after the real SSN
- So INSERT '**41234567896**' into 11 digit

```
CREATE TABLE BEULKE.CUST2 (
FIRST_NAME     CHAR (25)         NOT NULL DEFAULT ' ',
LAST_NAME      CHAR (25)         NOT NULL DEFAULT ' ',
ADDRESS1       CHAR (25)         NOT NULL DEFAULT ' ',
ADDRESS2       CHAR (25)         NOT NULL DEFAULT ' ',
CITY           CHAR (25)         NOT NULL DEFAULT ' ',
STATE_CD       CHAR (2)          NOT NULL DEFAULT ' ',
ZIPCODE        CHAR (25)         NOT NULL DEFAULT ' ',
PHONE          DECIMAL (10,0)    NOT NULL DEFAULT 9999999999,
SSN              DECIMAL (11,0)    NOT NULL DEFAULT 99999999999,
CC_NBR         DECIMAL (16,0)    NOT NULL DEFAULT 9999999999999999,
CC_EXP         DECIMAL (4,0)     NOT NULL DEFAULT 9999,
CC_SCD         DECIMAL (3,0)     NOT NULL DEFAULT 999
REAL SSN = 123456789
```

- SALT and Pepper the data

Does each application have a security risk rating?

- Techniques for applications
  - Insufficient logging and monitoring
    - Securing system and application logs and all debugging/audit information
      - Too much access to the logging within all the systems
  - Standard application error handling procedures and practices
    - Coding reviews for standard security techniques and practices
  - Poor connection, trust manager and certificate management controls
    - Architecture for always **secure** and **encrypted** communications
      - Stick with reference MVC architecture
  - Establish application security baseline
    - Establish **Security Audit Risk Rating** for each application!

*****WARNING*****

Default Access Cleaned Up

*****NOTE*** Analysis and testing should be done to determine the impact of any Db2 security *REVOKE* within your Db2 environment prior to execution!**

Remember authorizations cascade and REVOKE authorities cascade within your system.

Before revoking access *make sure to analyze and test the operations of your system*, its databases, the various utilities, and your application operations. Db2 security is very complex and inter-dependent.

One of the best ways to analyze the REVOKE impacts is by using a Redirected Restore of your Db2 system to a test environment where all the REVOKE statement cascades and impacts can be thoroughly tested and realized.

Another Db2 security REVOKE testing scenario can be analyzed and started in your test environment. Once the full impact of your Db2 Security REVOKEs are realized, they can be migrated into your other environments and finally your production environments.

Be careful!

Properly managed Db2 security is critical to protect your environment and to maintain an operational environment that is critical to your company's bottom line. This takes planning, testing and a detail security analysis!

*****WARNING*****

Connections and Services – Db2 LUW

- Discover these settings through Db2: `dbm` & `db cfg` commands
- Defaults used for the server, instance and database
- Shared Db2 Instance for Test and Production
  - Asking for trouble – what authorities are shared?
- Default TCP/IP within your environment
  - Where behind the company firewall is your database
- Default Port used for the Db2 Instance
  - Production available
  - Is any instance still using the default port?

Connections and Services – Db2 LUW

- Take care of default Instance Discovery Mode
  - Discovery Mode Let Db2 Instance to be discovered
    - "Hack Me" sign on your Db2 Instance

- Database Discovery Parameters
- **Db2 "GET DBM CFG"**
  - **Database Manager Configuration**

```
Discovery mode                        (DISCOVER) = SEARCH
Discover server instance         (DISCOVER_INST) = ENABLE
```

- **Db2 "GET DB CFG"**
  - **Database Configuration**

```
Discovery support for this database (DISCOVER_DB) = ENABLE
```

- <u>Disable</u> all of these instance Discovery Mode parameters!

Connections and Services – Db2 LUW

- Is encryption used within your connections?
  - Db2 Connect Version?
    - How up to date is your middleware?
  - Use AES encrypted Db2 Connect "`DATA_ENCRYPT`" settings
    - Encrypt ALL data on the wire between the Client and the Host database

- Are your connections using the Transport Layer Security (TLS)?
    - Provides security certificates on each end of your processing
    - Using compliant FIPS 140-2 interfaces?

- Is encryption used within your other environments?
  - All Test, QA and Performance databases also………

Connections and Services – Db2 LUW

- Easy process to take an existing old databases to encryption
  1. Take a non encrypted backup
     - Reduces encryption overhead on the backup and the restore

  2. Restore from the backup and use the new “Encrypt” clause
     - Take your database from exposed to encrypted
     - Legacy encryption used
     - There is a performance overhead for encryption
     - Some Server chips have encryption algorithms burned into them
       - Hardware assist with newer servers
  3. Encryption protects data at rest

Connections and Services – Db2 LUW

- Make your database backups encrypted
  - If your databases are already created
  - Encrypt with a specific key command is:

```
BACKUP DATABASE database_name
ENCRYPT ENCRLIB 'libdb2compr_encr.so'
ENCROPTS 'BEULKEKey=theBestEncryption';
```

← **Encrypted and Compressed**

  - Compress and Encrypt command is:

```
BACKUP DATABASE database_name
ENCRYPT ENCRLIB libdb2compr_encr.so;
```

- The Triple Data Encryption Standard (3DES) Is deprecated!
  - Recommend to use Advanced Encryption Standard (AES) native encryption option
  - AES is the new default

Connection and Services

- Trusted Context
  - WebSphere/web servers threads are created, authorized and de-allocated
  - Each Server for your system should have a ***<u>different name</u>***
  - Connection <u>PROPERTIES</u> should be unique and vary based on transaction type

```
CREATE TRUSTED CONTEXT my_trusted_context_name
       BASED UPON CONNECTION USING SYSTEM AUTHID user-id
       ATTRIBUTES (ADDRESS   ‘192.0.2.1’)
       DEFAULT ROLE trusted_role
      ENABLE
```

  - There are several steps for setting up Trusted Context
    - Follow the instruction from the Db2 SQL manuals and the information on IBM’s website at this URL:
    - SEARCH for “**Establishing an explicit trusted connection”** https://www.ibm.com/docs/en/db2/11.5?topic=connections-trusted-contexts-trusted

Four Levels of security

1. System-level authorization
   - SYSADM, SYSCTRL, SECADM, SYSMON
2. Database-level authorization
   - SECADM, DBADM, ACCESSCTRL, DATAACCESS, SQLADM
3. Object-level authorization
   - Object level authorization checks privileges when an operation is performed
4. Content-based authorization
   - Views provide a way to control which columns or rows accessible
   - Label-based access control (LBAC) determines read and write access to individual rows and individual columns

Default Access Cleaned Up? – Db2 LUW

- REVOKE PUBLIC Access
  - Limit discovery of your metadata
  - Start with TABSCHEMA
    - SYSCAT, SYSIBM, SYSIBMADM, and SYSTOOLS

```
SELECT 'REVOKE SELECT ON '|| <table schema name> ||'.'
||TABNAME||' FROM PUBLIC; '
FROM SYSCAT.TABLES
-- Start with SYSCAT%, SYSIBM%, SYSIBMADM%, SYSTOOLS% for your TABSCHEMA
WHERE TABSCHEMA LIKE '<table schema name>'
ORDER BY TABSCHEMA, TABNAME
WITH UR;
```

- REVOKE miscellaneous user access
  - Change TABSCHEMA to your databases' schema name
  - Eliminate all extra access authorities to your databases

Default Access Cleaned Up? – Db2 LUW

- Default database creation gives PUBLIC too much

-
  ```
  REVOKE BINDADD ON DATABASE FROM PUBLIC;
  REVOKE CREATETAB ON DATABASE FROM PUBLIC;
  REVOKE CONNECT ON DATABASE FROM PUBLIC;
  REVOKE IMPLICIT_SCHEMA ON DATABASE FROM PUBLIC;
  REVOKE ACCESSCTRL ON DATABASE FROM GROUP PUBLIC;
  REVOKE DATAACCESS ON DATABASE FROM GROUP PUBLIC;
  REVOKE DBADM ON DATABASE FROM GROUP PUBLIC;
  ```

-
  ```
  REVOKE ALTERIN ON SCHEMA <my schema name> FROM PUBLIC;
  REVOKE CREATEIN ON SCHEMA <my schema name> FROM PUBLIC;
  REVOKE DROPIN ON SCHEMA <my schema name> FROM PUBLIC;
  ```

-
  ```
  REVOKE EXTERNALROUTINE ON DATABASE FROM GROUP PUBLIC ;
  REVOKE LOAD ON DATABASE FROM GROUP PUBLIC ;
  REVOKE NOFENCE ON DATABASE FROM GROUP PUBLIC ;
  REVOKE QUIESCECONNECT ON DATABASE FROM GROUP PUBLIC ;
  REVOKE LIBRARYADM ON DATABASE FROM GROUP PUBLIC ;
  REVOKE SECURITYADM ON DATABASE FROM GROUP PUBLIC ;
  REVOKE SQLADM ON DATABASE FROM GROUP PUBLIC ;
  REVOKE WLMADM ON DATABASE FROM GROUP PUBLIC ;
  REVOKE EXPLAIN ON DATABASE FROM GROUP PUBLIC ;
  REVOKE CREATESECURE ON DATABASE FROM GROUP PUBLIC;
  ```

-
  ```
  REVOKE USE OF TABLESPACE USERSPACE1 FROM PUBLIC;
  ```

Default Access Cleaned Up? – Db2 LUW

- REVOKE Monitoring Tables access
  - Limit discovery of your metadata
  - Start with TABSCHEMA
    - SYSCAT, SYSIBM, SYSIBMADM, and SYSTOOLS
-
```
REVOKE SELECT ON MON_DB_SUMMARY from public
REVOKE SELECT ON MON_CONNECTION_SUMMARY from public
REVOKE SELECT ON MON_WORKLOAD_SUMMARY from public
REVOKE SELECT ON MON_SERVICE_SUBCLASS_SUMMARY from public
REVOKE SELECT ON MON_CURRENT_UOW from public
REVOKE SELECT ON MON_CURRENT_SQL from public
REVOKE SELECT ON MON_PKG_CACHE_SUMMARY from public
REVOKE SELECT ON MON_LOCKWAITS from public
REVOKE SELECT ON MON_TBSP_UTILIZATION from public
REVOKE SELECT ON MON_BP_UTILIZATION from public;
```

Audit your Db2 LUW system

- Authorizations within your system
  - Who has what
    - With what authority
  - Who granted it to them
  - What ROLES have access
    - What can that ROLE access
  - How are the SYSPROCs used

```
SELECT * FROM SYSCAT.SCHEMAAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT
CHAR(GRANTOR,35) AS GRANTOR,
CHAR(GRANTEE,35) AS GRANTEE,
GRANTEETYPE FROM SYSCAT.DBAUTH
WHERE SECURITYADMAUTH='Y'
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT
CHAR(GRANTOR,35) AS GRANTOR,
CHAR(GRANTEE,35) AS GRANTEE,
GRANTEETYPE FROM SYSCAT.ROLEAUTH
WHERE ROLENAME= '<search role name>'
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT *
FROM SYSCAT.ROUTINEAUTH
WHERE SPECIFICNAME LIKE 'AUDIT%'
  AND SCHEMA='SYSPROC'
FETCH FIRST 10 ROWS ONLY WITH UR;
--If the value of CONNECT_PROC is null
-- (i.e., not set), this is a finding.
```

Audit your Db2 LUW system

- OWNERS of your Objects
- Who owns what
- TS, Tables, Nicknames, Constraints and Sequences
- Who owns the Objects executing against your tables

```
SELECT TBSPACE,OWNER
FROM SYSCAT.TABLESPACES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT TABNAME,OWNER
FROM SYSCAT.TABLES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT TABNAME,OWNER
FROM SYSCAT.NICKNAMES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT CONSTNAME,OWNER
FROM SYSCAT.TABCONST
FETCH FIRST 10 ROWS ONLY WITH UR;;
```

```
SELECT SEQNAME,OWNER
FROM SYSCAT.SEQUENCES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT PKGNAME,OWNER
FROM SYSCAT.PACKAGES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT ROUTINENAME,OWNER
FROM SYSCAT.ROUTINES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT LIBNAME,OWNER
FROM SYSCAT.LIBRARIES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT MODULENAME,OWNER
FROM SYSCAT.MODULES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
SELECT TRIGNAME, OWNER
FROM SYSCAT.TRIGGERS
FETCH FIRST 10 ROWS ONLY WITH UR;
```

Audit your Db2 LUW system

- Know your containers & storage

```
SELECT
VARCHAR(CONTAINER_NAME,70) AS CONTAINER_NAME,
VARCHAR(TBSP_NAME,20) AS TBSP_NAME
FROM TABLE(MON_GET_CONTAINER('',-2))
FETCH FIRST 10 ROWS ONLY WITH UR;
```

- Container tablespace relationship
- Understand storage groups
  - Dependent storage drive

```
SELECT
VARCHAR(STORAGE_GROUP_NAME, 30) AS STOGROUP,
VARCHAR(DB_STORAGE_PATH, 40) AS STORAGE_PATH
FROM TABLE(ADMIN_GET_STORAGE_PATHS('',-1))
FETCH FIRST 10 ROWS ONLY WITH UR;
```

- More security queries within the Db2 LUW documentation
  - `https://www.ibm.com/docs/en/db2/11.5?topic=security-db2-model`

Proactive monitoring-Notification E-mail (SMTP) CRONTAB - Example:

```
db2 "select 'X1X', count(*) from syscat.TRIGGERS " | grep X1X | while read lit mycount
do
if [[ $mycount -gt 250 ]] then
print "There are more than 250 TRIGGERS in the DB2 Catalog! " | mail -s "DB2 LUW SEcurity Performance Alert"
myfirstname.mylastname@my.company.com
print "There are $mycount TRIGGERS in the DB2 Catalog "
else
print "There are $mycount TRIGGERS in the DB2 Catalog "
fi
done
db2 "select userid, appname, authid, appid, conntime, stoptime, fetchcount, rowsread, rowswritten, totalsorttime, cputime, sqlcode,'X1X'
from
S_ADAMS_XYZ900D6_SAMPLE. AUDIT_EMPLOYEE where ((rowsread > 1000) or (fetchcount > 1000) or (rowswritten > 1000)) and (appname not in
('DB2HMON','db2hmon')) order by appname, fetchcount, rowsread desc with ur" | grep X1X | \
while read myuserid myappname myauthid myappid myconntime mystoptime myfetchcount myrowsread myrowswritten mytotalsorttime mycputime
mysqlcode myX1X
do
if [[ $myrowsread -gt 1000 ]] then
print "This is an Expensive SQL stmt " | mail -s "DB2 Alert - Expensive rowsread SQL" myfirstname.mylastname@my.company.com
print "This is an Expensive SQL stmt Greater_than_1000_rowsread with rowsread as $myrowsread "
fi
if [[ $myfetchcount -gt 1000 ]] then
print "This is an Expensive SQL stmt!" | mail -s "DB2 Alert - Expensive fetchcount SQL" myfirstname.mylastname@my.company.com
print "This is an Expensive SQL stmt Greater_than_1000_fetchcount with fetchcount as $myfetchcount "
fi
if [[ $myrowswritten -gt 1000 ]] then
print "This is an Expensive SQL stmt!" | mail -s "DB2 Alert - Expensive rowswritten SQL" myfirstname.mylastname@my.company.com
print "This is an Expensive SQL stmt Greater_than_1000_rowswritten with rowswritten as $myrowswritten "
else
print "This is a SECURITY VIOLATION SQL stmt!" | mail -s "DB2 Alert – SECURITY VIOLATION SQL" myfirstname.mylastname@my.company.com
print "This is a SECURITY VIOLATION SQL stmt "
fi
done
db2 -v "terminate"
exit
```

Physical – Applications

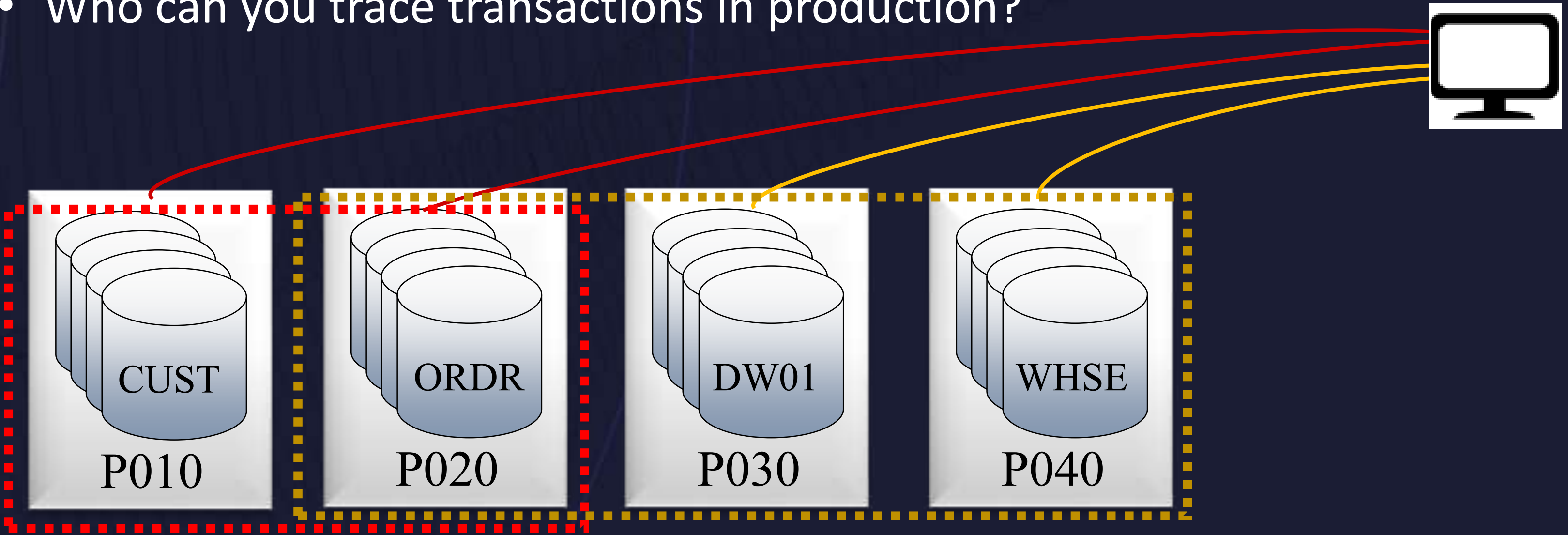
- **Hide _ALL_ Technology Versions**
- **OS, Web Server, TCPIP Address**
- **Disable Server Directory Listing**
- **Remove unnecessary server processes**
- **Separate Security Profile for each component and application**

- **Limit TimeOut Values - keep it low**
- **Limit # connections**
- **Limit Keep Alive Timeout short**
- **Limit Request Size**
- **Limit Reply size**
- **Limit # SQL for each connection/TX**

- **Trusted context connections only**
- **Remove *.NULLID defaults**
- **Remove default programs/classes**
- **Only approved services/programs**
- **Log & *analyze* denied access requests**
- **Only encrypted access**

How many virtual or direct connections

- Understand TCP/IP – DVIPA and-or VIPA settings
  - How different is Production versus Test?
    - Any Defaults used?
  - Sysplex dynamic routing turned on? To how many members?
  - Capturing transaction profiles?
    - Who can you trace transactions in production?



Securing your network connections

- Monitor & Analyze what enters your network and/or manipulate your data
- Who are the users, groups or Roles that can access
    - Who are the ids that can update/insert/delete?
- What can be manipulated and/or read?
    - Which ids can reference PII or HIPAA data?
- When are these security profiles limited by the time of the day?
- Where the connections coming from?
    - Are the connections geo validated?
    - Are the connections shifted to high security for updates?

Encrypt at database design development time

- Is encryption used within your databases?
  - Encryption protects data at rest
  - Make sure to use Db2 built in facilities for disk encryption
    - **CREATE DATABASE mydb ENCRYPT CIPHER AES KEY LENGTH 256;-**LUW
    - Remember there is some overhead to encryption

- Db2 z/OS requires an encryption password and
  - if desired, an associated hint can be specified
  - the DECRYPT_BIT, DECRYPT_CHAR, DECRYPT_DB functions
    - When SQL passes the password Db2 provides decrypted data

```
SET ENCRYPTION PASSWORD ='BEULKE2021';
INSERT INTO CUST1(SSN) VALUES ENCRYPT_TDES('123-45-6789');
SELECT DECRYPT_CHAR(SSN) FROM EMP;
```

Protect data at rest

- Is encryption used within your databases?
  - Database data at rest
    - No performance overhead from encrypting the database

- DS8800 Storage encryption
    - Encrypts/decrypts data as its written and read
    - Doesn't protect from access attacks
    - Storage device handles encryption-saves CPU

- Encrypt types –
  - Symmetric – one key for encrypting and decrypting the data
  - Asymmetric – two keys one to encrypt another to decrypt
    - Encryption is usually public key
    - Decryption is usually private

Audit Research – z/OS SQL

- PUBLIC is not your friend - REVOKE

1 Db2 Trusted Communications

2 Authorities over/usage privileges
  - Databases, plans, packages, Distinct Types, usage of BPs, SGs & TSs

```
--- CONTAINS ONE ROW FOR EACH TRUSTED CONTEXT.
SELECT *
FROM SYSIBM.SYSCONTEXT
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- CONTAINS ONE ROW FOR EACH TRUSTED CONTEXT.
SELECT *
FROM SYSIBM.SYSCTXTTRUSTATTRS
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- ONE ROW FOR EACH AUTHORIZATION
-- ID WITH WHICH THE TRUSTED CONTEXT CAN BE USED.
SELECT *
FROM SYSIBM.SYSCONTEXTAUTHIDS
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- RECORDS THE PRIVILEGES THAT ARE
---HELD BY USER OVER DATABASE
SELECT *
FROM SYSIBM.SYSDBAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--RECORDS THE PRIVILEGES THAT
--ARE HELD BY USERS OVER PLAN.
SELECT *
FROM SYSIBM.SYSPLANAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--RECORDS THE PRIVILEGES THAT ARE
--- HELD BY USERS OVER PACKAGES.
SELECT *
FROM SYSIBM.SYSPACKAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- PACKAGE OWNER CAN BE A ROLE
-- ALSO IN DOWNERTYPE
SELECT *
FROM SYSIBM.SYSPACKDEP
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- PLAN OWNER CAN BE A ROLE
-- ALSO IN DOWNERTYPE
SELECT *
FROM SYSIBM.SYSPLANDEP
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
---SYSIBM.SYSRESAUTH RECORDS
-- CREATE IN AND PACKADM  ON
-- PRIVILEGES FOR COL; USE PRIVILEGES
--- FOR DISTINCT TYPES,  BPs, SGs & TSs
SELECT *
FROM SYSIBM.SYSRESAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

Audit Research - z/OS SQL

- Run time executables within your environment
  - 3 • AUDIT Policies & Executable modules
  - 4 • ROLES, Ids

```
---SECADM -- CONTAINS ONE ROW FOR
--- EACH AUDIT POLICY.
SELECT *
FROM SYSIBM.SYSAUDITPOLICIES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- CONTAINS AUDITING OPTION COLUMN
--- AUDIT ALL/CHANGE/NONE
SELECT *
FROM SYSIBM.SYSTABLES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- CONTAINS SECURITY DETAILS ON
--- SPs, UDFs & CAST FUNCTIONs
SELECT *
FROM SYSIBM.SYSROUTINEAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- CONTAINS EXTERNAL_SECURITY
--- COLUMN Db2/SESSION_USER/DEFINER
SELECT *
FROM SYSIBM.SYSROUTINES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
-- THE SYSIBM.SYSROLES TABLE
-- CONTAINS ONE ROW FOR EACH ROLE.
SELECT *
FROM SYSIBM.SYSROLES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
---CONTAINS A ROW FOR EACH PARAMETER
-- OF A ROUTINE OR MULTIPLE ROWS FOR
---TABLE PARAMETERS (ONE FOR EACH
---COLUMN OF THE TABLE).
--- ROUTINE CAN HAVE A ROLE IN OWNERTYPE
SELECT *
FROM SYSIBM.SYSPARMS
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
-- THE SYSIBM.SYSSCHEMAAUTH TABLE
-- CONTAINS ONE OR MORE ROWS FOR EACH
-- USER THAT IS GRANTED A PRIVILEGE ON A
-- PARTICULAR SCHEMA IN THE DATABASE.
SELECT *
FROM SYSIBM.SYSSCHEMAAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
-- SYSIBM.SYSSEQUENCEAUTH TABLE
-- RECORDS THE PRIVILEGES THAT ARE HELD
-- BY USERS OVER SEQUENCES
SELECT *
FROM SYSIBM.SYSSEQUENCEAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

Audit Research - z/OS SQL

5 User Ids research

- Understand the extend of the Ids in your system
- Determine risk of each user id
- Begin list to eliminate obsolete ids
- Ids by database and application cross reference

```
-- THE SYSIBM.SYSUSERAUTH TABLE RECORDS THE
-- SYSTEM PRIVILEGES THAT ARE HELD BY USERS
SELECT *
FROM SYSIBM.SYSUSERAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
-- THE SYSIBM.SYSTABAUTH TABLE RECORDS THE
-- PRIVILEGES THAT USERS HOLD ON AND VIEWS
SELECT *
FROM SYSIBM.SYSTABAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- SECADM - ONE ROW FOR EACH ROW PERMISSION
--- AND COLUMN MASK
 SELECT *
 FROM SYSIBM.SYSCONTROLS
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--- LISTS THE DEPENDENT OBJECTS FOR EACH ROLE
SELECT *
FROM SYSIBM.SYSOBJROLEDEP
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
-- THE SYSIBM.SYSCOLAUTH TABLE RECORDS THE
-- UPDATE OR REFERENCES PRIVILEGES  THAT ARE
--- HELD BY USERS ON INDIVIDUAL
--- COLUMNS OF A TABLE OR VIEW.
SELECT *
FROM SYSIBM.SYSCOLAUTH
FETCH FIRST 10 ROWS ONLY WITH UR;
```

What you can do as a developer to help security

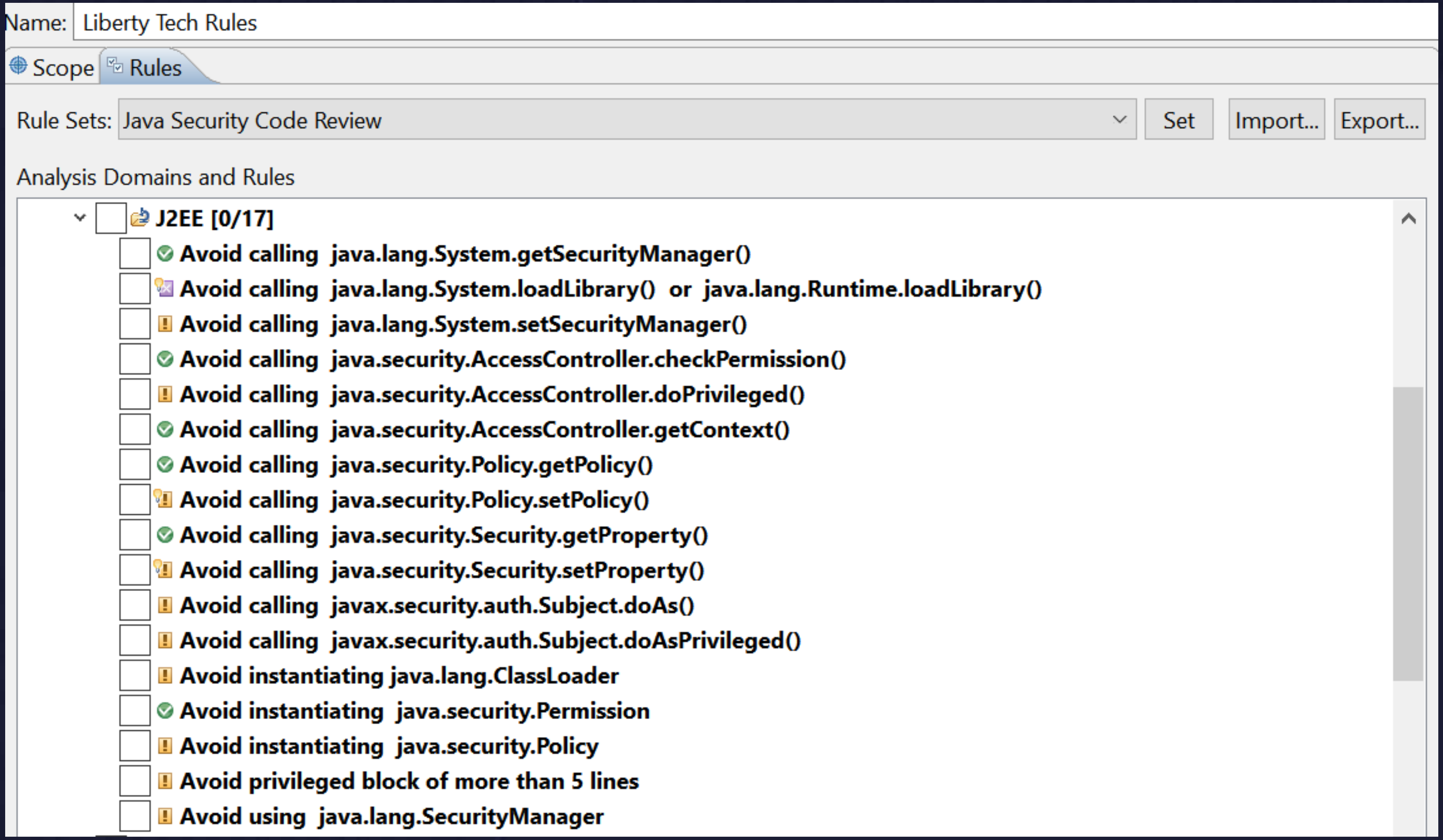
- Check out my Java Db2 Best Practices and Performance presentation
  - Whole presentation of ideas – check it out
    - Lock down JDBC options usage

- Limit fully dynamic SQL within your application
  - Commit scope, Thread Save, stay away from “Frameworks”
  - Understand JVM memory requirements
  - Use encryption routines within your applications
  - Use application modules correctly, appropriately leverage the java modules - No POJO

- Keep it in the database - Limit temporary files throughout your application
  - Understand the Multi-Level security in place – OS, Database, Authentication and User authorizations
  - Limit the access and distribution of PII data - especially on reports
  - Understand where your PII crown jewels of data are and have restricted tighter access

Java Modules to avoid! - 1

- Java modules with known issues



Name: Liberty Tech Rules

Scope | Rules

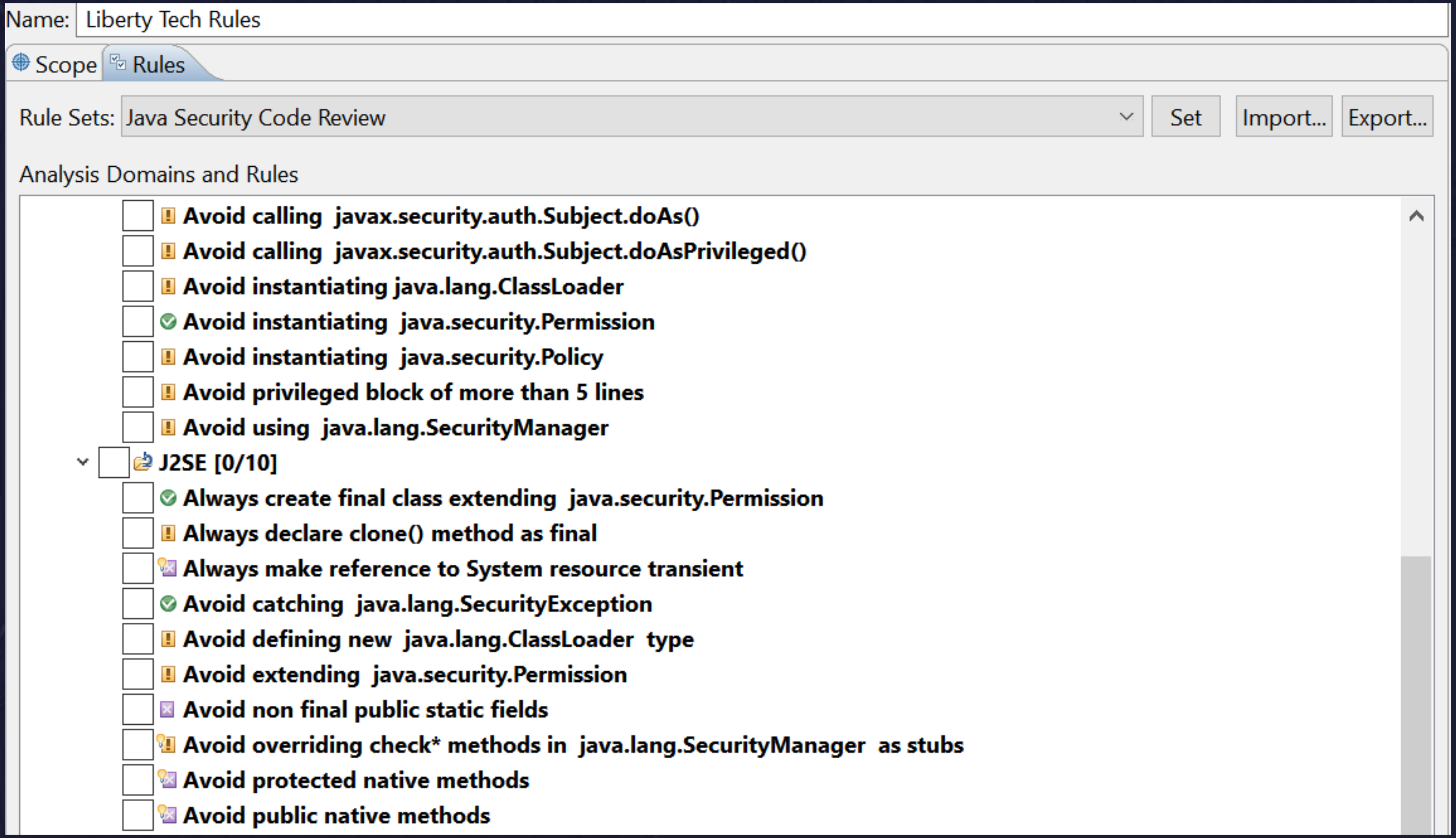
Rule Sets: Java Security Code Review | Set | Import... | Export...

Analysis Domains and Rules

- J2EE [0/17]
  - Avoid calling java.lang.System.getSecurityManager()
  - Avoid calling java.lang.System.loadLibrary() or java.lang.Runtime.loadLibrary()
  - Avoid calling java.lang.System.setSecurityManager()
  - Avoid calling java.security.AccessController.checkPermission()
  - Avoid calling java.security.AccessController.doPrivileged()
  - Avoid calling java.security.AccessController.getContext()
  - Avoid calling java.security.Policy.getPolicy()
  - Avoid calling java.security.Policy.setPolicy()
  - Avoid calling java.security.Security.getProperty()
  - Avoid calling java.security.Security.setProperty()
  - Avoid calling javax.security.auth.Subject.doAs()
  - Avoid calling javax.security.auth.Subject.doAsPrivileged()
  - Avoid instantiating java.lang.ClassLoader
  - Avoid instantiating java.security.Permission
  - Avoid instantiating java.security.Policy
  - Avoid privileged block of more than 5 lines
  - Avoid using java.lang.SecurityManager

Java Modules to avoid - audit! -2

- Java modules with known issues


Name: Liberty Tech Rules

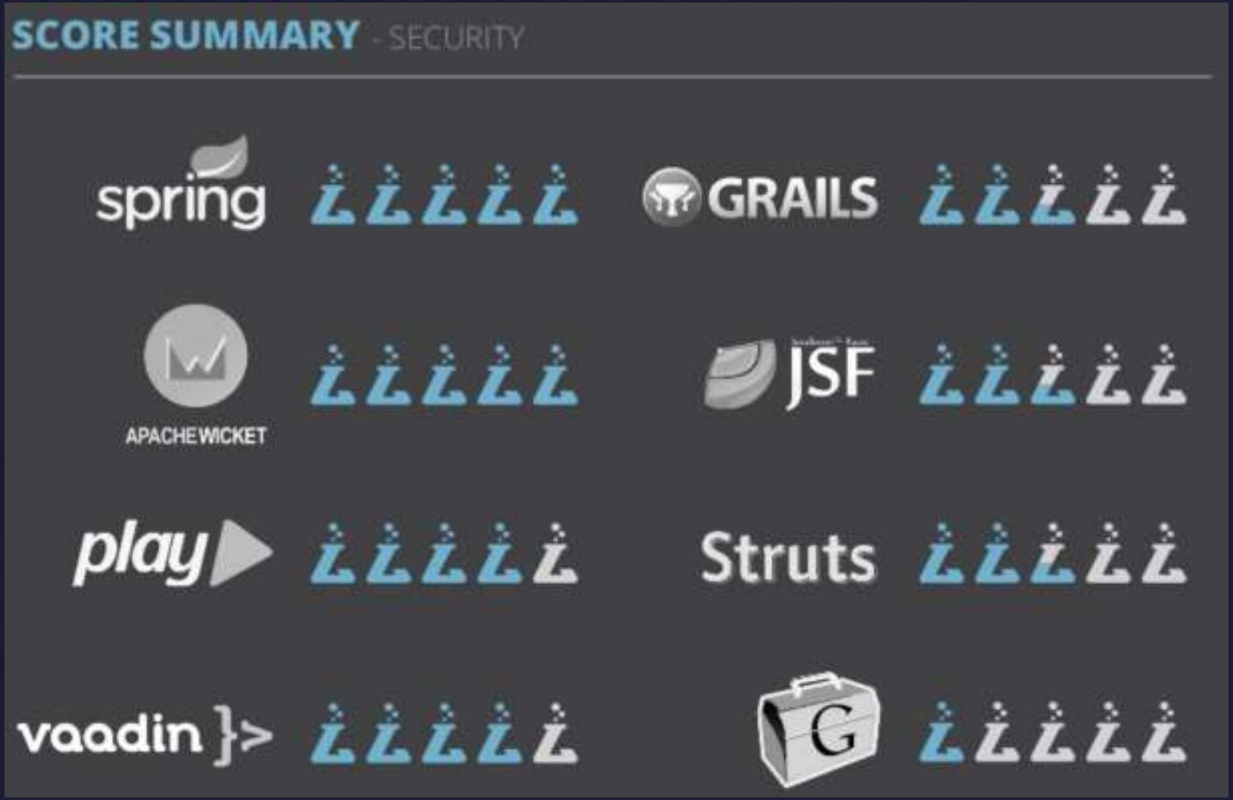
Rule Sets: Java Security Code Review

Analysis Domains and Rules

- Avoid calling javax.security.auth.Subject.doAs()
- Avoid calling javax.security.auth.Subject.doAsPrivileged()
- Avoid instantiating java.lang.ClassLoader
- Avoid instantiating java.security.Permission
- Avoid instantiating java.security.Policy
- Avoid privileged block of more than 5 lines
- Avoid using java.lang.SecurityManager
- J2SE [0/10]
  - Always create final class extending java.security.Permission
  - Always declare clone() method as final
  - Always make reference to System resource transient
  - Avoid catching java.lang.SecurityException
  - Avoid defining new java.lang.ClassLoader type
  - Avoid extending java.security.Permission
  - Avoid non final public static fields
  - Avoid overriding check* methods in java.lang.SecurityManager as stubs
  - Avoid protected native methods
  - Avoid public native methods

Framework liability?

- Picture says it all
  - How secure is your framework?
  - How many releases are your applications behind?
  - Java 16 coming which version are you on?
  - Old/New Spring releases are ***very*** vulnerable!
  - POJO security is achievable needs verification!
    - Also needs to stay up with software fixes

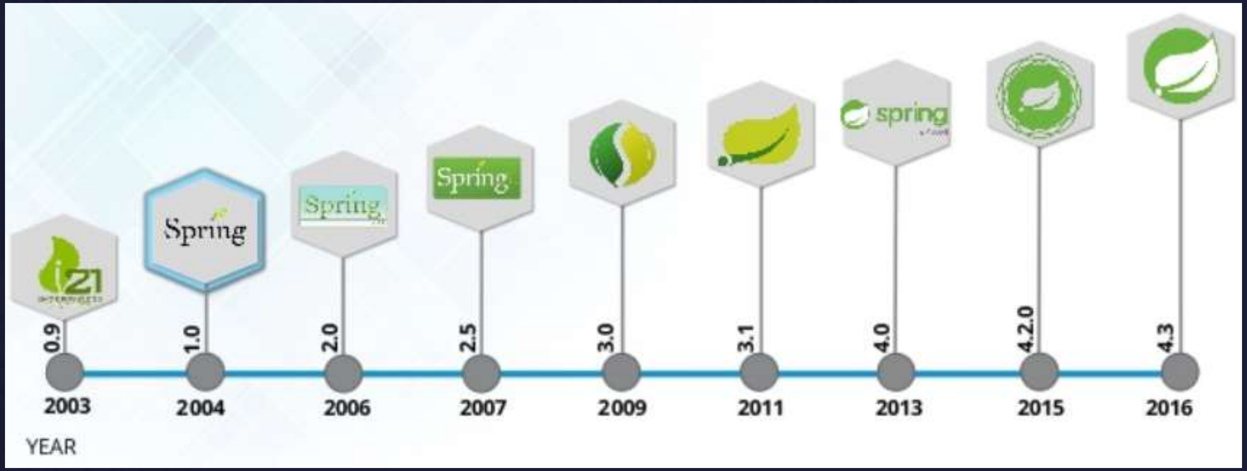


https://www.slideshare.net/kunalashar/the-2014-decision-makers-guide-to-java-web-frameworks

✓Check your software against the NIST National Vulnerability Database
  ✓ https://nvd.nist.gov/vuln

Framework in production reviewed/updated lately?

- Applications security best practices
  - Eliminate or upgraded old software versions
    - Frameworks – Spring, Ruby Groovy etc.
    - Old Java and supporting software libraries
    - Especially **Open Source code** with ***known*** issues
  - Old Application (JUNIT) testing reviews
    - JavaScript security execution
    - XSS Cross-Site Scripting
    - Research app for indirect site references
  - All types of ***injection*** possibilities that need inspection
    - R, Python, JS, XML, SQL, NoSQL, LDAP etc..
      - Audit everywhere a value is passed to a program
      - Verify using parameter markers in all application languages!!!



Summary/Checklist – Developers

- Validate your application software is current and up-to-date with all patches!
  - Minimize unknown Open Source from your software base as possible
  - Follow development security best practices, standard security procedures and pier code review
  - Update Version of Java and patches

- Develop your applications with security first!!
  - Verify your software and its version against the NIST & CVE know security issues database
  - Don't use outdated or known modules that have security issues

- Verify your source code
  - Using FIPS 140-4 level, use two factor authentication when possible
  - Verify all communication between application and database is encrypted
  - Verify using parameter markers in all application languages!!!
  - Make sure to only log non critical PII data
  - Scan your source code for know security vulnerabilities – bad modules list

Summary/Checklist - DBAs

- Understand your PII data within your database tables
- If possible use Steganography in your designs to protect PII data
  - Salt & Pepper is an easy way to add Steganography security with only minor changes

- Review all interfaces for encrypted communication
  - Prioritize using TLS/SSL and encrypted interfaces at all times
  - Encrypt everything possible – database tables, backups & files at rest
  - Minimize public database network connection information

- Use the security queries to minimize/eliminate PUBLIC access
  - Partner with RACF/ACF2 Security team to backstop/validate all access allowed
  - Eliminate minimize all old access authority
  - Automate logs of all usage of service accounts between platforms/servers/processes

Thank you!

Db2 Security Best Practices Volume III+

By Dave Beulke

Dave Beulke and Associates

Dave @ d a v e b e u l k e .com

Check out my Db2 Security Blogs series

Proven Security &
Performance Tips:
www.DaveBeulke.com