Db2 Security Best Practices

Staying up with the threats

BY Dave Beulke

DAVE @ DAVEBEULKE.COM

Twitter: @DBeulke
LinkedIn www.linkedin.com/in/davebeulke

Dave@davebeulke.com

- Member of the inaugural 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 Performance Tuning

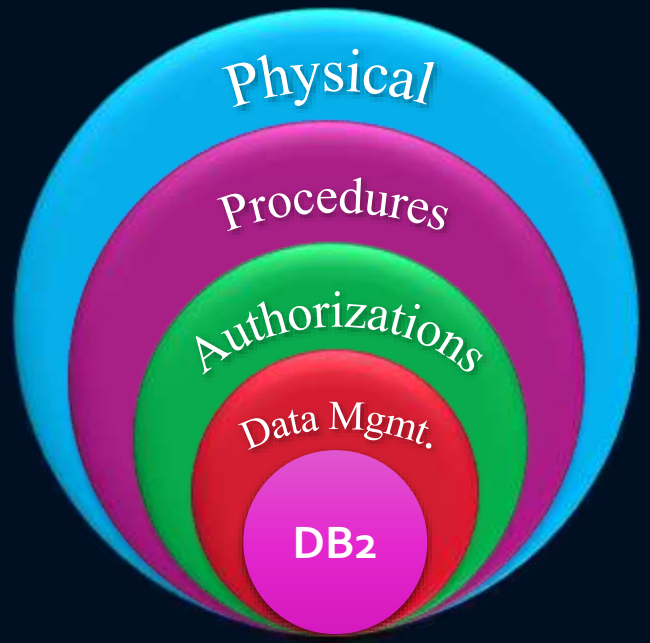
- **Educational Seminars**
  - Java Security for Application Developers
  - Db2 Version 12 Transition
  - Db2 Performance for Java Developers
  - Data Analytics Designs for Performance
  - How to Do a Java Performance Review

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..
    - Everything is connected in the architecture of RESTful Services
  - 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 are blamed for everything

- Goes beyond a single operating system
  - System administrators – SYSADMs, SECADMs
  - DBAs – DBADMs
  - Operators – SYSOPR
- Goes beyond a single application
  - So many processes
  - Variety of programming languages – new open source languages are full of holes
  - How many levels of authorized application does your company have?
- Welcome to the matrix of security complexity
  - Thousands/millions of security situations



Technology updates are most important security factor!

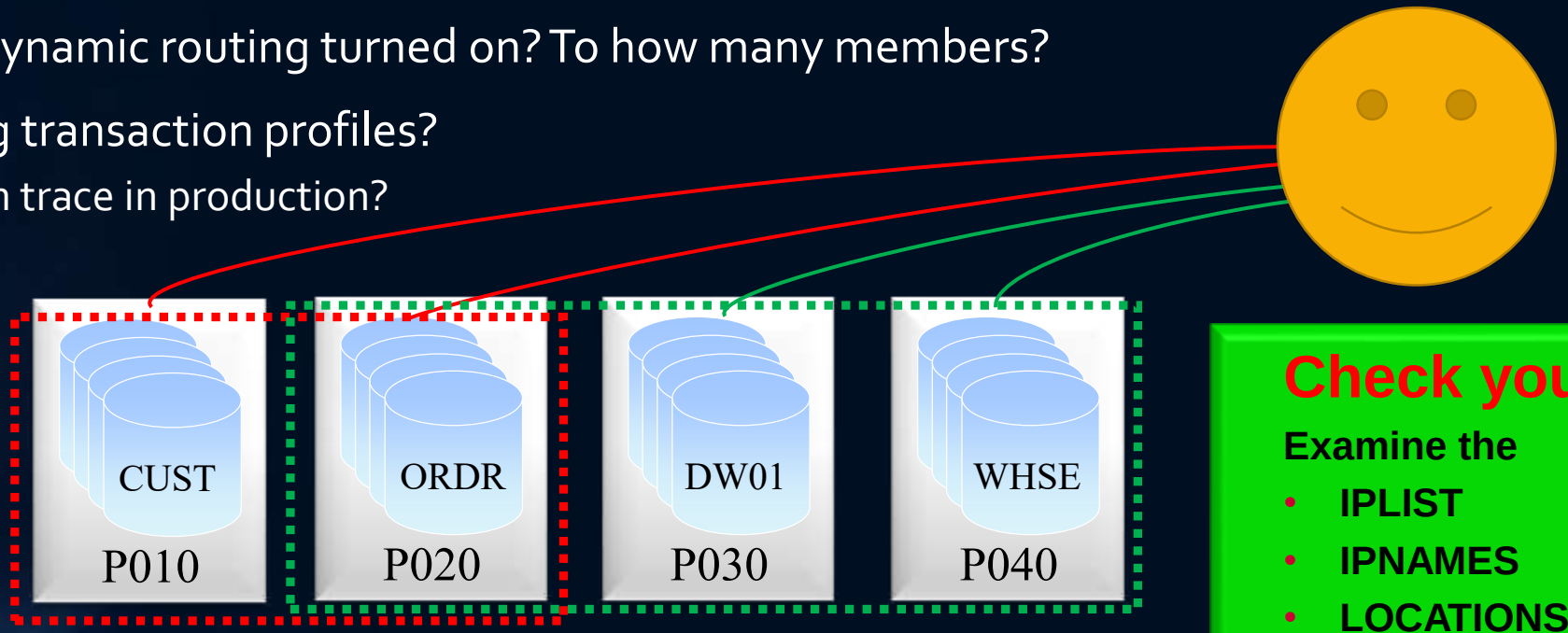
- Evaluate the technology and security factors of Db2 z/OS applications accessing it!
  - Vendor viability
  - Version overall age
  - Patch level
  - Patch frequency
  - Open source support contract
- Early identification of the security vulnerability and misconfigurations
- Evaluate and audit any shared security services and/or controls
- Understand entire stack security requirements and technology capabilities
  - According to Gartner **99% of hacks are because of outdated technology**

According to IBM, Crowdstrike and Akamai

- Number of threats continue to raise
  - Types of threats are getting more complex
  - Most hacks are because of somebody's actions or lack of actions
    - https://www.ibm.com/security/digital-assets/xforce-threat-intelligence-index-map/
    - https://www.crowdstrike.com/resources/reports/2020-crowdstrike-global-threat-report/
  - IBM X-report (including publicly accessible cloud storage, unsecured cloud databases, and improperly secured rsync backups, or open internet connected network area storage devices) accounted for 86 percent of the records compromised in 2019
    - https://threatpost.com/poorly-secured-docker-image-rapid-attack/154874/
  - **https://threatpost.com/70-of-apps-open-source-bugs/156040/**
    70 Percent of Mobile, Desktop Apps Contain Open-Source Bugs
    - https://blogs.akamai.com/sitr/2020/04/a-brief-history-of-a-rootable-docker-image.html

How many virtual or direct connections

- Understand TCP/IP - DVIPA/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 trace in production?



Check your settings!

Examine the

- **IPLIST**
- **IPNAMES**
- **LOCATIONS**
- **LULIST**

DBAs collaborate with everyone-start security documentation

- Start setting security expectations during development
  - **Application technology base inventory**
    - Risk profile of technology – version and patch history
- Document system's important data and its security
  - GDPR, PII profile requirements/exposure
    - **Application Interface inventory**
    - **Document all application sensitive, proprietary, HIPAA, GDPR, PII data**
    - Security usage profiles unique and shared with other applications
    - Document all interfaces, services and shared application access points
- Socialize, document and audit the important security aspects:
  - Systems architecture – cloud, hybrid, HTAP, outsourced, cross-platform, database operational profile
    - Work with installer, configuration review for security definitions and **interfaces handling for PII, HIPAA, GDPR data**
  - Document **all** the interfaces available to any application, user, administrator, vendor and operations
    - Start with the interfaces to the important data

Danger increases!

Average probability is 27.7% that organizations in the study will have a data breach in the next 2 years.

Last year, the average probability was only 25.6%!

Standard protocol for connections

- Federal Information Processing Standard – (FIPS-140 2)
  - DB2 z/OS still the leader implemented with TLS in DB2 z/OS
  - Government standard for secure interaction
    - TLS SSL client/server cryptographic protocol
      - **Redpaper** Configuring TLS/SSL Secure Client/Server Communications www.redbooks.ibm.com/redpapers/pdfs/redp4799.pdf
      - *Should be used for all your connections for CICS and DB2 applications*
- Implement IP filtering
  - Monitor report rogue access
- Policy base Traffic IP Routing
  - Do not allow any out of pattern access
- SSL is not secure must be using TLS!

BoD ☞ CEO ☞ CISO ☞ CDO ☞ DBA - Cycle of engagement

- Document security baseline of legacy applications first and evaluate security of new architectures and applications
- Evaluation of the **entire** technology stack is the key to determining your security exposures
- Scheduling yearly/regular security audits, drives the information baseline for security for upper management evaluation
- Documentation to cover your security procedures?
- What security audit tasks are included in normal DBA activities or application maintenance schedules?

Establish data management security inventories

- Establish DBA Security documentation
  - **48% of companies do not regularly report on security**
    - Have a plan of protection to detect, protect and react
    - Begin to develop cyber breach response plan (CBRP)
- Understand your legal liabilities of a data breach
  - Ready – GDPR=$,$$$,$$$ -Good for budget allocation
- Inventory all hard and soft targets
  - Talent, audit security tools, PII data, home grown or packaged software etc.
- Need to raise the DBA profile of security awareness and document security audit interactions
  - Drive the process or be driven by the security issues – your choice
- Raise the profile of the security evaluation work for everyone especially the Board of Directors
  - Start a commitment of reporting on security bi-weekly/monthly/yearly basis

Know your levels of monitoring

- Best practices for systems
  - **Multi-layered** protection
    - Multi-Tiered system access inventoried
    - Multiple logon authentication - 2-factor authorization
    - Almost all trusted devices and clients (>99%)
    - Monitoring Tools standard for PII HIPAA
  - Known active security lifecycle procedures
    - Active lifecycle security system & application audits
- System and Application testing techniques
  - Security & configurations standardized
    - UNIX & z/OS
  - Cross platform monitoring/auditing tools utilized
  - Automated Security breach response/tools
    - z/OS RACF 3-strikes
    - UNIX Security automation, logging and response

Assess and tighten production testing security perimeter

- Storage/Data pools used within the different environments
  - Understand your storage configuration to **realize shared device exposures**
  - Make sure the DS88OO Encryption license is purchased with the hardware
  - Map out the channel connections used within your environment
  - Verify the security ids that have access to your Db2 databases and their HLQs
    - Not just Db2 ids but also all the systems, storage and operations user ids with access to those storage pools
  - **Encrypt *all* data at rest!**

- Security ids
  - **Where in your systems or connections can your user id be changed into another id?**
  - What services or operational authorities are there over your Db2 systems, applications or tools?
  - Where are individual ids still used within your system? Every access should be a ROLE!

Encrypt database data at design development time

- Verify encryption used within your databases if possible
  - Encryption protects data at rest
  - Make sure to use DB2 built in facilities for disk encryption
    - DS8800 Encryption licenses purchased for full coverage

- DB2 z/OS can requires an encryption password and hint
  - 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 ='BEULKE2020';
INSERT INTO CUST1(SSN) VALUES ENCRYPT_TDES('123-45-6789');
SELECT DECRYPT_CHAR(SSN) FROM EMP;
```

Security practices for Db2 database definition

- Best practices for databases
  - Broad encryption type protection
    - Data at rest storage encryption key store protection
  - Design your database with steganography features
    - Table splits/naming
    - Column splits/naming
    - Data Procedures secured for data access
- Design/improve with GDPR, PII and HIPAA granular security layer
  - Table special auditing
  - Columnar security
  - Element access control
  - Column masking/encryption
  - Column Obfuscation
- Verify trusted and encrypted communication **everywhere** your system can control!

Steganography – Obscure to secure

- 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 with FIRST_NAME = 'John' - J
```

- SALT and Pepper is data shift!

Steganography

- So data shift of five positions based on first name
  - INSERT '678912345' into SSN since FIRST_NAME ="John" - J
- Application knows the data shift

```
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 with FIRST_NAME = 'John' - J
```

- 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.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 (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 '**4**123456789**6**' into 11 digit

```
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 (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

Db2 Functionality – Built in Steganography

- Db2 functionality encryption-like along with redefined Columns to different encoding formats
  - SBCS data - a single-byte character set representation
  - DBCS data - a double-byte character set representation
  - Mixed Data – mixture of characters from a single-byte character set (SBCS) and a multiplebyte character set (MBCS)
  - BINARY definition - binary strings
  - EBCDIC table can contain one or more Unicode columns
    - Understand the definition and UNICODE usage restrictions
- Row and Column Access Controls - RCAC
  - Permissions based on user/values
- Column Masks
  - Column permissions

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 minimized
- Limit Request Size -
- Limit Reply size
- Limit # SQL for each connection/TX

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

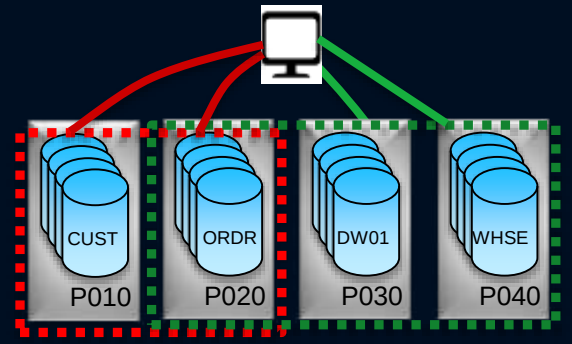
Framework liability?

- Picture says it all
  - How secure is your framework?
  - How many releases are your applications behind?
  - Java 10 coming which version are you on?
  - Old Spring releases are ***very*** vulnerable!
  - POJO security is achievable but difficult and needs verification!
    - Also needs to stay up with software fixes

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

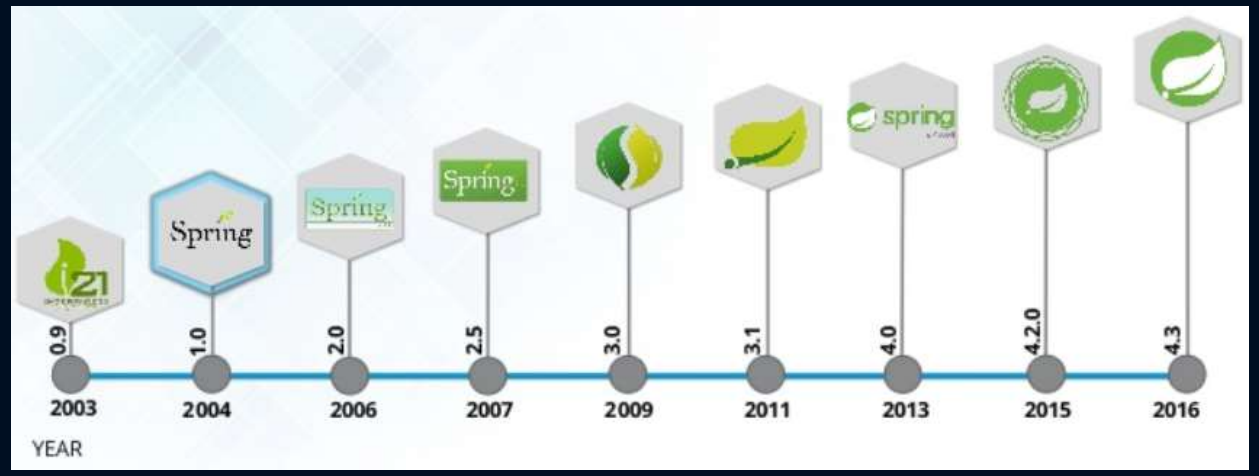
Frameworks can be the most vulnerable and risky

- Framework is only secure if…
  - **Programming is done with the latest APIs, certificates are used correctly and interfaces security reviewed**
  - Configurations is confirmed to be configured and controlled properly
  - Change control and implementation is secured with good procedures
- Spring can use several configurations to secure the environment
  - Are you using the XML based or Java based Spring security classes, configuration and procedures
    - Have you migrated from the old one to the new one? Did you update the configuration to current conventions?
    - Did you update the interface partner security also?
- HttpSecurity has 10 different methods
  - Are each of your applications set up correctly? Reviewed lately?
  - The security **antMatcher("/api/**")** needs to be invoked before **addFilterAfter(…)**
    - So ***filter is only applied to URLs matching the pattern "/api/**".***



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



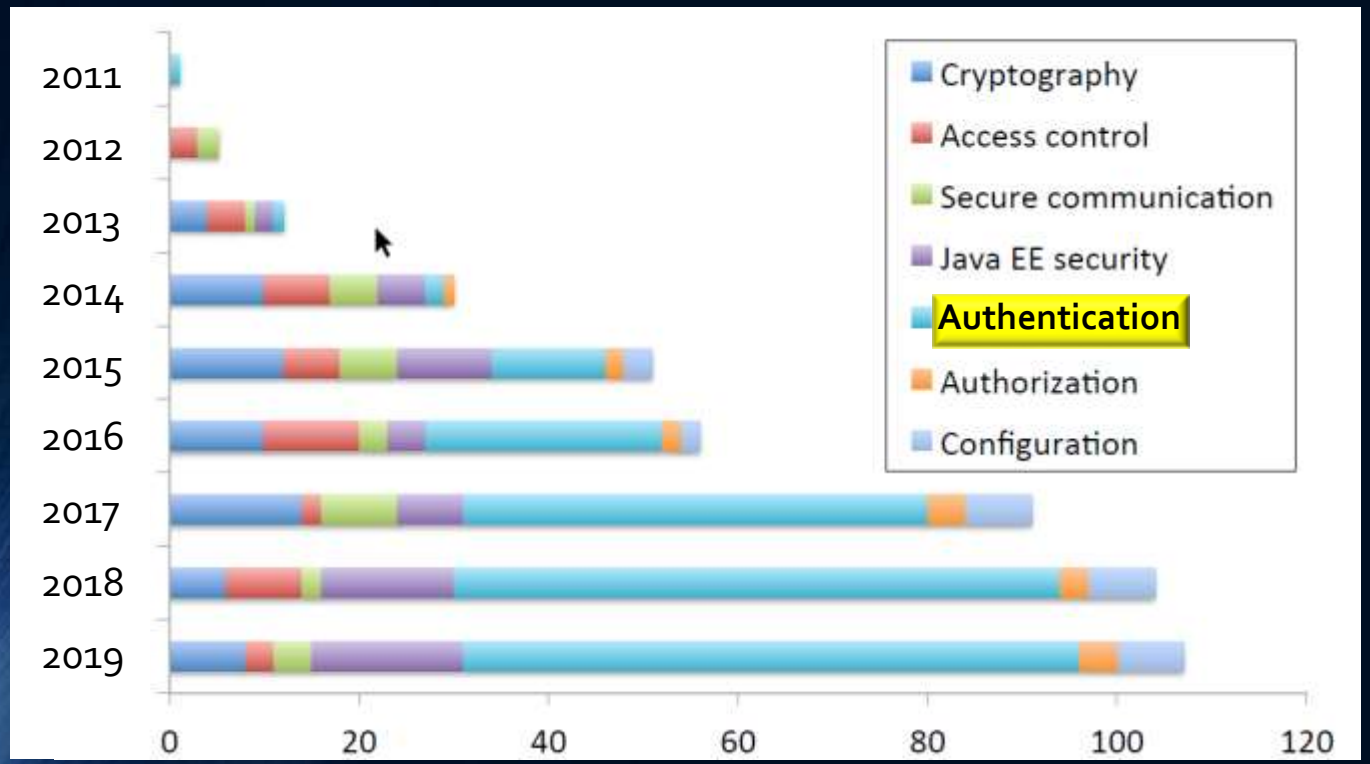
  - All types of ***SQL injection*** possibilities that need inspection
    - R, Python, JS, XML, SQL, NoSQL, LDAP etc..
      - Everywhere a program passes a value to a program accessing the database

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 Java and Cloud architectures
  - Establish application security baseline
    - **Establish Security Audit Risk Rating for each application!**

7 phases of security

- Security roles, access control, and authentication requirement
  - Authentication is most important and popular of the security aspects



  - Problems caused by wrong versions of software libraries and version conflicts between dependent processes
  - Always want two factor authentication with user interfaces!

How do applications access

- Governance, Risk and compliance management
- How many levels/categories are in your authorizations?
  - Authorities within or shared across environments
    - Roles
    - Groups
    - User Ids should be avoided and minimized
- Row and Column Access Control - RCAC
  - Steganography techniques are built in to your design and Db2
  - Application impact of RCAC controls
    - Guarantees data alignment with RCAC rules
    - Impacts SELECT Result Sets, INSERT/UPDATE/DELETEs because of data RCAC alignment

Research all versions of your software

- Research your framework, application libraries and special situations
  - Older or Community version of JBoss, Spring, etc….
  - IBM/Redhat has its own CVE
- National Vulnerability Database – **https://nvd.nist.gov/**
- NIST is the national standard – national crisis
  - Mitre also - https://cve.mitre.org/ also https://attack.mitre.org/
- Research your exposures and endpoint's status
  - iOS and Android rogue apps
  - Chinese phones send data back
  - **Google tracks every Android phone movement!**
    - https://qz.com/1131515/google-collects-android-users-locations-even-when-location-services-are-disabled/

Understand your security threat landscape

- Threat scenario for each application?
- Where are your PII data valuables?
- Email exposures
  - Attachment scanning
  - Link validation
  - Email training
- Cloud provider risk
  - Our cloud is move secure than theirs

Audit Research - 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 - 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 - 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;
```


dave @ d a v e b e u l k e. c o m

Audit Research - SQL

6 Understand the connections

- Understand the many connections
  - Determine risk of each connection
  - Minimize TCP/IP connections – VIPA/DVIPA

```
--CONTAINS A ROW FOR EACH IP ADDRESS
--THAT CORRESPONDS TO A REMOTE DRDA SERVER
SELECT *
FROM SYSIBM.IPLIST
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--CONTAINS A ROW FOR EACH REMOTE DRDA SERVER THAT
--DB2 CAN ACCESS USING TCP/IP
SELECT *
FROM SYSIBM.IPNAMES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--CONTAINS A ROW FOR EVERY ACCESSIBLE
--REMOTE SERVER
SELECT *
FROM SYSIBM.LOCATIONS
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--CONTAINS A ROW FOR EACH REAL LU NAME THAT IS
--ASSOCIATED WITH THE DUMMY LU NAME FOR A DATA
--SHARING GROUP
SELECT *
FROM SYSIBM.LULIST
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--CONTAINS A ROW FOR THE CONVERSATION LIMIT FOR EACH
--COMBINATION OF LU NAME AND VTAM LOGON MODE
--DESCRIPTION
SELECT *
FROM SYSIBM.LUMODES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--CONTAINS A ROW FOR EACH REMOTE SNA
--CLIENT OR SERVER
SELECT *
FROM SYSIBM.LUNAMES
FETCH FIRST 10 ROWS ONLY WITH UR;
```

```
--CONTAINS A ROW FOR EACH VTAM LOGON MODE AND
--COMBINATION OF AUTHORIZATION ID, PLAN NAME,
--AND LU NAME
SELECT *
FROM SYSIBM. MODESELECT
FETCH FIRST 10 ROWS ONLY WITH UR;
```

Ideas for auditing your environment security

1. Tighten up your infrastructure security definitions every year
2. Integrate security risk assessments into maintenance and development lifecycle
3. Work to eliminate ids access and redefine controls to ROLES/RACF
4. Know GDPR, PII, and HIPAA elements, use Db2 facilities and steganography methods to protect them
5. Encrypt database, backups, archives any data copies at rest, and all communications to protect and minimize the threats
6. Monitor and log ALL activity – protect and encrypt all logs!

Get security audit and documentation going today!

dave@davebeulke.com

Thank you!

Twitter: @DBeulke
LinkedIn www.linkedin.com/in/davebeulke

D a v e @ d a v e b e u l k e . c o m

- Take any questions now or email them if you need ideas/support later!

Support IDUG's Virtual Meeting in July - Sign up at IDUG.org