Breaking Databases

via SQLi attacks

Azqa Nadeem

PhD Student @ Cyber Security Group

TU Delft

About Cyber Security lecture series

- A hot topic, a buzz term
- Introducing the Cyber Security lecture series
  - Cyber security topics in existing courses
- Announcements
  - Assignment 3
  - Exam questions

https://www.tudelft.nl/cybersecurity/

Agenda for today

- Part I
  - Data breaches and their threat landscape
  - Information Security principles
  - Top threats for databases
  - Mitigating security threats
- Part II
  - SQL injection attacks
  - Injecting SQL queries ← Hands-on!
  - Analysing SQLi attacks
  - Best practices to avoid SQLi

Go to https://b.socrative.com/login/student/

Scan me

Room Name: **IDMQ3**

Why would anyone ever hack a database?

The role of databases

- *A database is the heart of an organization.*

- "Database servers are the most compromised asset in an organization."
  – Verizon 2018

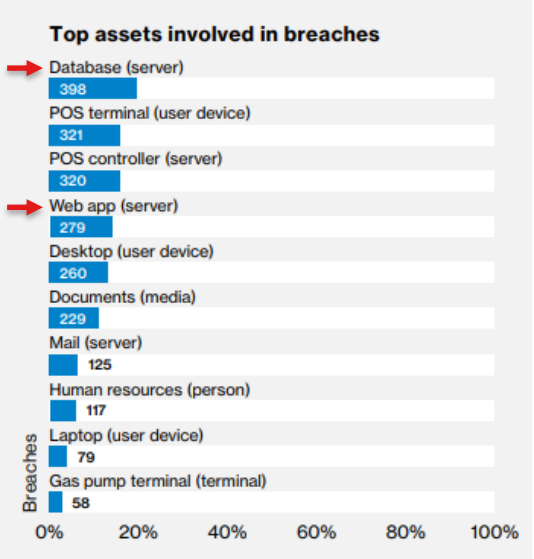


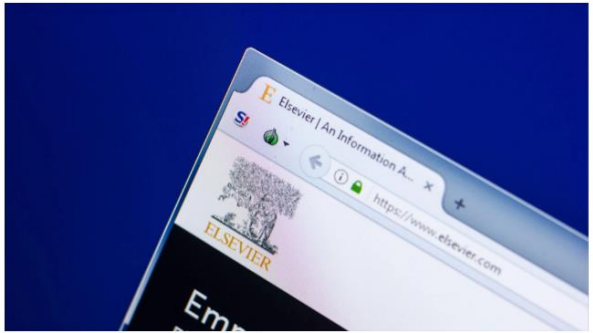
Figure 8. Top varieties of assets within confirmed data breaches (n=2,023)

… In the news

CYBERSECURTY | By Joseph Cox | Mar 18 2019, 7:09pm

Education and Science Giant Elsevier Left Users' Passwords Exposed Online

Due a to a misconfigured server, a researcher found a constant stream of Elsevier users' passwords.



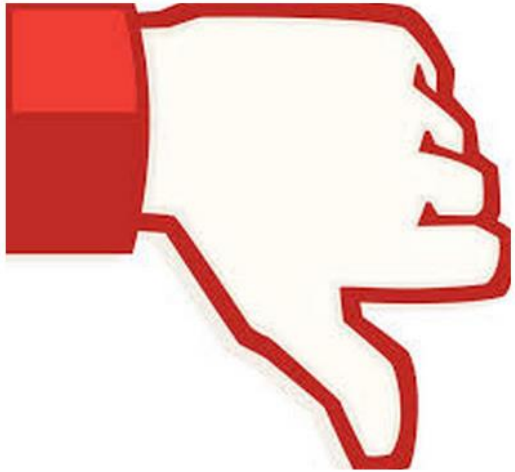
https://motherboard.vice.com/en_us/article/vbw8b9/elsevier-user-passwords-exposed-online

… In the news

21 MAR 19

Facebook Stored Hundreds of Millions of User Passwords in Plain Text for Years

Hundreds of millions of **Facebook** users had their account passwords stored in plain text and searchable by thousands of Facebook employees — in some cases going back to 2012, KrebsOnSecurity has learned. Facebook says an ongoing investigation has so far found no indication that employees have abused access to this data.



https://krebsonsecurity.com/2019/03/facebook-stored-hundreds-of-millions-of-user-passwords-in-plain-text-for-years/

… In the news

8,861 views | Dec 4, 2018, 01:47pm

Marriott Breach Exposes Far More Than Just Data

David Volodzko Contributor
Manufacturing
I am an editor at the technology and information company Brightwire.

Signage on a door to a Marriott International hotel in Chicago, Illinois. A cyber breach in Starwood's reservation system had allowed unauthorized access to information about as many as 500 million guests since 2014. Photographer: Daniel Acker/Bloomberg © 2018 BLOOMBERG FINANCE LP

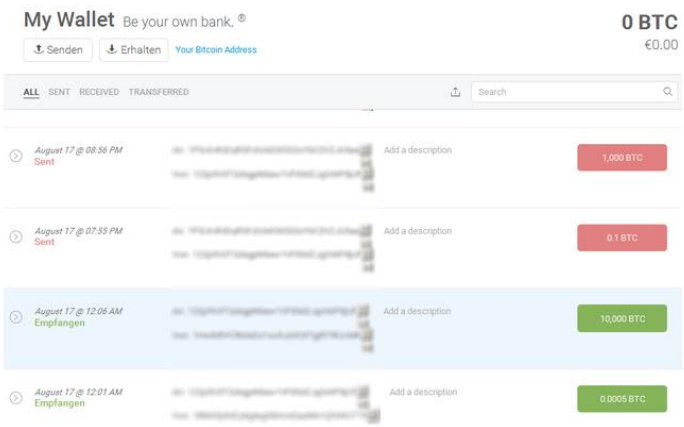
https://www.forbes.com/sites/davidvolodzko/2018/12/04/marriott-breach-exposes-far-more-than-just-data/#1f0d3c276297

… In the news

How I hacked hundreds of Bitcoins! AMA

hacker0 (25) in bitcoin • 2 years ago

It all begins 3 years and a 3 month ago.



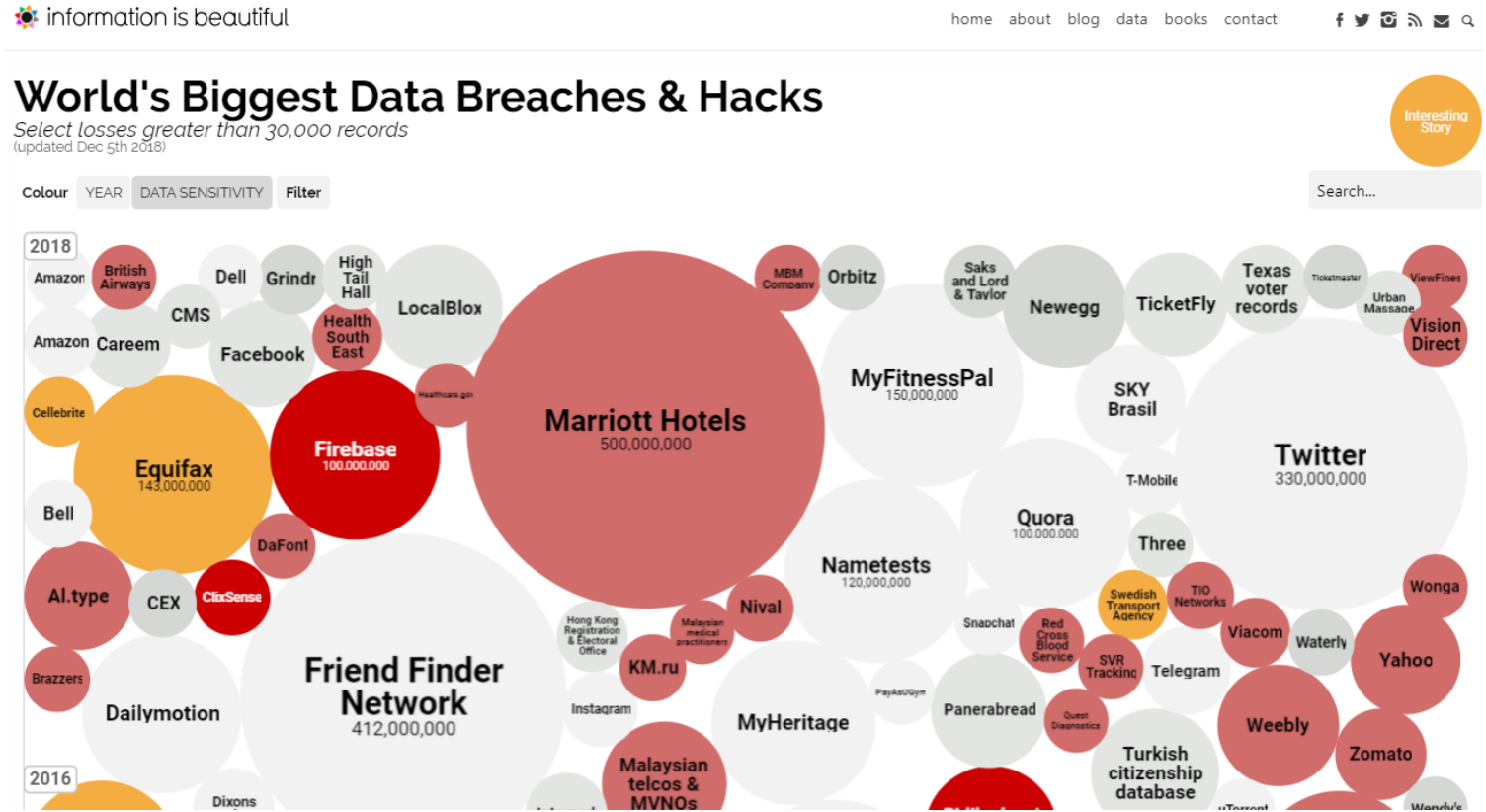
Beginning 2013:

I was a hacker who focused on phishing victims bank details and selling them. I was working full time in a company and doing this black market

https://steemit.com/bitcoin/@hacker0/how-i-hacked-hundreds-of-bitcoins-ama

… In the news

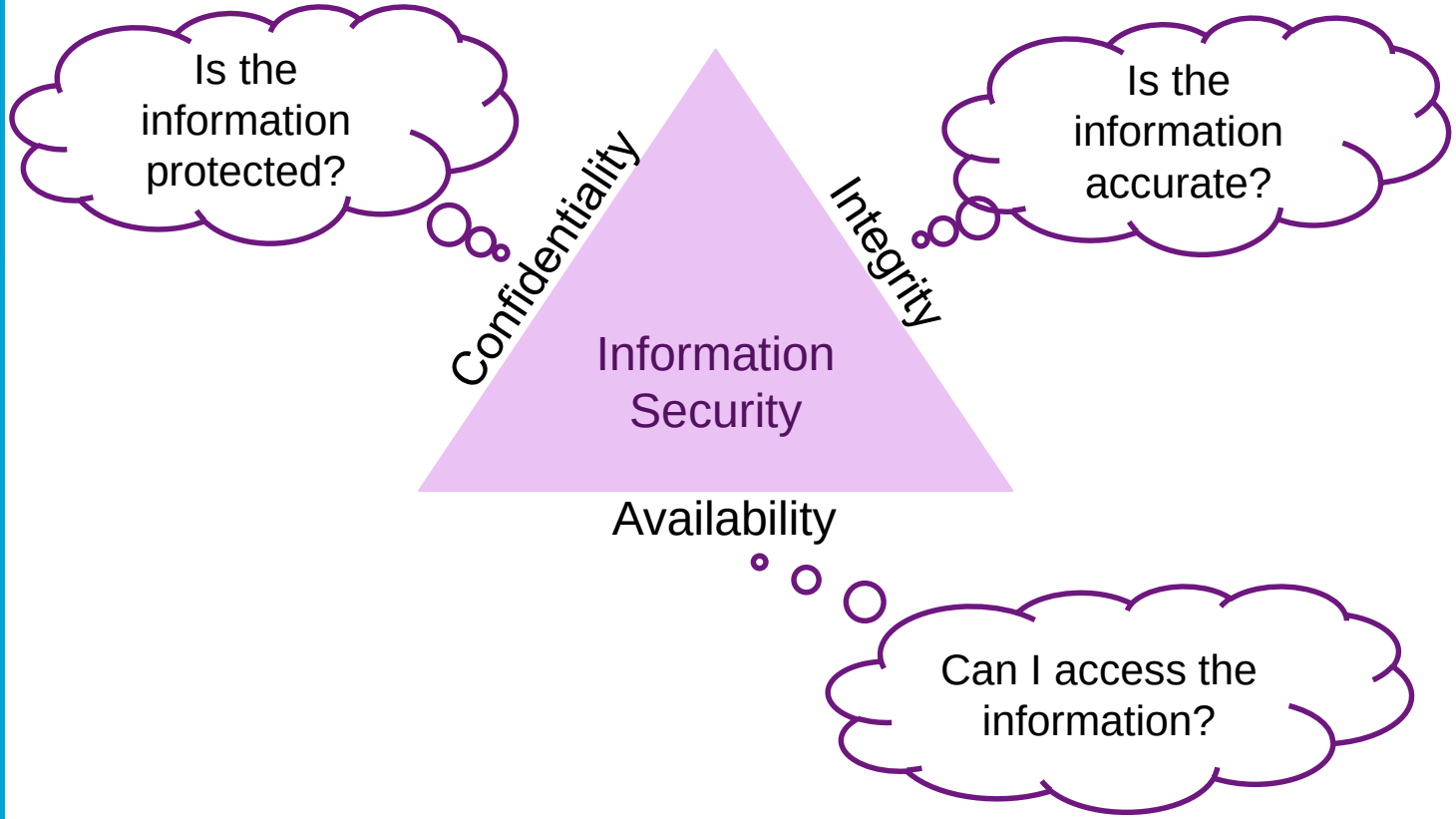
Extra notes:
MyHeritage does DNA sequencing



https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

The CIA triad

Extra notes:
. Can unauth see it?
. Can unauth change it?
. Can legit user access it?



Threats to DB Security

Room: IDMQ3

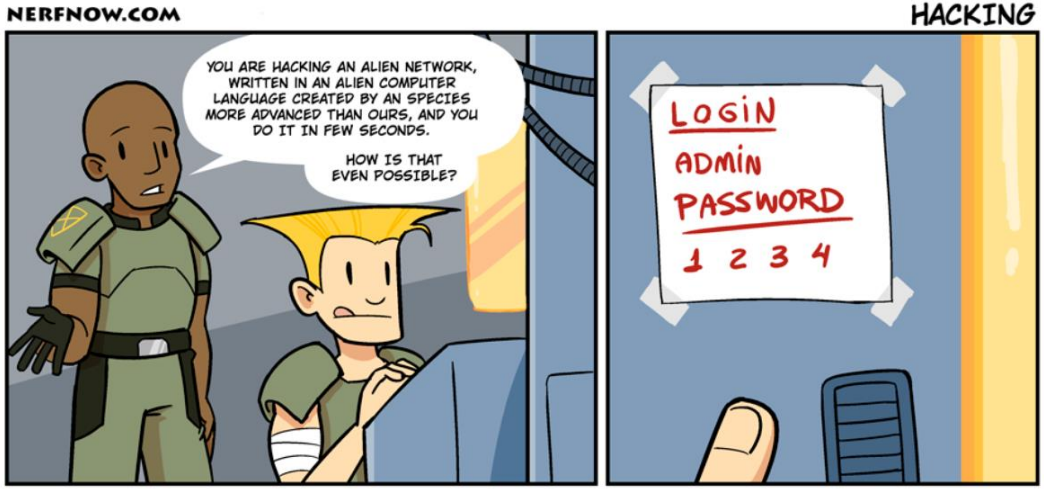
Worst Passwords of 2018

1. 123456
2. password
3. 123456789
4. 12345678
5. 12345
6. 111111
7. 1234567
8. sunshine
9. qwerty
10. iloveyou
11. princess
12. admin
13. welcome
14. 666666
15. abc123
16. football
17. 123123
18. monkey
19. 654321
20. !@#$%^&*
21. charlie
22. aa123456 w
23. donald
24. password1
25. qwerty123

https://www.securitymagazine.com/articles/89694-the-top-100-worst-passwords

Threats to DB Security

1. Weak authentication



Verizon 2017-2018 Data Breach Investigations Report

Threats to DB Security

Extra notes:
. Default username/passwords
. Easy-to-guess passwords
. Passwords written on sticky notes

1. Weak authentication

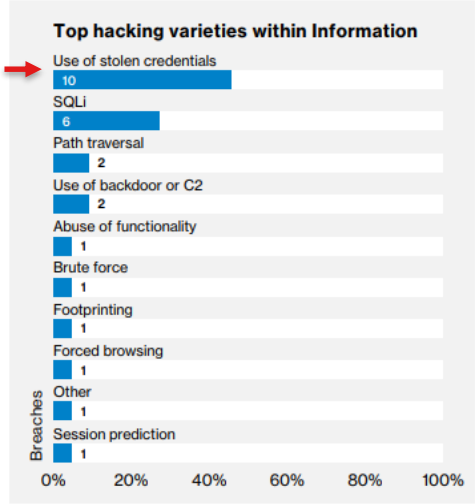


Figure 33. Top hacking varieties within Information breaches (n=22)

Verizon 2017-2018 Data Breach Investigations Report

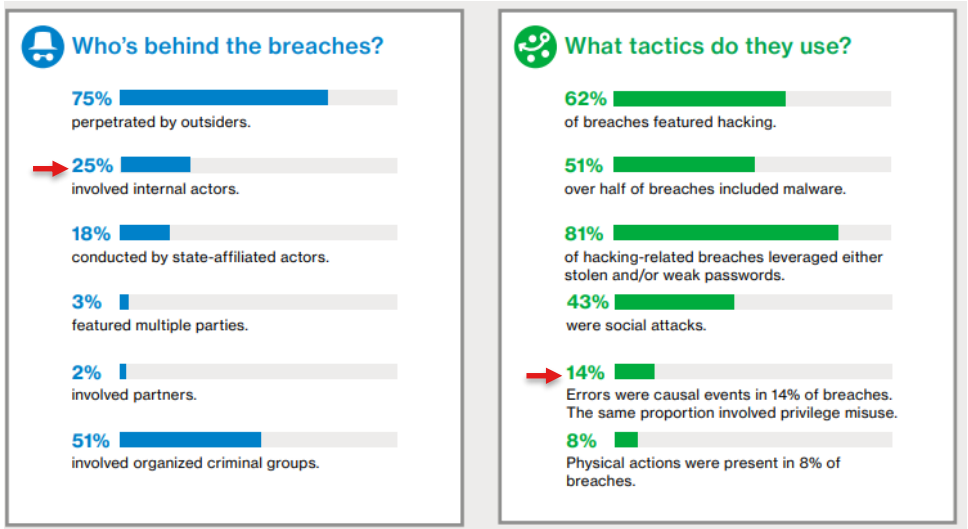
Threats to DB Security

1. Weak authentication
2. ?

"Got a minute, sir?"

Threats to DB Security

1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats

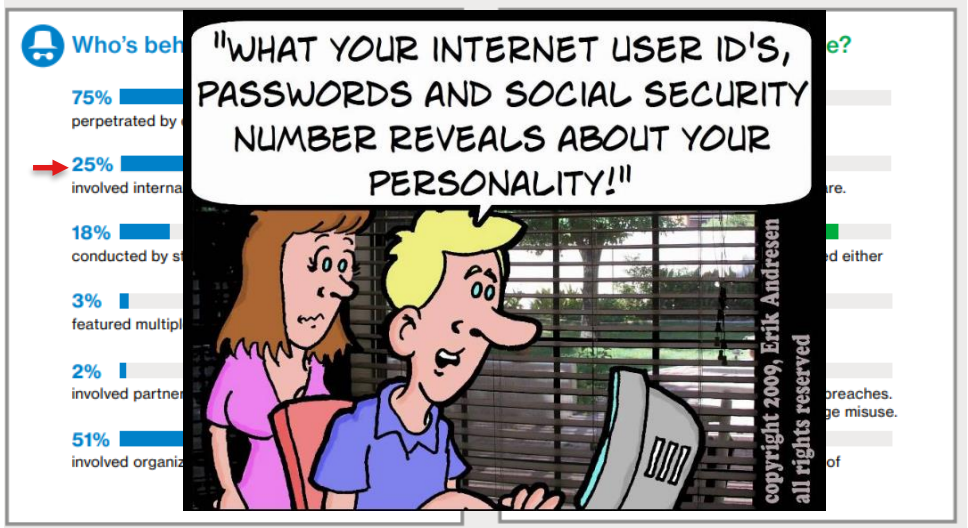


Verizon 2017-2018 Data Breach Investigations Report

Threats to DB Security

Extra notes:
. Giving away privileges like they're candy.
. Insider threats – Tricky business -- balance between convenience and security
. Phishing attacks

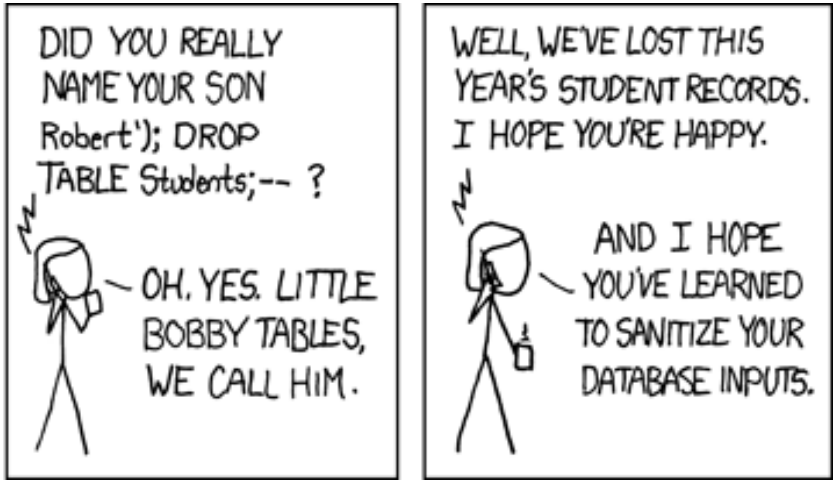
1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats



Verizon 2017-2018 Data Breach Investigations Report

Threats to DB Security

1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats
4. ?



Threats to DB Security

1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats
4. DB injection attacks
   - SQL injection attacks

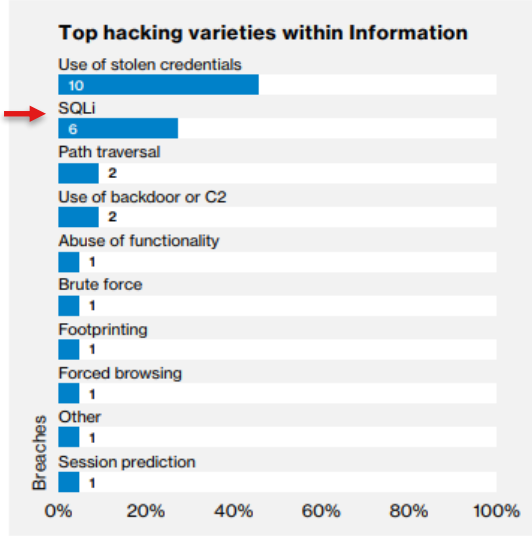


Figure 33. Top hacking varieties within Information breaches (n=22)

Verizon 2017-2018 Data Breach Investigations Report

Threats to DB Security

1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats
4. DB injection attacks
   - SQL injection attacks
   - NoSQL injection (NoSQL does not mean you are safe!)

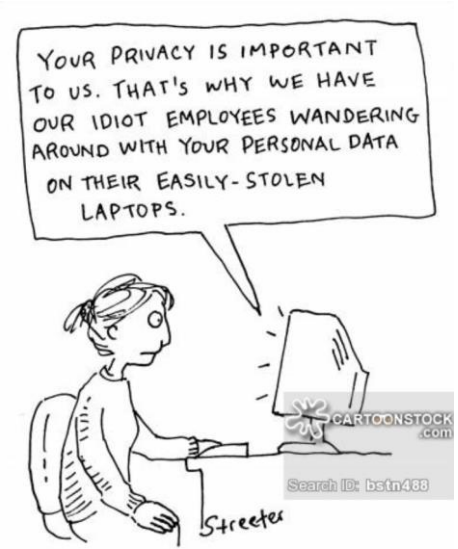
From the MongoDB documentation

- "One valid way to run the Mongo database is in a trusted environment, with no security and authentication"
- This "is the default option and is recommended"

Verizon 2017-2018 Data Breach Investigations Report

Threats to DB Security

1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats
4. DB injection attacks
5. ?

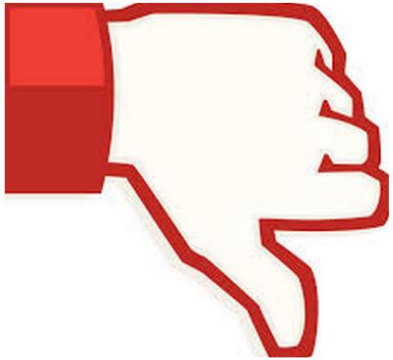


Threats to DB Security

1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats
4. DB injection attacks
5. Unmanaged sensitive data
   - Storing sensitive data unprotected

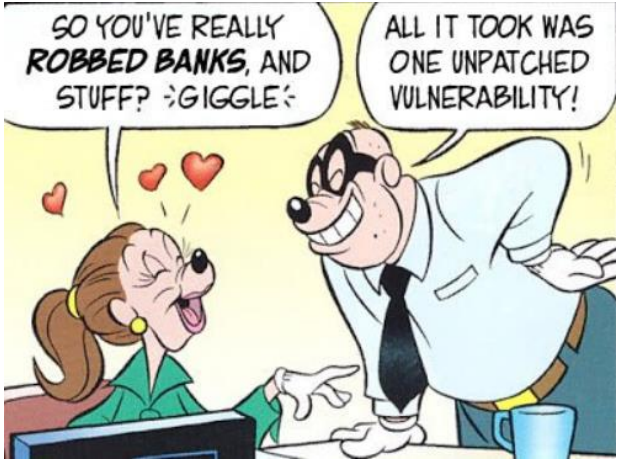
21 Facebook Stored Hundreds of Millions of User Passwords in Plain Text for Years
MAR 19

Hundreds of millions of **Facebook** users had their account passwords stored in plain text and searchable by thousands of Facebook employees — in some cases going back to 2012, KrebsOnSecurity has learned. Facebook says an ongoing investigation has so far found no indication that employees have abused access to this data.



Threats to DB Security

1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats
4. DB injection attacks
5. Unmanaged sensitive data
6. ?



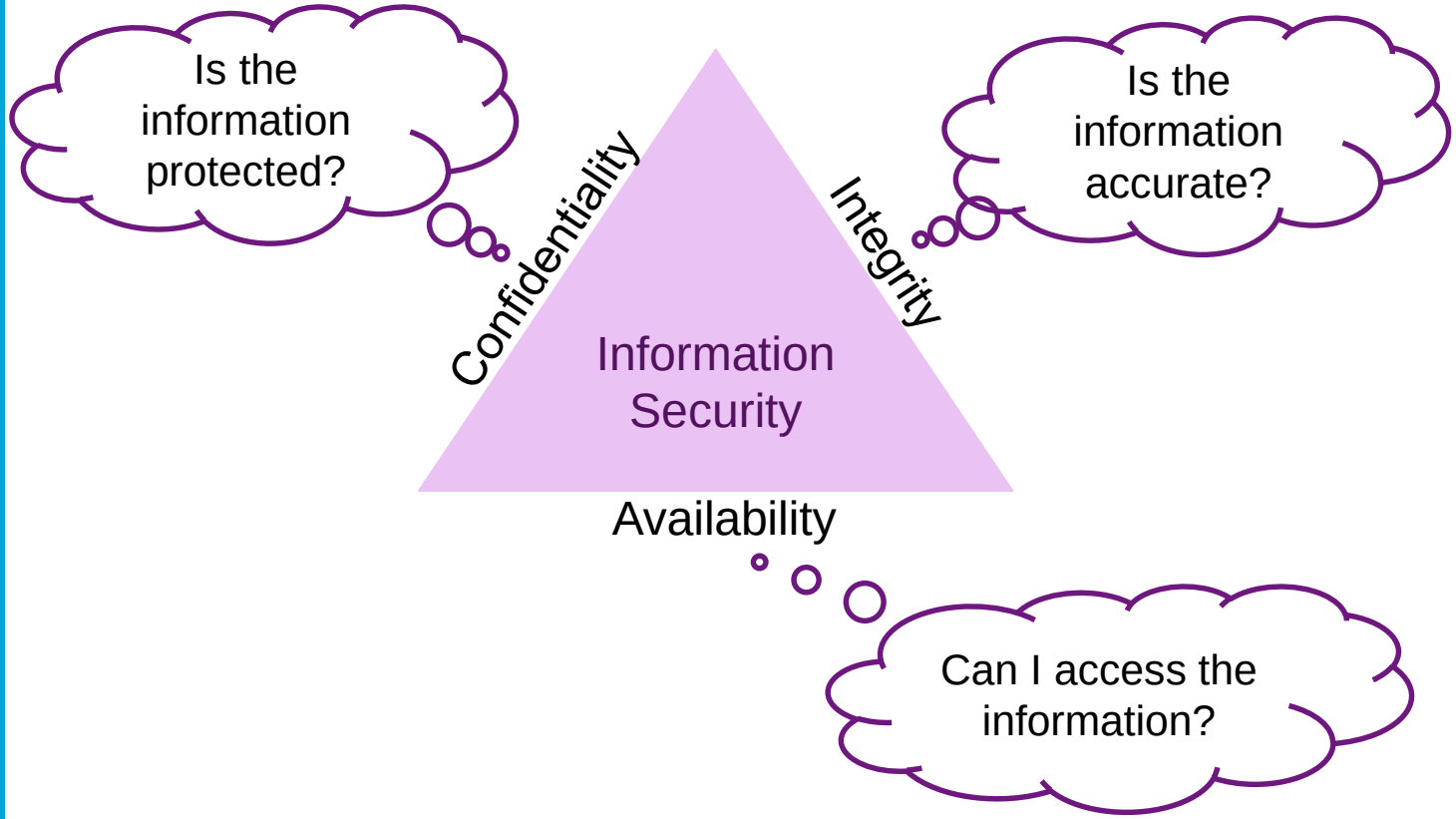
Threats to DB Security

1. Weak authentication
2. Excessive database privileges
3. (Inadvertent) Insider threats
4. DB injection attacks
5. Unmanaged sensitive data
6. Vulnerable DBs
   - or unpatched Operating System
   - Causing DoS attack

Extra notes:
Equifax (credit risk assessment) had a major breach exposing personal information of about 143M people. The breach of was caused due to an unpatched apache web server.

TU Delft

The CIA triad



Extra notes:
Encrypting the entire database and performing encrypted query operations is expensive and may not be feasible in all settings.
Read more about it: 1) https://en.wikipedia.org/wiki/Database_encryption
2) https://arxiv.org/abs/1512.03498

Mitigating DB security threats

- Encrypting databases
  - Data-in-transit
  - Data-at-rest
- Never use default usernames/passwords
- Use 2nd Factor Authentication
- Least privilege – need-to-know basis
- Log everything!!
- Update everything regularly
- Maintaining regular backups in air gapped environment
- Disable public error reporting
- Messy architecture means difficult maintenance
- Employee awareness – humans are the weakest link

TU Delft

Summary Part I

- Databases are the heart of an organization
- Information security – CIA triad
- Databases face a number of threats
  - Weak authentication and insider threats are the most common
- Awareness and simple security practices can mitigate those threats

TU Delft

Agenda for today

- Part I
  - Data breaches and their threat landscape
  - Information Security principles
  - Top threats for databases
  - Mitigating security threats
- Part II
  - SQL injection attacks
  - Injecting SQL queries ← Hands-on!
  - Analysing SQLi attacks
  - Best practices to avoid SQLi

SQL Injection

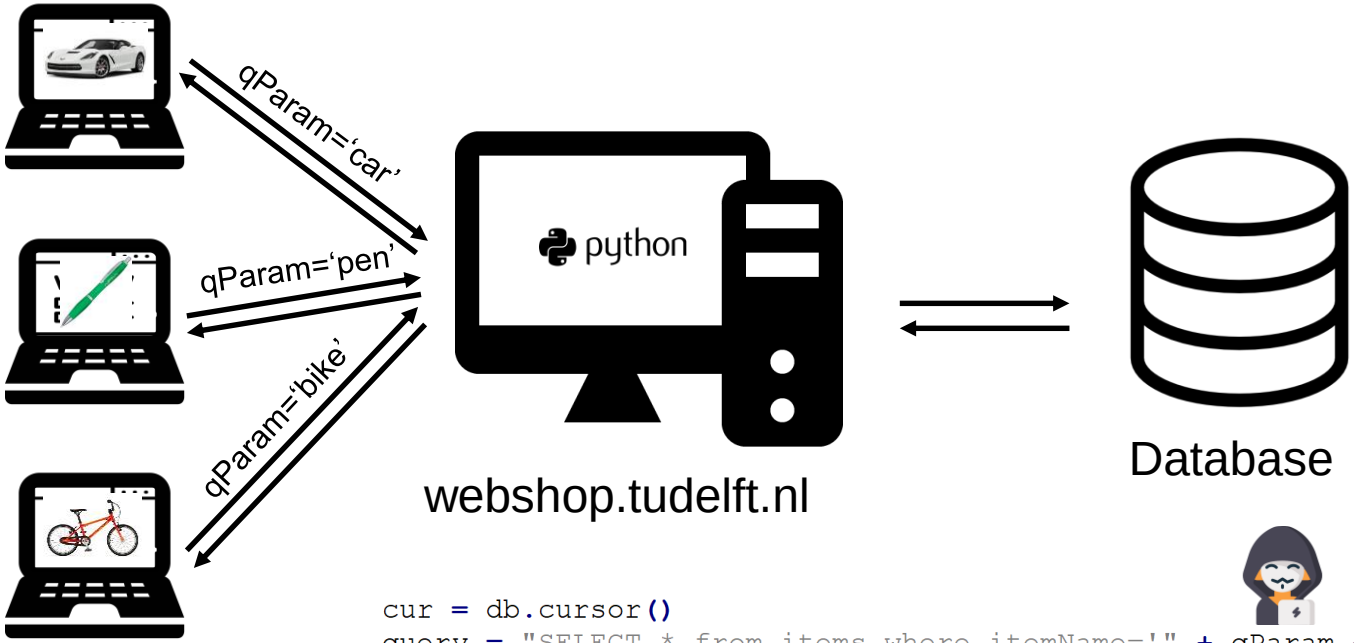
- SQL Injection (SQLi) refers to an **injection attack** wherein an attacker can **execute malicious SQL statements** that control a web application's **database server** (also known as *RDBMS*).

- Look out if you have:
  - Web application
  - Data stored in databases
  - User-controlled parameters

Extra notes:
Can affect any website or web application that makes use of an SQL-based database, so this vulnerability is one of the oldest, most prevalent and most dangerous of web application vulnerabilities.

https://www.acunetix.com/websitesecurity/sql-injection/

How does a typical web app work?




```
cur = db.cursor()
query = "SELECT * from items where itemName='" + qParam + "';"
cur.execute(query)
```

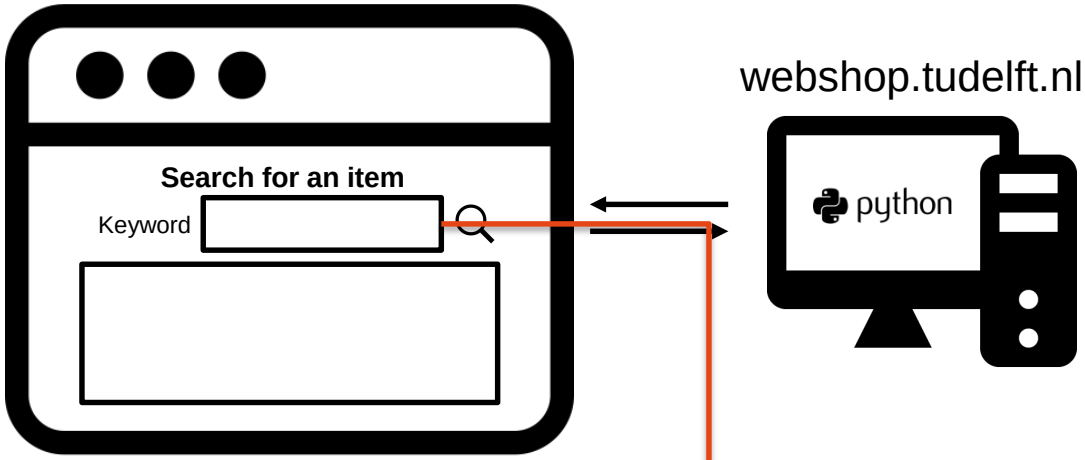
Extra notes:
Who is to blame?
- Database developers? Oracle?
- Web developer?
- Schema designers?

What can attackers do?

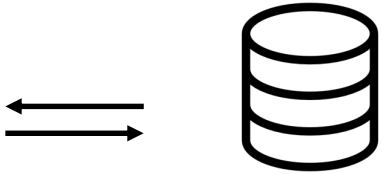
- Insert backdoor
  - INSERT INTO users (username, password) VALUES (‘attacker’, ‘youvebeenhacked’)
- Steal information
  - SELECT * FROM users WHERE userType=‘admin’
- Delete records/tables
  - DELETE FROM users;
  - DROP SCHEMA webshop;

TU Delft

Scenario



webshop.tudelft.nl



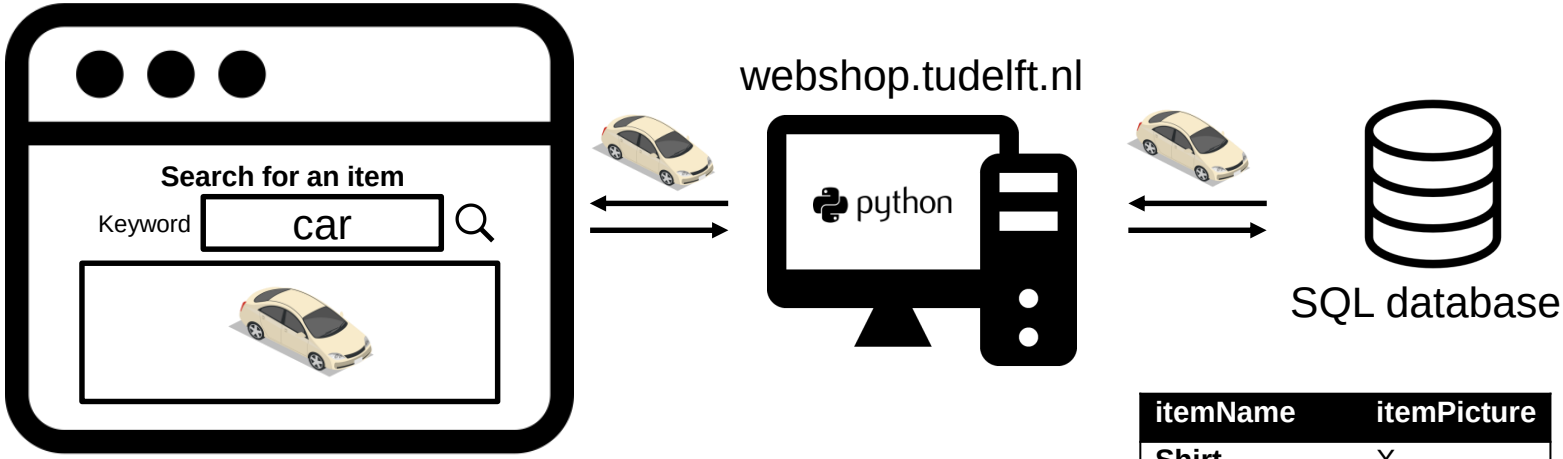
SQL database

```
"SELECT itemPicture FROM inventory WHERE
   itemName LIKE '" + [                ] + "';"
```

itemName	itemPicture
Shirt	X
Pen	X
Car	X

Inventory

Scenario

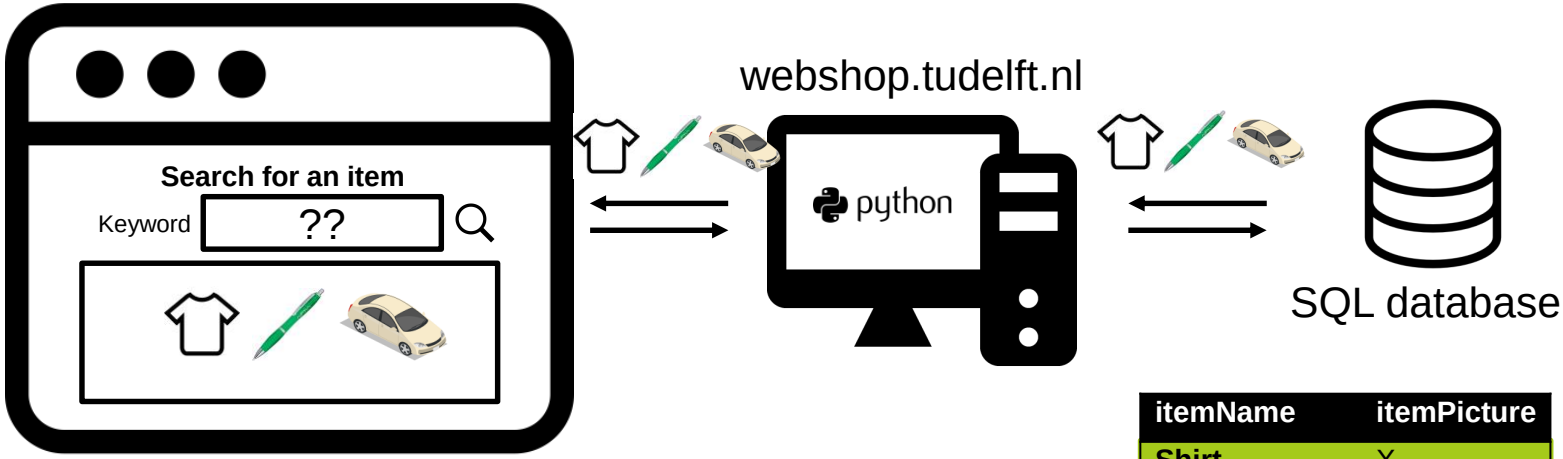


itemName	itemPicture
Shirt	X
Pen	X
Car	X

Inventory

```
"SELECT itemPicture FROM inventory WHERE
   itemName LIKE '" + car + "';"
```

Task1: How to list all items?



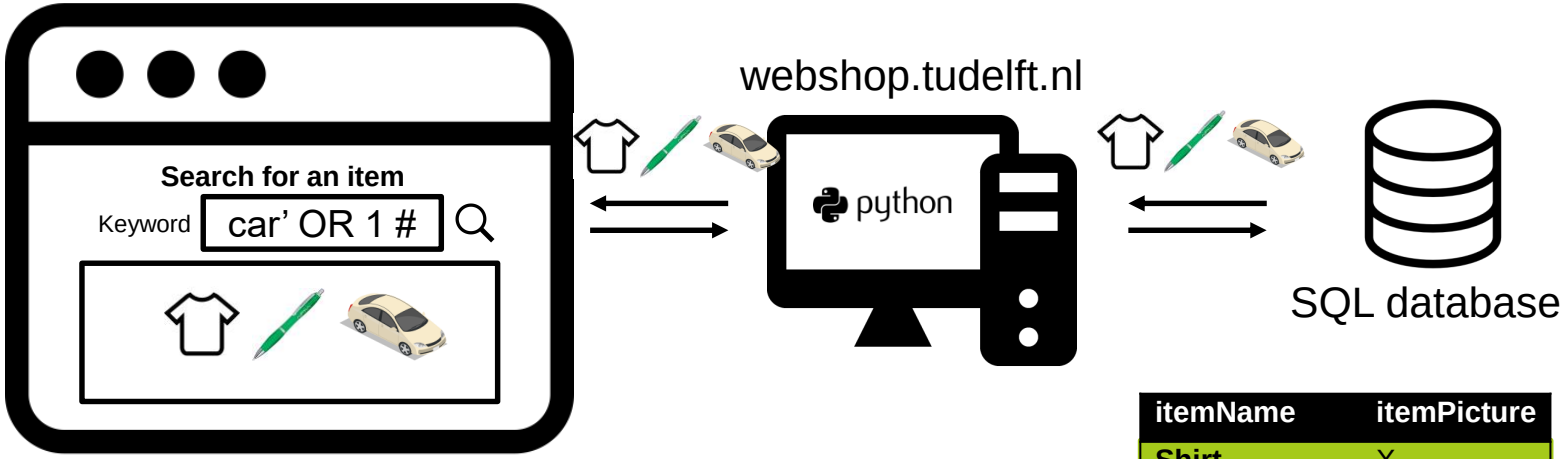
itemName	itemPicture
Shirt	X
Pen	X
Car	X

Inventory

```
"SELECT itemPicture FROM inventory WHERE
   itemName LIKE '" + ?? + "';"
```

Task1: How to list all items?

Room: IDMQ3



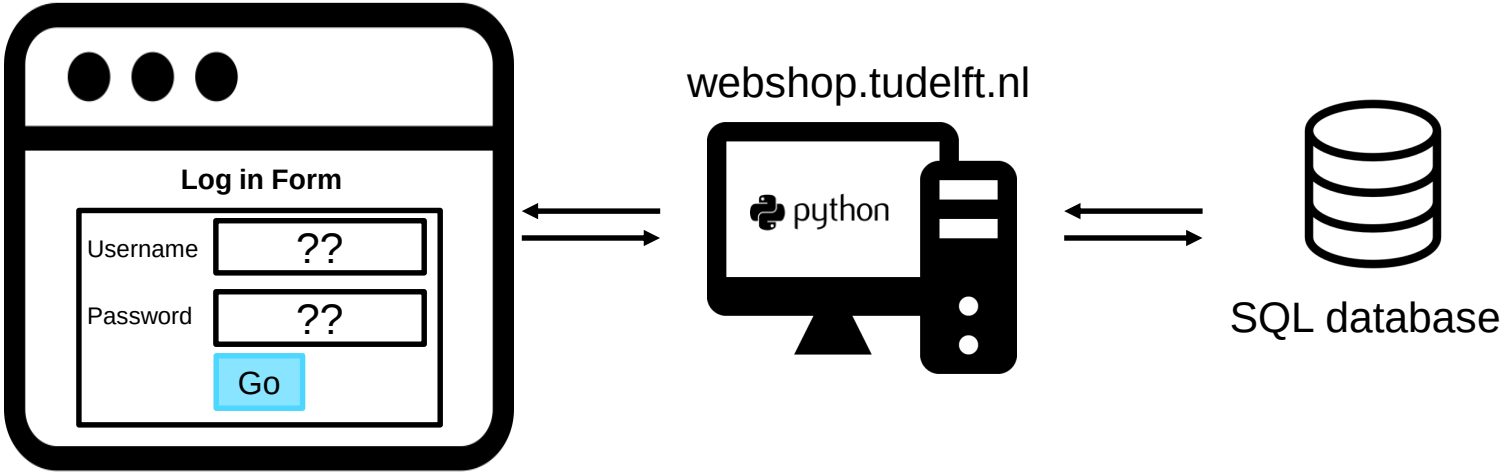
itemName	itemPicture
Shirt	X
Pen	X
Car	X

Inventory

```
SELECT itemPicture FROM inventory WHERE
 itemName LIKE 'car' OR 1 # ';
```

→ **Tautology**

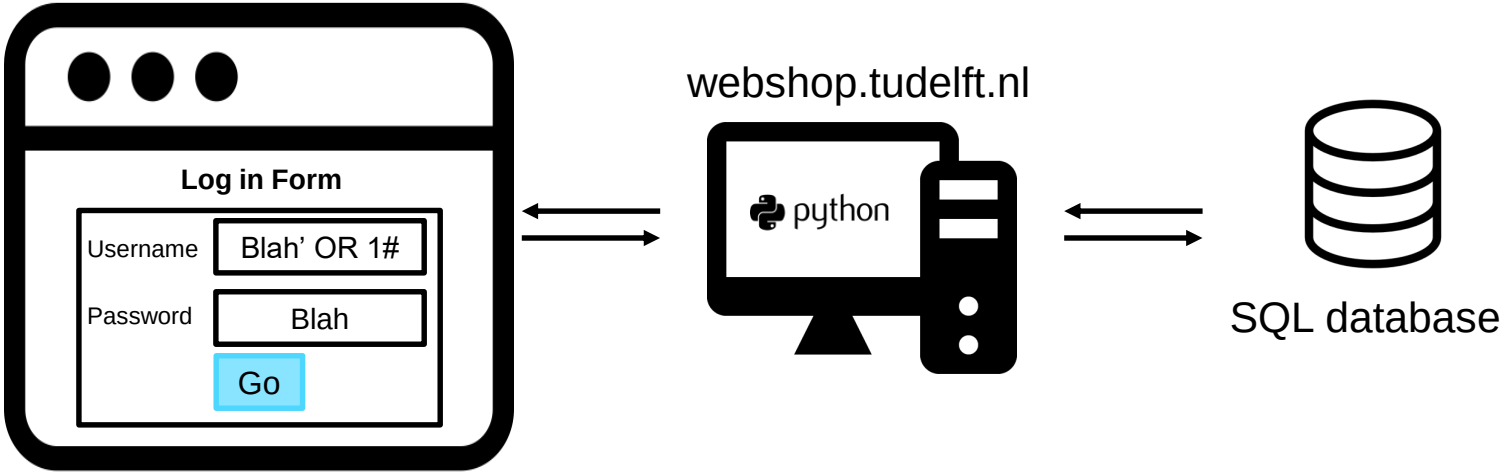
The login scenario…




```
"SELECT status FROM users WHERE
   username'" + ?? + "' AND
   password='" + ?? + "';"
```

Extra notes:
= missing after username

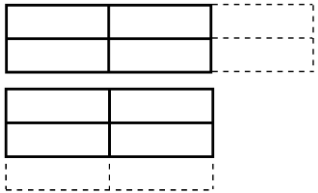
Another Tautology-based SQLi




```
SELECT status FROM users WHERE
 username= 'Blah' OR 1 # ' AND
 password= 'Blah';
```

Running multiple queries

- Useful keywords:
  - JOIN (Append horizontally)
  - UNION (Append vertically)



- `SELECT firstName, lastName FROM users;`

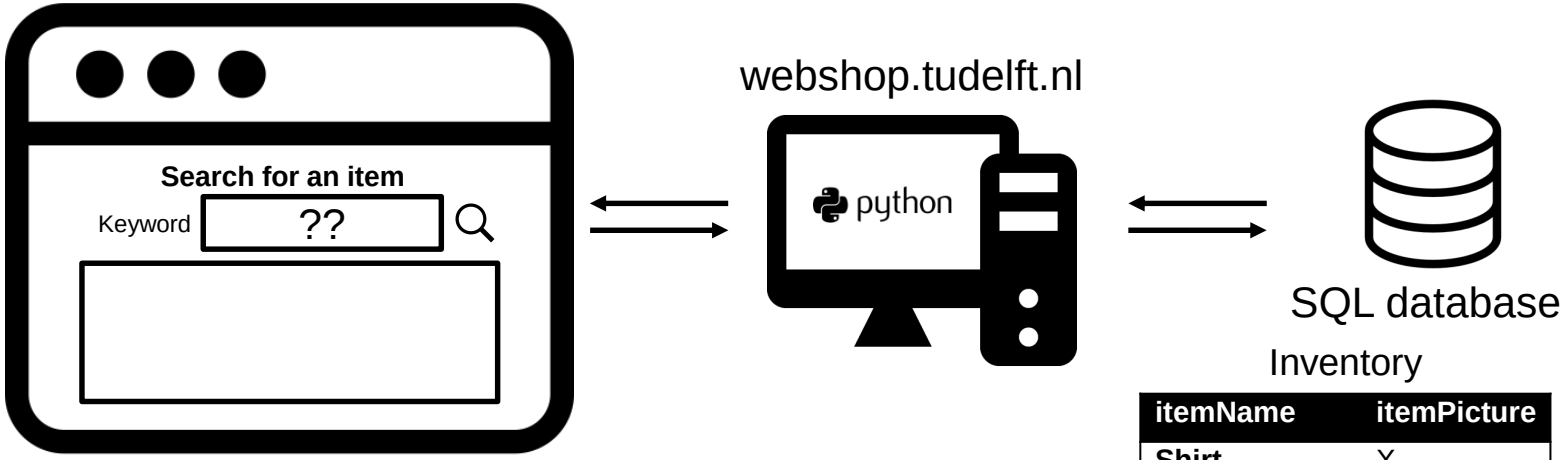
Fluffy	Bunny

- `SELECT firstName, lastName FROM users`
  `UNION`
  `SELECT 1,2 from dual;`

Fluffy	Bunny
1	2

Extra notes:
Dual is a one row, one column table in Oracle databases, called Dummy with value X.

Task 2: How to dump user data?



Inventory

itemName	itemPicture
Shirt	X
Pen	X
Car	X

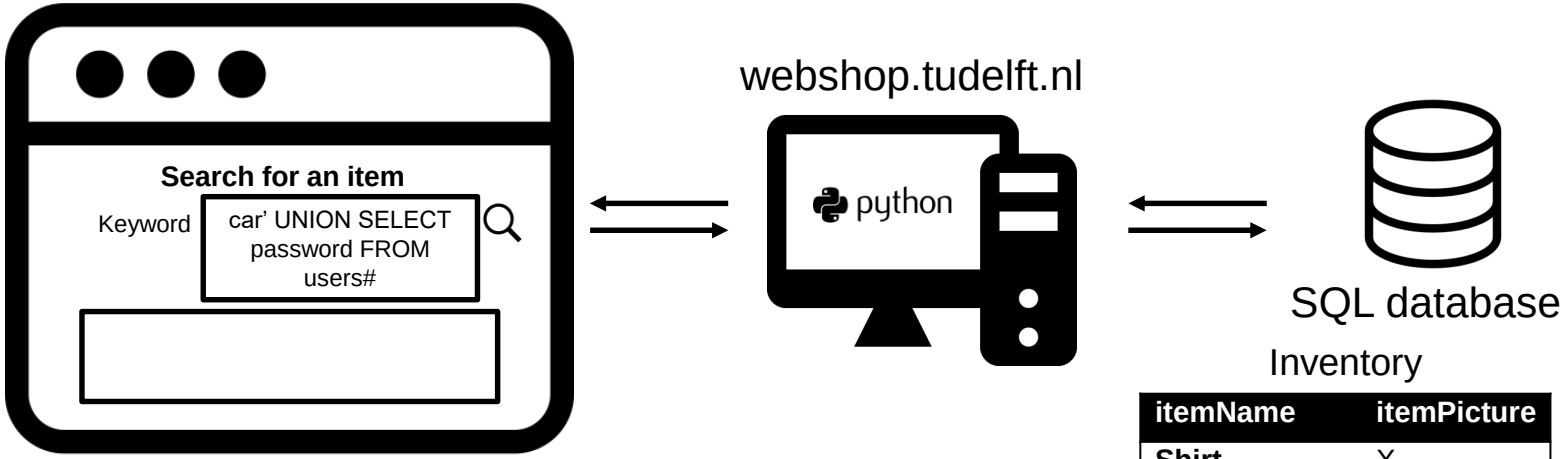
```
"SELECT itemPicture FROM inventory WHERE
   itemName LIKE '" + ?? + "';"
```

Users

username	password
fluffyBunny	cArR0T
admin	admin123

Task 2: How to dump user data?

Room: IDMQ3



Inventory

itemName	itemPicture
Shirt	X
Pen	X
Car	X

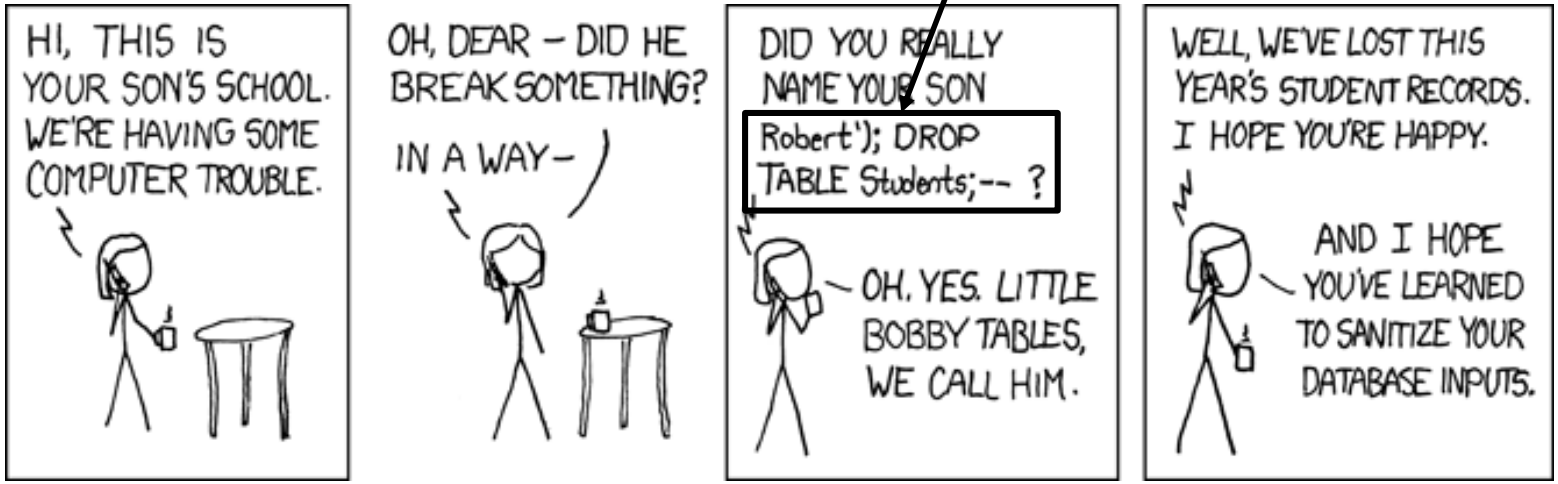
```
SELECT itemPicture FROM inventory WHERE
 itemName LIKE 'car' UNION
 SELECT password from users#';
```

Users

username	password
fluffyBunny	cArR0T
admin	admin123

Extra notes:
It's called: Union-based SQLi attack

Piggy backed query



https://www.explainxkcd.com/wiki/index.php/327:_Exploits_of_a_Mom

Why is it happening?

- Mixing of Code and data

```
SELECT profile FROM users WHERE
uname= ‘Blah‘ AND pwd= ‘Blah‘
```

```
SELECT profile FROM users WHERE
uname=‘Blah‘ OR 1=1 # ‘ AND pwd= ‘Blah‘
```

TU Delft

Why is it happening?

- Mixing of Code and data

```
SELECT profile FROM users WHERE
uname= ‘Blah‘ AND pwd= ‘Blah‘
```

```
SELECT profile FROM users WHERE
uname=‘Blah‘ OR 1=1 # ‘ AND pwd= ‘Blah‘
```

SQLi Avoidance

1. Input sanitization
   - Clean the input in order to use it

```
if re.match("^[a-zA-Z0-9_]*$", inputField):
    replaced = re.sub("^[a-zA-Z0-9_]*$", '', inputField)
inputField = replaced
```

- Problem:
  - Not all scenarios are known

SQLi Avoidance

1. Input sanitization
2. Escaping the input
   - To avoid *data* being mistaken as *code*
   - Input: *'what is 'www"*
   - Processed as: `'what is 'www''`
   - Must be processed as: `'what is \'www\''`

- Problem:
  - Possibly a 2nd Order SQLi attack
    - Effect not seen immediately

TU Delft

2nd Order SQLi

Users

username	password
fluffyBunny	CARROT
admin	admin123
Robert'; Drop table users;#	Blah

1)

Username: Robert'; Drop table users;#

Password: Blah

Register

```
INSERT INTO users values
('Robert\'; Drop table users;\#', 'Blah');
```

2)

Username: Robert'; Drop table users;#

Password: Blah

Login

```
select profile from users where
username='Robert\'; Drop table users;#'
and password='Blah';
```

Welcome, *Robert'; Drop table users;#*

3)

Password: Blah2

Confirm: Blah2

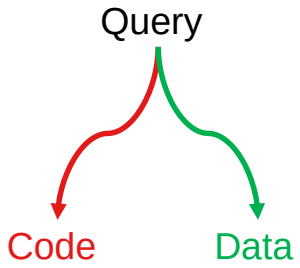
Update password

```
update users set password='Blah2'
where username='Robert'; Drop table users;#';
```

TU Delft

SQLi Avoidance

1. Input sanitization
2. Escaping the input
3. Prepared statements
   - Separation of concerns
   - Pre-compile legitimate query
   - Add placeholders for *data*




```
statement = con.prepareStatement("SELECT profile FROM users where userName=?");
statement.setString(1, inputField)
statement.executeUpdate();
```

Extra notes:
Learn more about Prepared statements here: https://youtu.be/jTasm64rz-c and https://stackoverflow.com/questions/23845383/what-does-it-mean-when-i-say-prepared-statement-is-pre-compiled

Summary Part II

- Executing SQL code on a database is called an SQL Injection attack
- SQLi is caused by mixing of code and data
- Prepared statements are *the most* useful in avoiding SQLi
- However, user input must always be sanitized

Extra notes:
Prepared statements can be used in all cases EXCEPT when using Dynamic Object Mappers (e.g. Hibernate, Jackson) because we don't have variables to bind with beforehand. In such cases, escaping and sanitizing user input are the only options.

Additional material

- https://www.esecurityplanet.com/network-security/6-database-security-best-practices.html
- NoSQL injection attacks:
  - https://www.owasp.org/images/e/ed/GOD16-NOSQL.pdf
  - https://www.owasp.org/index.php/Testing_for_NoSQL_injection
  - http://blogs.adobe.com/security/files/2011/04/NoSQL-But-Even-Less-Security.pdf?file=2011/04/NoSQL-But-Even-Less-Security.pdf
- https://codecurmudgeon.com/wp/sql-injection-hall-of-shame/
- Type of SQLi attacks: https://pdfs.semanticscholar.org/81a5/02b52485e52713ccab6d260f15871c2acdcb.pdf
- Try it yourself:
  - https://www.codingame.com/playgrounds/154/sql-injection-demo/sql-injection
  - http://leettime.net/sqlninja.com/
  - https://www.veracode.com/security/sql-injection

TU Delft

Time for questions