App Dev

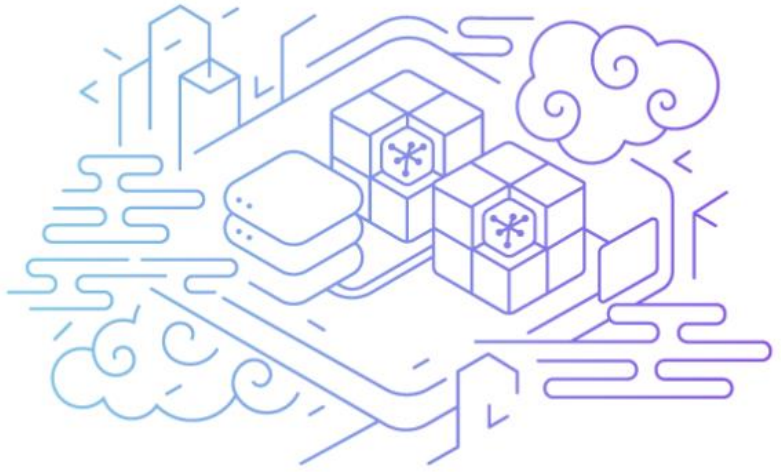
Stefano Balietti

Center for European Social Science Research at Mannheim University (MZES)
Alfred-Weber Institute of Economics at Heidelberg University

@balietti | stefanobalietti.com | @nodegameorg | nodegame.org

Building Digital Skills: 5-14 May 2021, University of Luzern

Module 9: Web Server



HEROKU

Express 4.17.1

Fast, unopinionated, minimalist web framework for Node.js

Module 9: Web Server

Learning Goals

- Setup the **ExpressJS** server in NodeJS
- Understanding middleware
- Creating custom routes and password protected routes
- Creating RESTful APIs
- Upload your server on the **Heroku** cloud
- Implement a simple Captcha

Module 9: Web Server

- **Express JS** provides a minimalistic and fast web server API to server static assets and templates
- Express JS is the most installed server in NodeJS ecosystem (a recent alternative *fastify*)
- Can be used in production in tandem with the **NginX** server (not covered here)
- **Heroku** is a cloud platform *as a service* supporting several programming languages.
- Heroku focuses on **apps** and provides the backend infrastructure to run them.

Routes

- Programming in Express means adding "routes" sequentially
- Each routes **catch a given path** in the URL (wildcards are possible)
- Each route takes a **callback** that manipulates two objects:
  - **request** from the browser (e.g., for checking IP or cookies)
  - **response** (e.g., send a file back)
- The **order** in which are added matters! Earlier routes are executed first

Routes Examples

```
// A route to intercept requests to home (root).
app.get('/', (req, res) => {
  // Something here
});
```

Routes Examples

```
// A route to intercept requests to home (root).
app.get('/', (req, res) => {
  // Something here
});
```

The request object contains information about the request, such as:

- **req.body** (POST)
- **req.params** (GET)
- **req.cookies**
- **req.ip**

Routes Examples

```
// A route to intercept requests to home (root).
app.get('/', (req, res) => {
  // Something here
});
```

The request object contains information about the request, such as:

- **req.body** (POST)
- **req.params** (GET)
- **req.cookies**
- **req.ip**

The response object contains methods to handle the request, such as:

- **res.cookie**
- **res.send**
- **res.sendFile**
- **res.status**
- **res.end**

Routes Examples

```
// A route to intercept requests to home (root).
app.get('/', (req, res) => {
  // A quick message.
  res.send('This is your home page.');
});
```

Routes Examples

```
// A route to intercept requests to home (root).
app.get('/', (req, res) => {
  // A quick message.
  res.send('This is your home page.');
});


// Mail route.
app.get('/mail', (req, res) => {
  res.sendFile('/path/to/mail/file.html');
});
```

Middlewares

```
// A route to intercept requests to home (root).
app.use('ROUTE', (req, res, next) => {
  // Do something (e.g., authenticate user).

  // Everything OK, go to the next middleware/route callback.
  next();
});
```

Middlewares

```
// A route to intercept requests to home (root).
app.use('ROUTE', (req, res, next) => {
  // Do something (e.g., authenticate user).

  // Everything OK, g                          ute callback.
  next();
});
```

For instance: **'/mail/'**
But it can also be skipped to be applied globally.

The localhost Address

After Express has started, it serves files from this address:

http://localhost:3000

ADDRESS

YOUR OWN COMPUTER (INSTEAD OF A REMOTE SERVER)

It is equivalent to the numeric IP: 127.0.0.1

PORT

A INTERNAL ADDRESS WITHIN YOUR COMPUTER (THINK IT AS THE DOOR BELL IN A LARGE BUILDING)

Cloud Providers

Netlify (PaaS): front-end development
Vercel (PaaS): in-between Netlify and Heroku
Heroku (PaaS): back-end development (apps sleep!)

Digital Ocean (BaaS and PaaS): you want to maintain your own virtual server at a very reasonable price.

Firebase (BaaS): access to Google services
Azure (BaaS): access to Microsoft services
AWS (BaaS): access to Amazon services

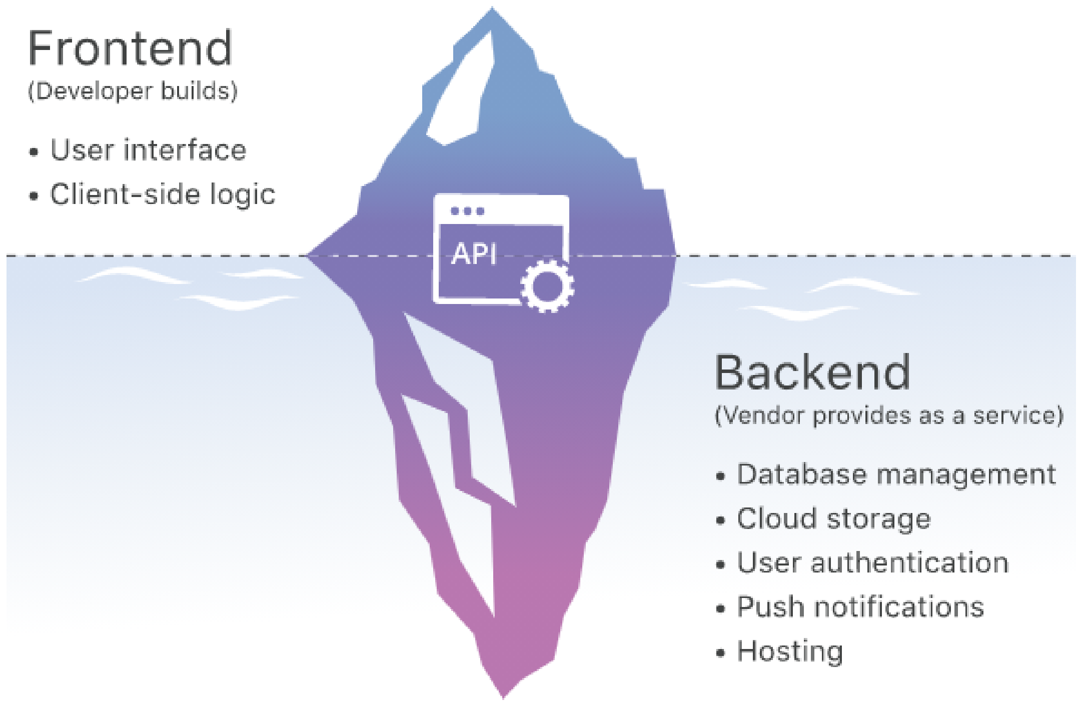


Image source

https://medium.com/@benwang_2362/the-difference-between-iaas-paas-baas-and-saas-91133d728917

Module 6: Resources

- https://expressjs.com/
- https://expressjs.com/en/4x/api.html
- https://scotch.io/tutorials/whats-new-in-expressjs-5-0
- https://www.geeksforgeeks.org/different-servers-in-node-js/
- https://javascript.plainenglish.io/fastify-express-benchmark-4c4aebb726d6
- https://www.heroku.com

Security

Learning Goals

- Best practices to secure Express server
- How to store passwords: encryption vs hashing
- How to tame bots: how detect them, honeypots and captchas

Securing Express

Express highlights in this page the best security practice

https://expressjs.com/en/advanced/best-practice-security.html

See exercise: 7_secure.js

Securing Express

1. **Never** use *deprecated* or *vulnerable* versions of Express

Keep an eye on the security update page

Securing Express

2. Always use *TLS* (Transport Layer Security), enabling **https://**

TLS encrypts all data before it is sent from the client to the server

Add a route to redirect all HTTP traffic to HTTPS

Remember! POST requests do not encrypt data, more secure than GET:

- are not cached/bookmarked/browser history
- much larger data length restriction (~8Kb vs ~2Gb, but can be configured)

Securing Express

3. In doubt, put the **Helmet** on!

Helmet is a package to automatically configure Express with a higher level of security

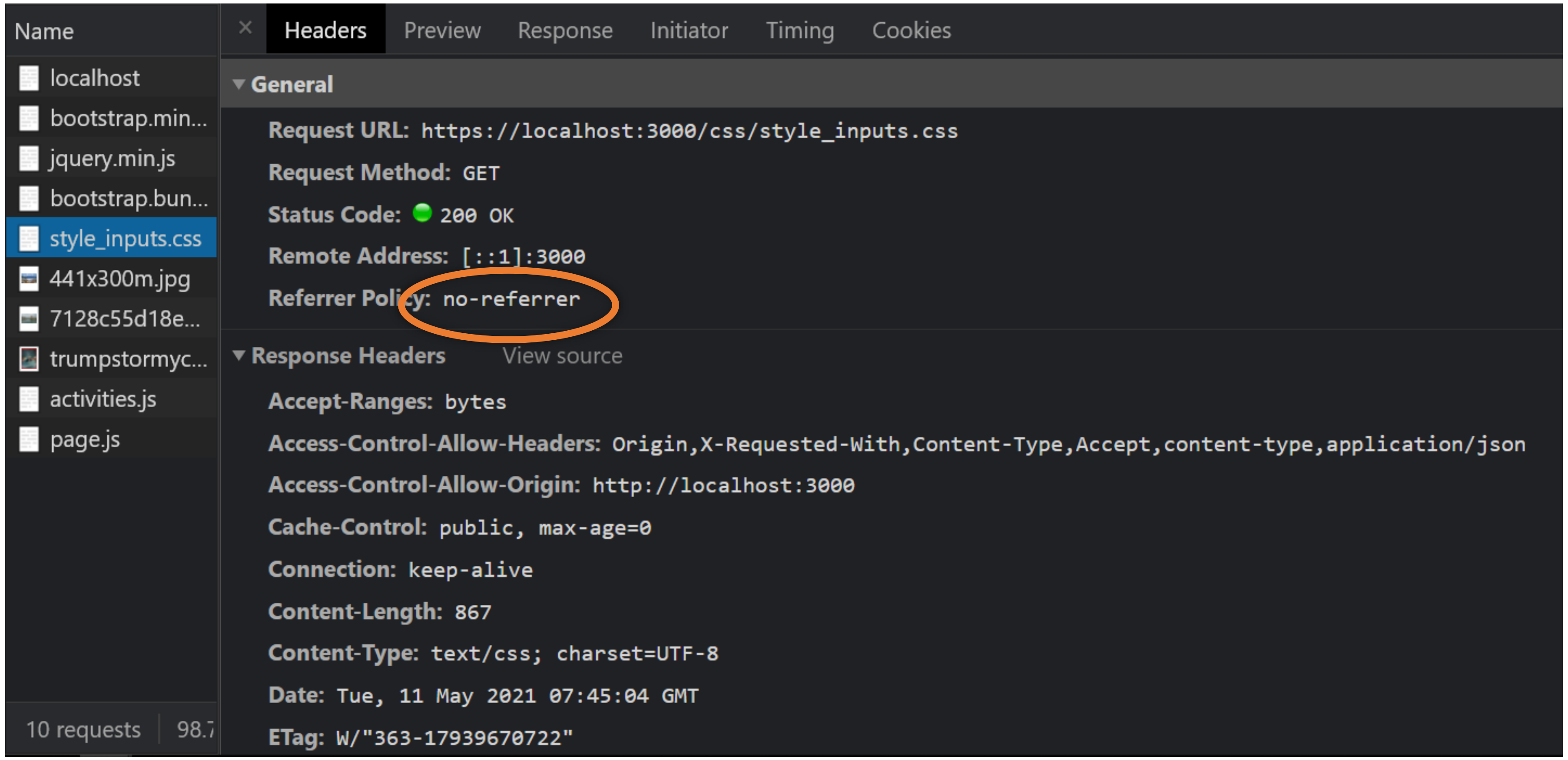
Default security can be too tight, i.e., disabling all scripts without a hash or a nonce attribute

At least remove the "Powered by Express" header.

A Deeper Look at the Headers

Open the DevTools, click on Network tab and click on a resource (e.g., style_input.css)

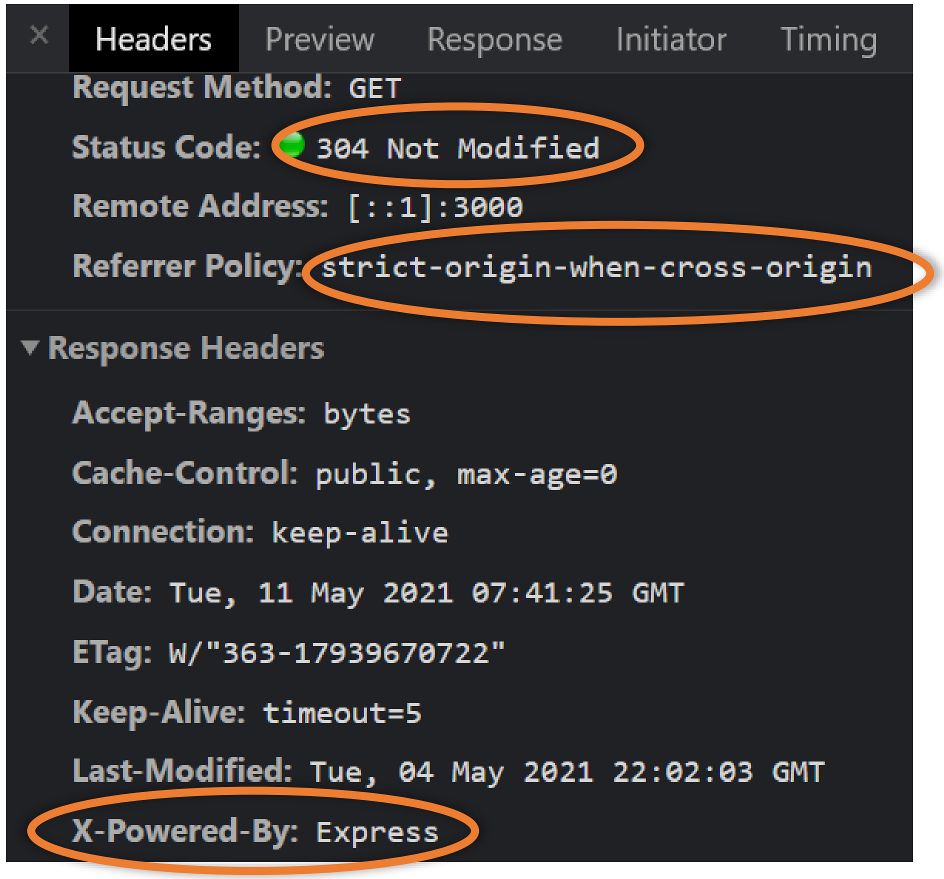
WITH HELMET



```
Request URL: https://localhost:3000/css/style_inputs.css
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:3000
Referrer Policy: no-referrer

Accept-Ranges: bytes
Access-Control-Allow-Headers: Origin,X-Requested-With,Content-Type,Accept,content-type,application/json
Access-Control-Allow-Origin: http://localhost:3000
Cache-Control: public, max-age=0
Connection: keep-alive
Content-Length: 867
Content-Type: text/css; charset=UTF-8
Date: Tue, 11 May 2021 07:45:04 GMT
ETag: W/"363-17939670722"
```

Scroll down for more headers

WITHOUT HELMET



```
Request Method: GET
Status Code: 304 Not Modified
Remote Address: [::1]:3000
Referrer Policy: strict-origin-when-cross-origin

Accept-Ranges: bytes
Cache-Control: public, max-age=0
Connection: keep-alive
Date: Tue, 11 May 2021 07:41:25 GMT
ETag: W/"363-17939670722"
Keep-Alive: timeout=5
Last-Modified: Tue, 04 May 2021 22:02:03 GMT
X-Powered-By: Express
```

Securing Express

4. Prevent brute-force attacks with a rate-limiter package

The **rate-limiter-flexible** package for instance lets you define a number of **points** that can be consumed within a given **duration.** You can associate actions to point consumption (to a logged user or to a IP address) and when it goes to zero access to resource is denied.

Securing Express

5. Whitelist IPs that can access the API

- Setting up **CORS**

- Manually **checking IP** access

Securing Express

5. Whitelist IPs that can access the API

- Setting up **CORS**:
  - The cors package handles it nicely
  - Can specify address, protocol, and port.
  - The web server still receives and responds to requests normally.
  - The browser will hide the responses if the CORS policy is not respected.
- Manually **checking IP** access:
  - Completely prevent access to the resource
  - IP can masked by a proxy (e.g., Nginx) check headers 'x-real-ip'

Securing Express

6. Create a **strong access key** for the API

Use the crypto module or the uuid package

Never store the password in plain, neither in separate file nor hardcoded in code

Create an hash that can be stored outside of codebase (e.g., database or fs)

Load the hash in memory and compare incoming requests

The bcrypt module is recommended for hashing

Must use TSL to encrypt API key in incoming requests.

Securely Authenticating Users

5. If you are dealing with authenticating users in the browser **JSON Web Tokens** are recommended. Playground: https://jwt.io/

- Store them as cookies and use the HttpOnly tag.
- httpOnly tag makes the cookie unavailable to JavaScript, they are just sent with the Headers on every request.

Securely Authenticating Users

5. If you are dealing with authenticating users in the browser **JSON Web Tokens** are recommended. Playground: https://jwt.io/

- Store them as cookies and use the HttpOnly tag.
- httpOnly tag makes the cookie unavailable to JavaScript, they are just sent with the Headers on every request.

Or you may us a **plugin** auth:

- https://auth0.com/ (Max 7000 active users for free)
- https://magic.link/ (Max 100 active users for free)

Useful Packages

- The **passport package** makes it easier to integrate different authorization methods (including oauth)

- The **pm2 package** spawns a NodeJS child process in the background **(MUST)**.
  - If you run the Express server yourself you need to make sure the server is kept-alive after you close the connection/terminal.
  - Pm2 will restart the server upon errors, handle memory limits, and help you configure the node.js process

- The **nodemon package** will watch your files and restart your server whenever there is a change. Very very useful when developing a server application.

- The **dotenv package** is a simple package to load data (e.g., passwords and keys) into the NodeJS process

Securing Express: Resources

- https://expressjs.com/en/advanced/best-practice-security.html
- https://dev.to/omergulen/step-by-step-node-express-ssl-certificate-run-https-server-from-scratch-in-5-steps-5b87
- https://blog.gitguardian.com/secrets-api-management/

Password management

Adobe Data Breach

- In 2013 Adobe announced a huge data breach that affected 3.000.000 to 38.000.000 customers.

- A huge dump of the offending customer database was recently published online.

- In the statement Adobe announce that:

“The attackers may have gained access to your… encrypted password.”

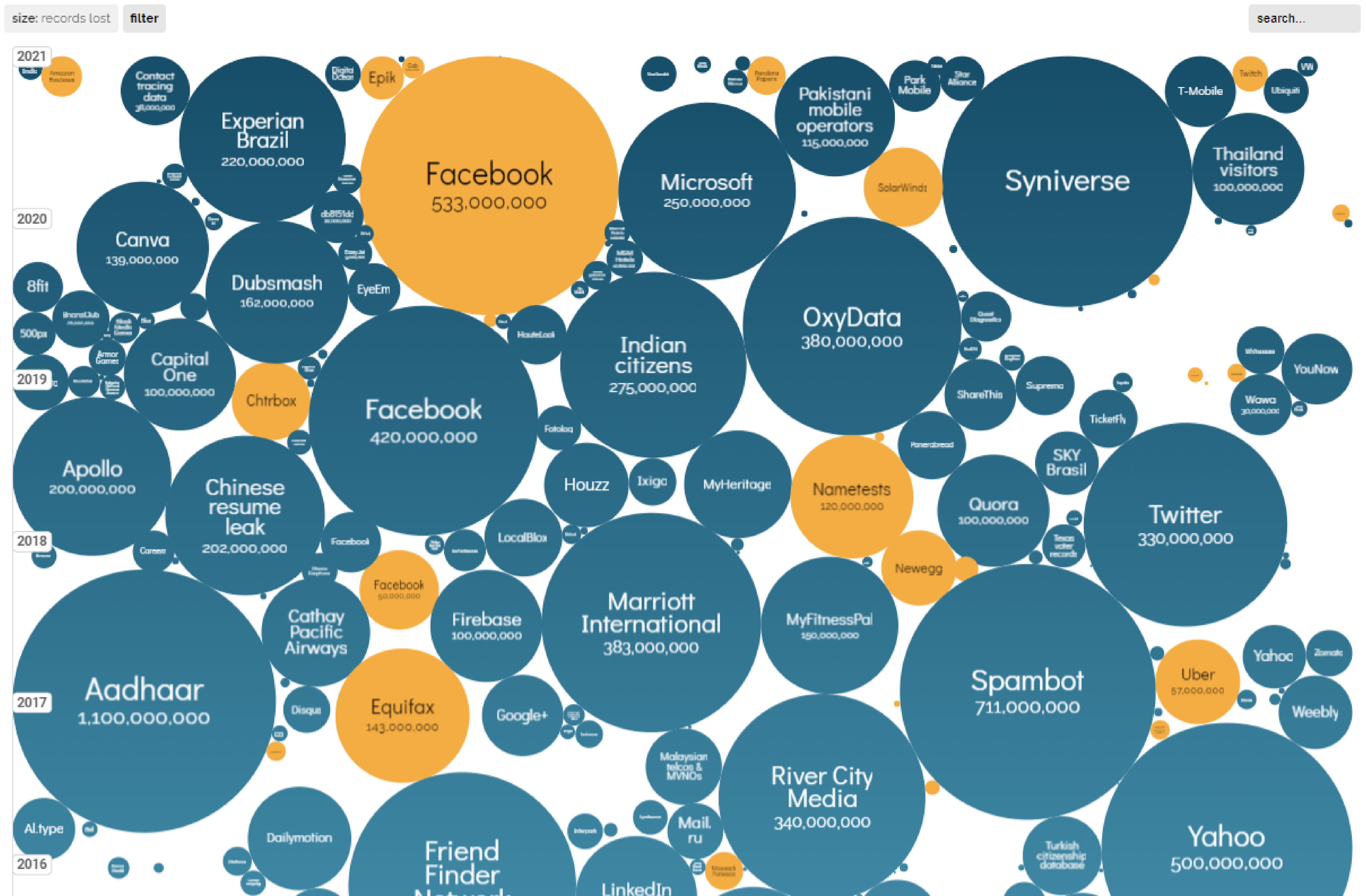
- But what is encrypted password mean ?

Adobe Hacked, Source: https://www.venzagroup.com/adobe-systems-data-breach-compromises-information-millions-users/

World's Biggest Data Breaches & Hacks

Selected events over 30,000 records

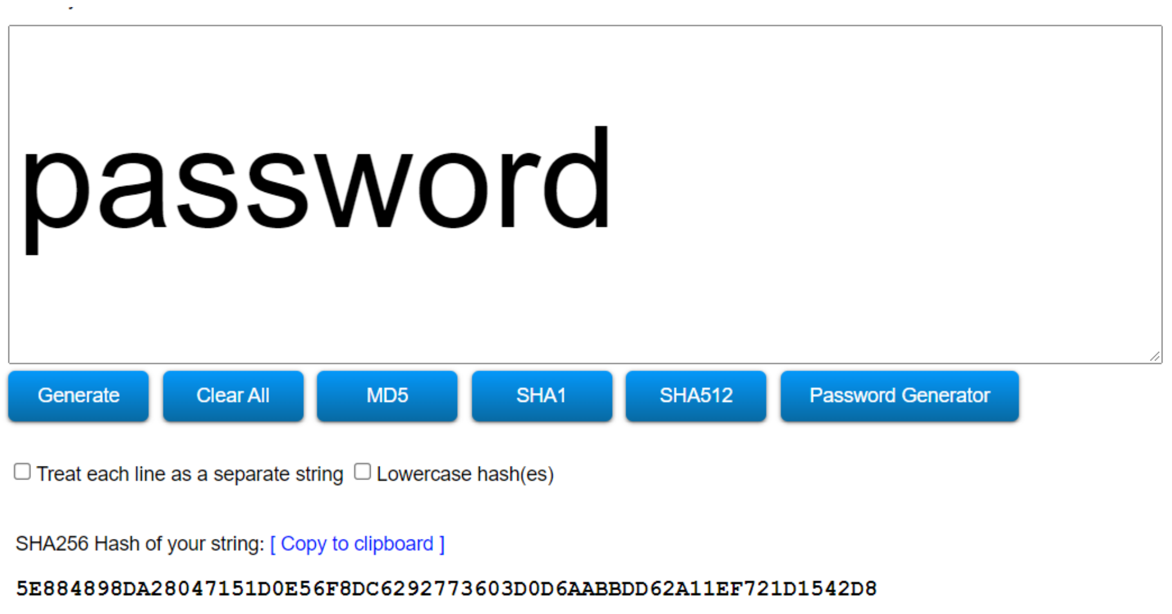
UPDATED: Oct 2021



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

Encrypted Password ?

- For password storage **do not use encryption**
- Use a **one-way mathematical function** called a **hash** that uniquely depends on the password
- You can calculate the hash from the password, but not the other way around



password

☐ Treat each line as a separate string ☐ Lowercase hash(es)

SHA256 Hash of your string: [Copy to clipboard]

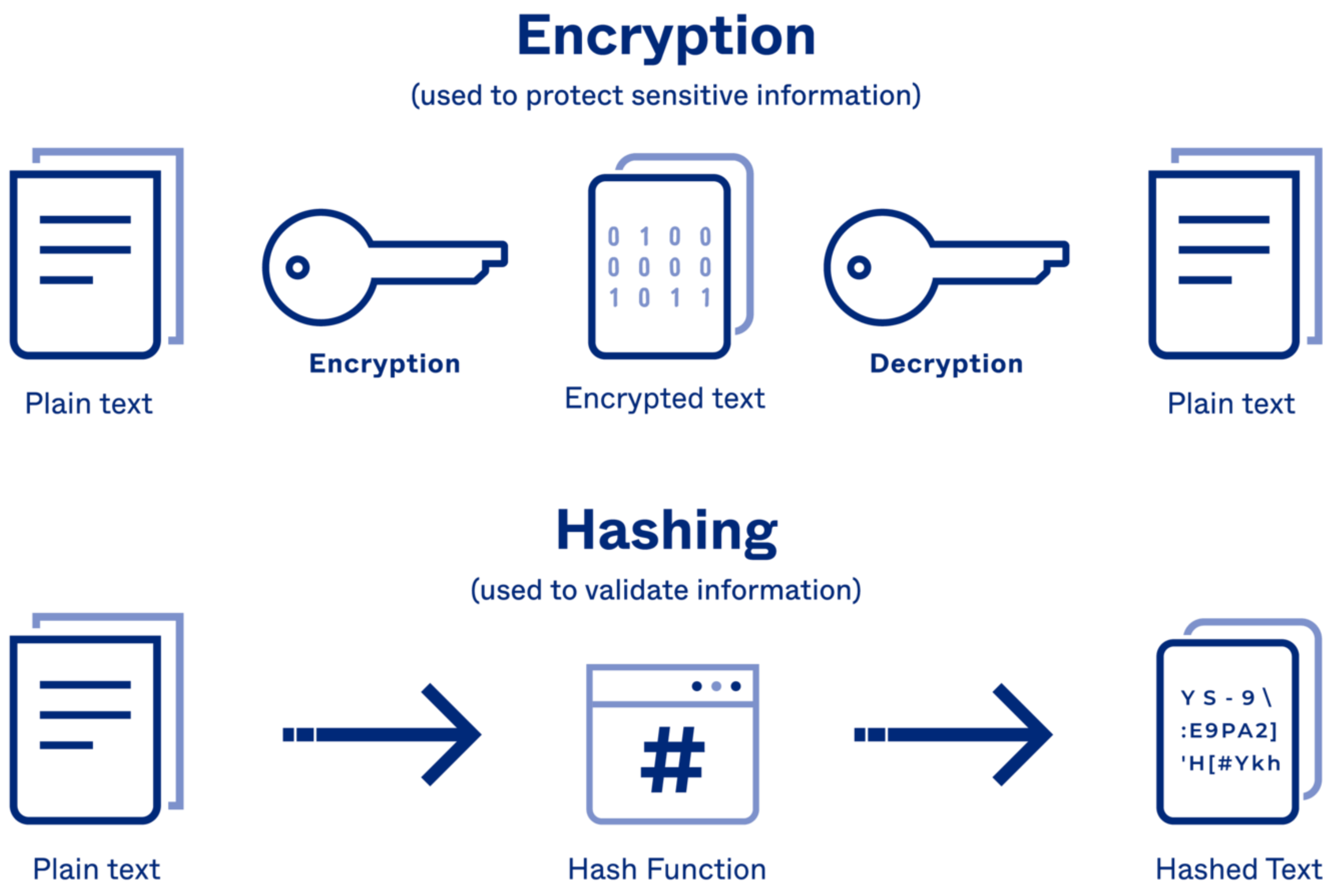
```
5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
```

https://passwordsgenerator.net/sha256-hash-generator/

Encrypted Password ?

- You **never actually store the password at all**, password **processed in memory** to verify hash
- To make it more secure you usually add some ***salt***: a random string that you store with the user's ID and mix into the password when you compute the hash.
- Finally you also use a **key derivation function (KDF)** like **bcrypt.**

Encrypted Password ?



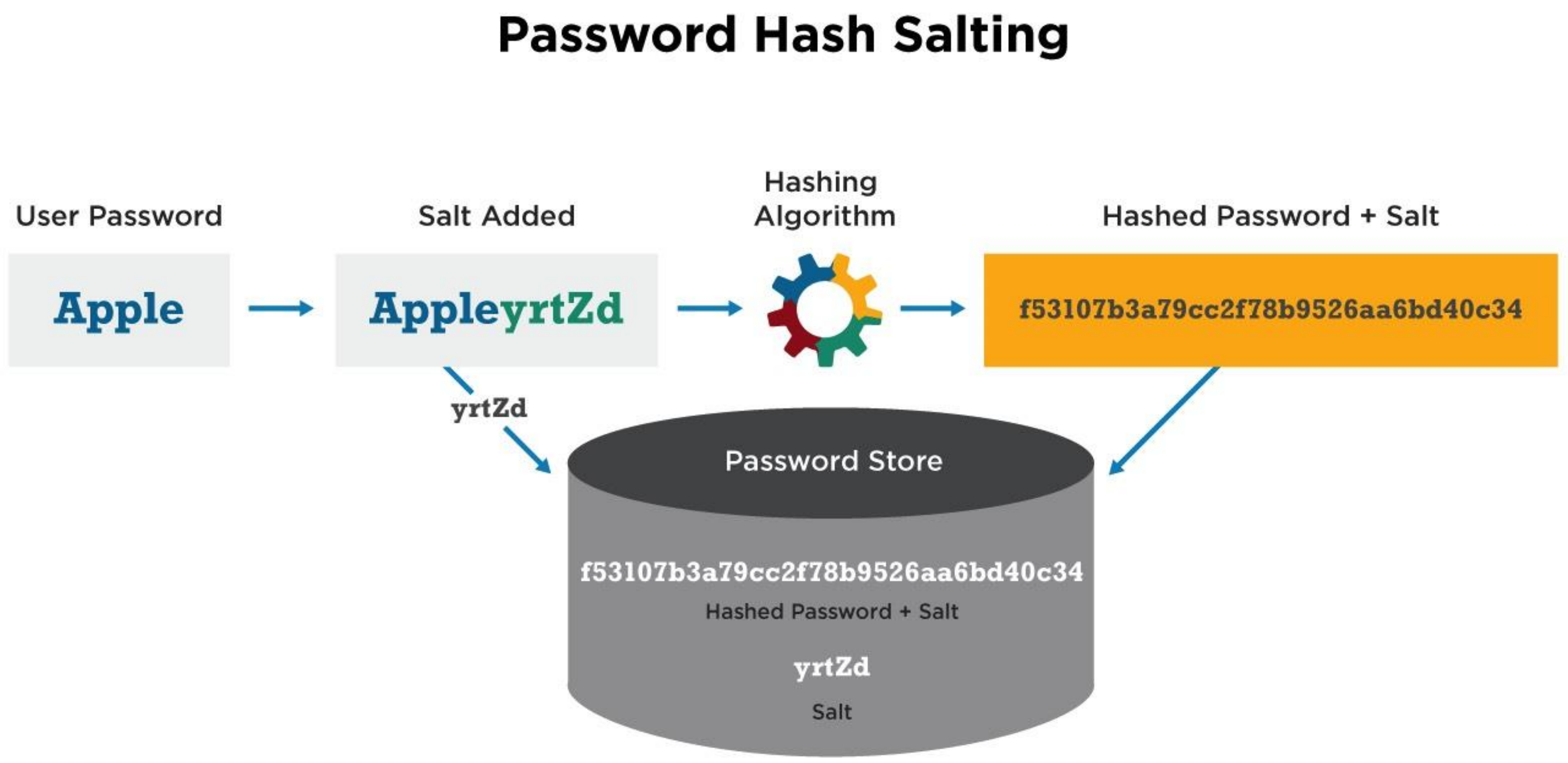
okta

Encryption vs Hashing, Source: https://www.okta.com/identity-101/hashing-vs-encryption/

Hashing and Salting

- A **hash function** maps data of arbitrary size **to fixed-size values**
- Hashed passwords are **deterministic**: same input -> same output
- This makes hashed passwords vulnerable to a **dictionary attack**
- A **cryptographic salt** can mitigate this risk
- Salt are **random bits** added to each password instance before its hashing
- **Salts create unique passwords** even in the instance of two users choosing the same passwords

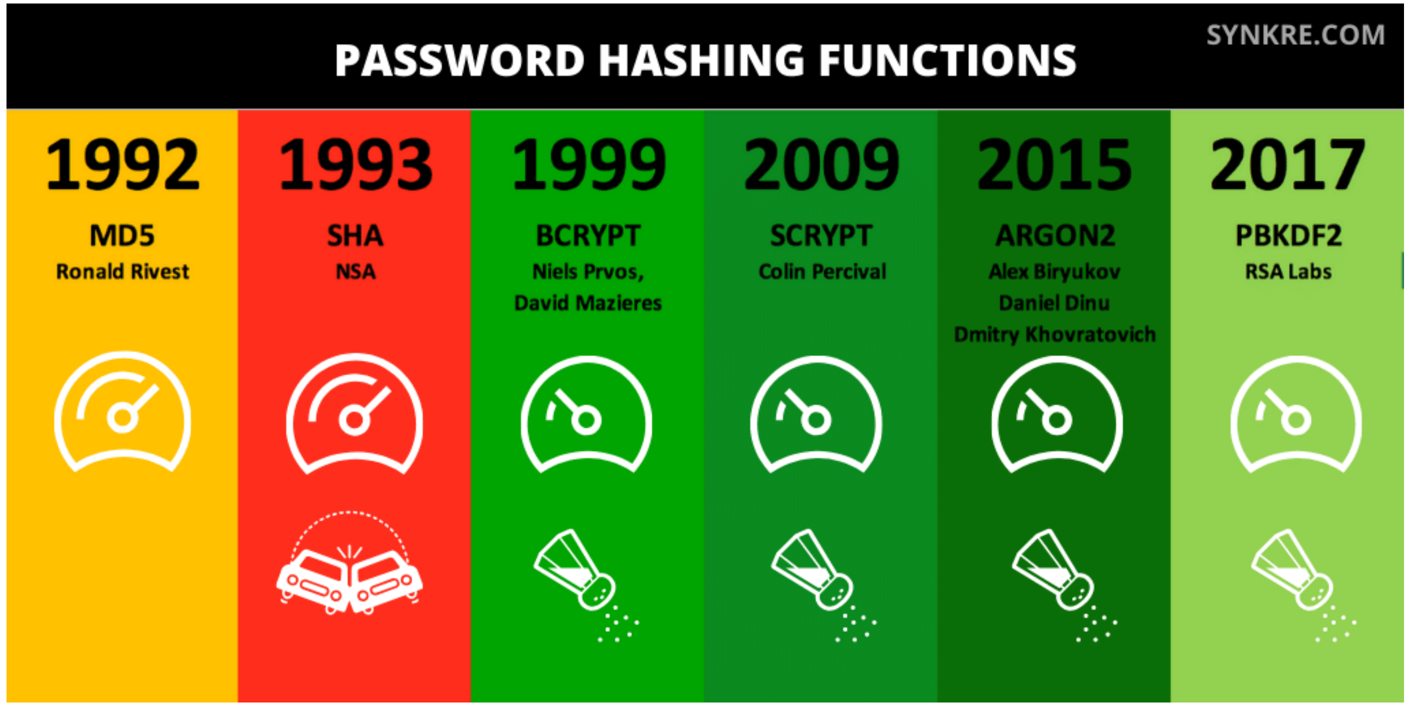
Hashing and Salting



Password Hash Salting, Source: https://cyberhoot.com/cybrary/password-salting/

bcrypt

- bcrypt is a **key derivation function** that based on **Blowfish cipher** and **crypt** hash function.
- The safety of the password depends on how fast the selected cryptographic hashing function can calculate the password hash.
- A fast function would execute faster when running in much more powerful hardware.

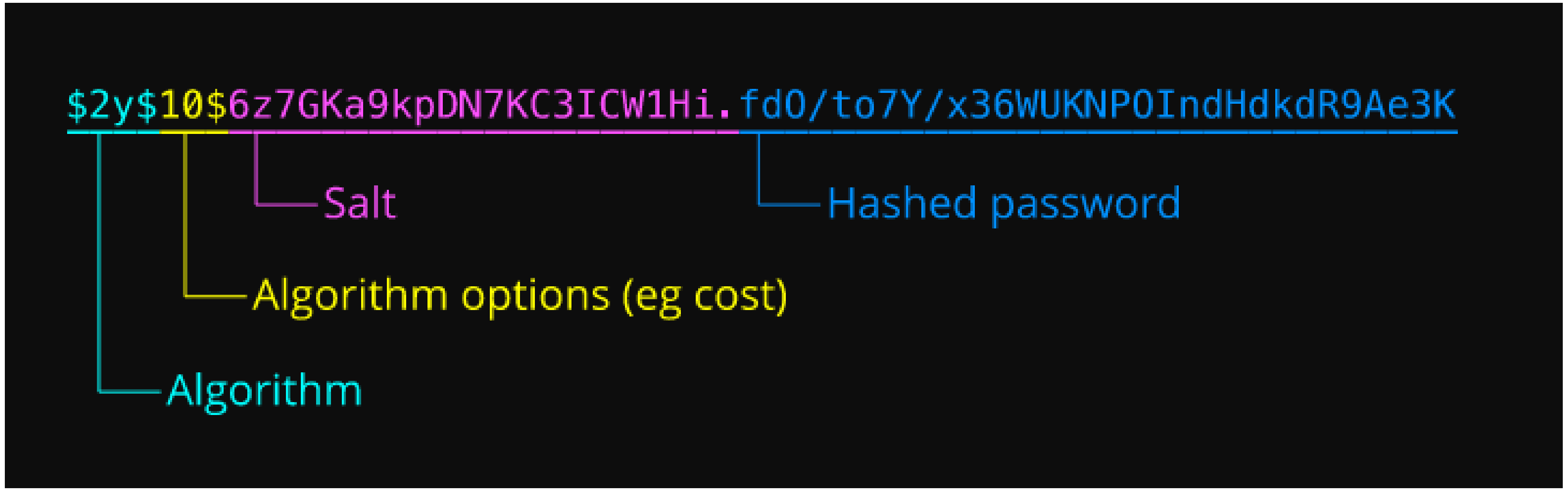


Password Hashing Functions, Source: https://synkre.com/how-secure-is-bcrypt/

bcrypt

- bcrypts goal is to create a cryptographic hash function that can be tuned to run slower in newly available hardware; that is, the function scales with computing power to mitigate this attack vector that mentioned above.

- bcrypt use Blowfish to add extra computational demand to hash functions to protect against dictionary and brute force attacks by slowing down the attack.

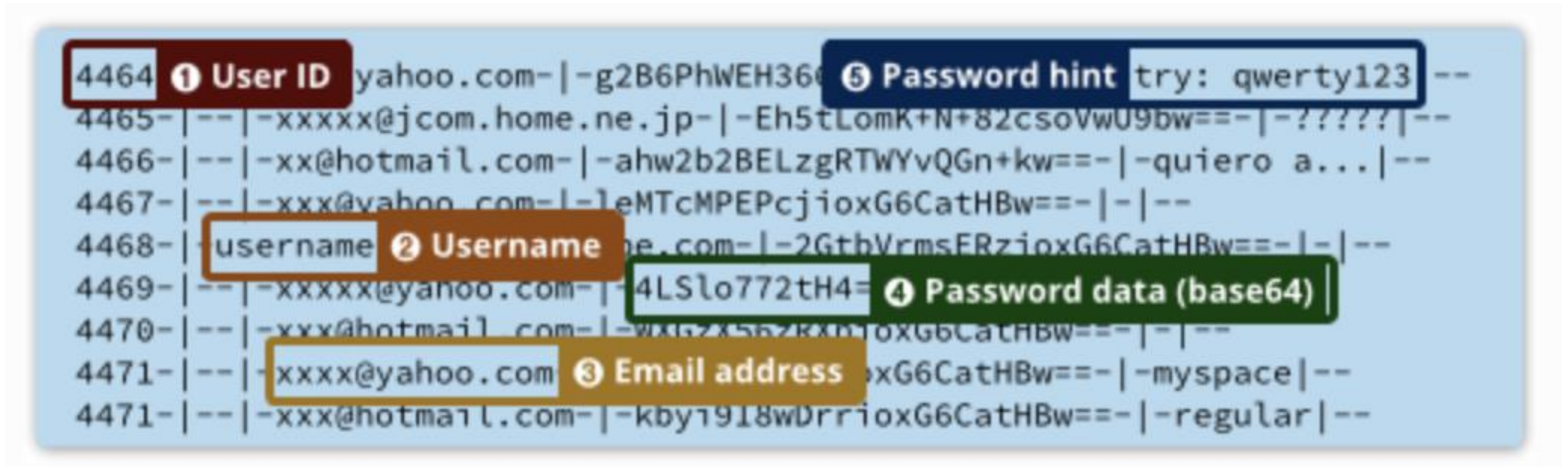

$2y$10$6z7GKa9kpDN7KC3ICW1Hi.fd0/to7Y/x36WUKNPOIndHdkdR9Ae3K
Salt
Hashed password
Algorithm options (eg cost)
Algorithm

bcrypt, Source: https://commons.wikimedia.org/wiki/File:Bcrypt.png

Back to Adobe Case

- Adobe passwords were encrypted, not hashed.
- In this case it means anyone who computes, guesses or acquires Adobes decryption key immediately gets access to *all* the passwords in the database.
- In the end, with very little effort, someone can recover an awful lot of information about the breached passwords, including: identifying the top five passwords precisely, plus the 2.75% of users who chose them; and determining the exact password length of nearly one third of the database.
- Basically using salted hashes on the other hand wouldn’t have yielded up any such information.

Breaking the Adobe's Encryption



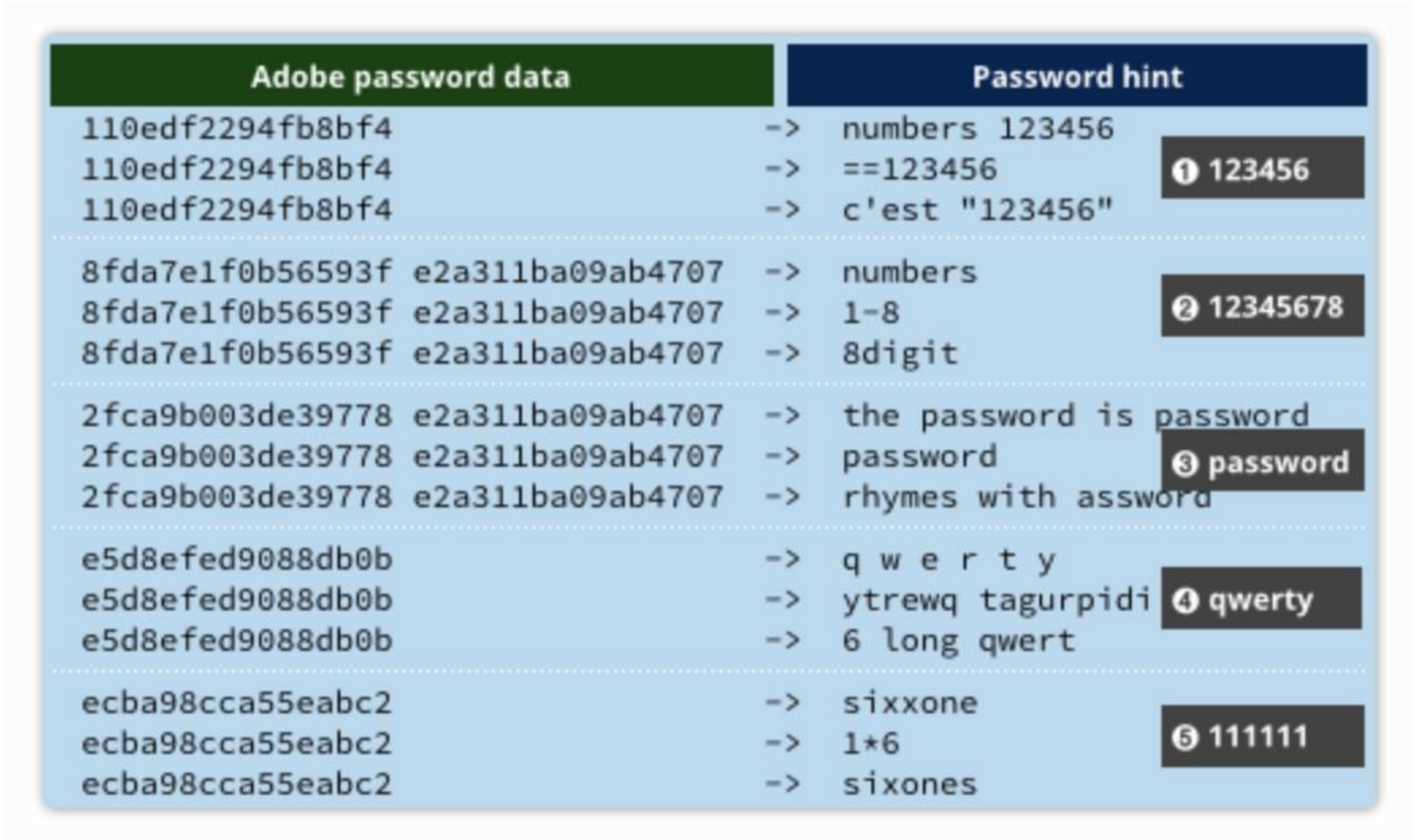
```
4464 yahoo.com-|-g2B6PhWEH36 try: qwerty123 --
4465-|--|-xxxxx@jcom.home.ne.jp-|-Eh5tLomK+N+82csoVwU9bw==-|-?????|--
4466-|--|-xx@hotmail.com-|-ahw2b2BELzgRTWYvQGn+kw==-|-quiero a...|--
4467-|--|-xxx@yahoo.com-|-leMTcMPEPcjioxG6CatHBw==-|-|--
4468-| username e.com-|-2GtbVrmsERzioxG6CatHBw==-|-|--
4469-|--|-xxxxx@yahoo.com-|-4LSlo772tH4=
4470-|--|-xxx@hotmail.com-|-WXGzX5bzRXbjoxG6CatHBw==-|-|--
4471-|--| xxxx@yahoo.com xG6CatHBw==-|-myspace|--
4471-|--|-xxx@hotmail.com-|-kbyi918wDrrioxG6CatHBw==-|-regular|--
```

Data Dump, Source: https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/

- From the dumped data we can get the encrypted passwords and password hints.
- With all data lengths a multiple of eight, we’re almost certainly looking at a block cipher that works eight bytes (64 bits) at a time.
- That, in turn, suggests that we’re looking at DES, or its more resilient modern derivative, Triple DES, usually abbreviated to 3DES.

Breaking the Adobe's Encryption

- Finally the last step is to create a crib sheet to recover passwords. This can be done by assuming that the most common passwords in the past leaks will remain as most common.



Adobe password data		Password hint	
110edf2294fb8bf4	->	numbers 123456	❶ 123456
110edf2294fb8bf4	->	==123456	
110edf2294fb8bf4	->	c'est "123456"	
8fda7e1f0b56593f e2a311ba09ab4707	->	numbers	❷ 12345678
8fda7e1f0b56593f e2a311ba09ab4707	->	1-8	
8fda7e1f0b56593f e2a311ba09ab4707	->	8digit	
2fca9b003de39778 e2a311ba09ab4707	->	the password is password	❸ password
2fca9b003de39778 e2a311ba09ab4707	->	password	
2fca9b003de39778 e2a311ba09ab4707	->	rhymes with assword	
e5d8efed9088db0b	->	q w e r t y	❹ qwerty
e5d8efed9088db0b	->	ytrewq tagurpidi	
e5d8efed9088db0b	->	6 long qwert	
ecba98cca55eabc2	->	sixxone	❺ 111111
ecba98cca55eabc2	->	1*6	
ecba98cca55eabc2	->	sixones	

- Without any considerable work, we already have a lot of clues to recover the passwords.

- For example, assuming e2a311ba09ab4707 is 8 zero-bytes, we can say that 27% of all passwords are eight characters long.

Crib Sheet, Source: https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/

Taming Bots

Bots

- Bots are computer programs trained to behave like humans.

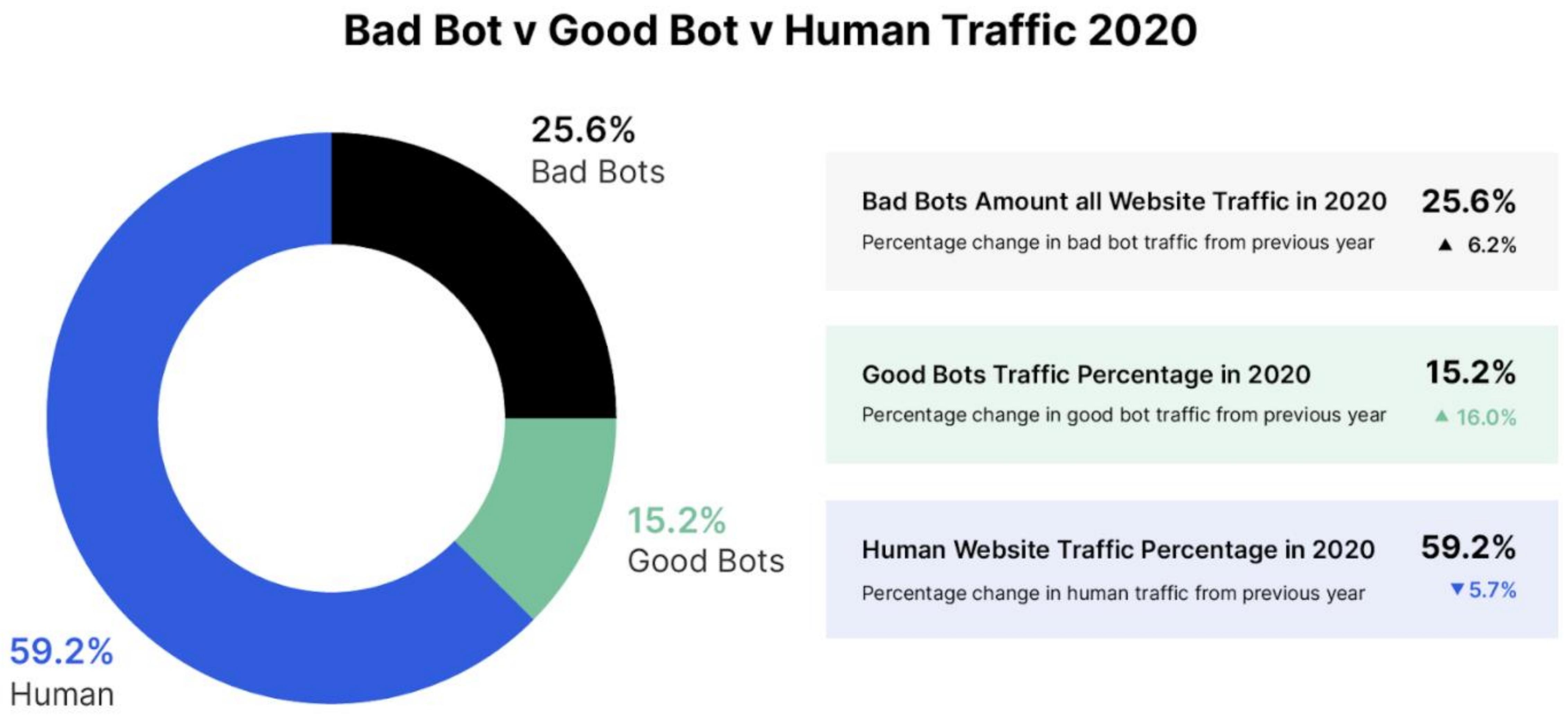
- Help us to automated lot of tasks, save time and money.

- As bots become even more human-like every year, it has made it more difficult to accurately **detect bots crawling your website**.

- Bot traffic even makes up over **40%** of website visits

- Effects of bad bots:
  - Scrape your content, scrape links or worse still - your data
  - Attempt to disrupt your site performance
  - Post spam content and spurious form generation

Bots



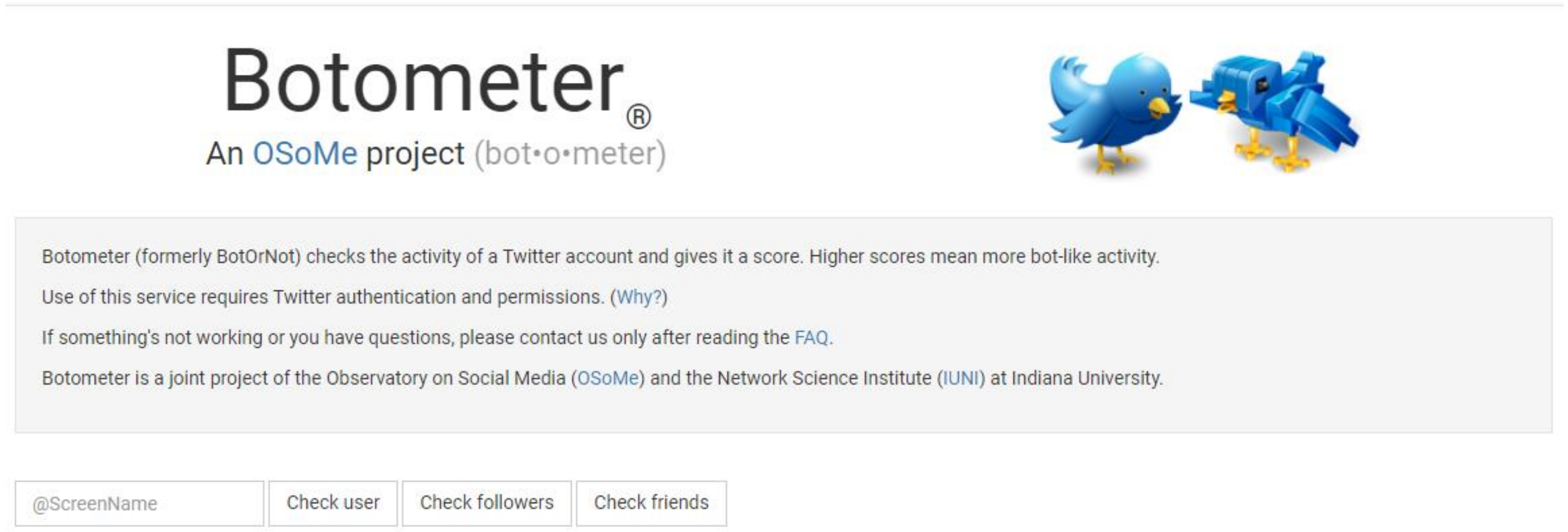
Bad Bots Amount all Website Traffic in 2020	**25.6%**
Percentage change in bad bot traffic from previous year	▲ 6.2%
Good Bots Traffic Percentage in 2020	**15.2%**
Percentage change in good bot traffic from previous year	▲ 16.0%
Human Website Traffic Percentage in 2020	**59.2%**
Percentage change in human traffic from previous year	▼ 5.7%

Bad Bot v Good Bot v Human Traffic 2020, Source: https://www.imperva.com/blog/bad-bot-report-2021-the-pandemic-of-the-internet/

Bots

- Bots are 5-20% of all Twitter users

Img source



Botometer®

An OSoMe project (bot•o•meter)

Botometer (formerly BotOrNot) checks the activity of a Twitter account and gives it a score. Higher scores mean more bot-like activity.

Use of this service requires Twitter authentication and permissions. (Why?)

If something's not working or you have questions, please contact us only after reading the FAQ.

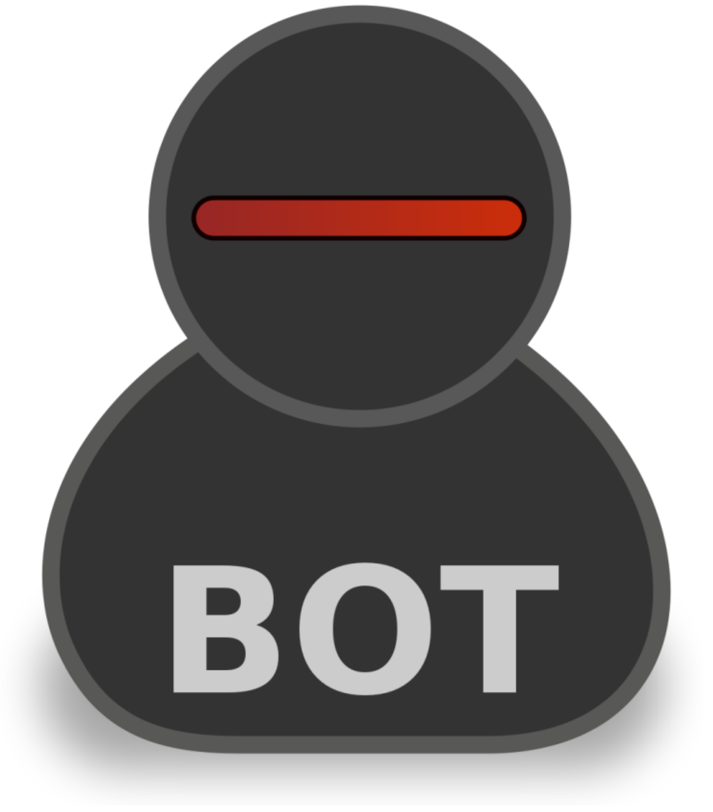
Botometer is a joint project of the Observatory on Social Media (OSoMe) and the Network Science Institute (IUNI) at Indiana University.

https://botometer.osome.iu.edu/

Bot Detection

Some Indicators of Existence of Bots:

- Direct traffic sources
- Reducing server performance or website speed
- Faster browsing rate
- Inconsistent page views
- Increasing bounce rate
- Junk user information
- Content scraping and stealing
- Spike in traffic from an unexpected location
- Passive/active fingerprinting



Bot, Source:
https://pl.wikipedia.org/wiki/Bot_(program)

Bot Detection

- **Direct traffic sources:**

When there’s a “bot attack” on your site, or to a particular page, the only channel contributing to the traffic will be “direct” traffic.

- **Reducing server performance or website speed:**

A slowdown in your hosting server performance or speed mainly comes from bots.

- **Faster browsing rate:**

Machines can browse a website much faster than humans. So when you experience a huge amount of traffic in a short period, it’s mostly because of bots.

Bot Detection

- **Inconsistent page views:**

When bots enter a website, they tend to load up a large number of pages all at once. Therefore bots can lead to an unusual increase in page views.

- **Increasing bounce rate:**

If the average page duration for a website (time spent on-page) reduces, and the bounce rate increases (visitors not viewing other pages or interacting with the page), this is a clear indicator that your website is being visited by rogue bots.

- **Junk user information:**

weird, unusual account creation or sign-ups with weird email addresses, accompanied by potential fake names and phone numbers is a huge indicator of bots on your website.

Bot Detection

- **Spike in traffic from an unexpected location:**

A sudden increase in users from a particular region, country, city or other location, which you may not be familiar with, or not related to your website can indicate existence of bots.

- **Passive/active fingerprinting:**

Web browsers are generally complex and packed full of information (headers, userAgent, IP, extensions, etc.) Your system will send a request to every browser, querying this information on the browser as an *identifier*. Utilizing a solution like **device fingerprinting** can access deep information about a user to easily identify bots.

Honeypots

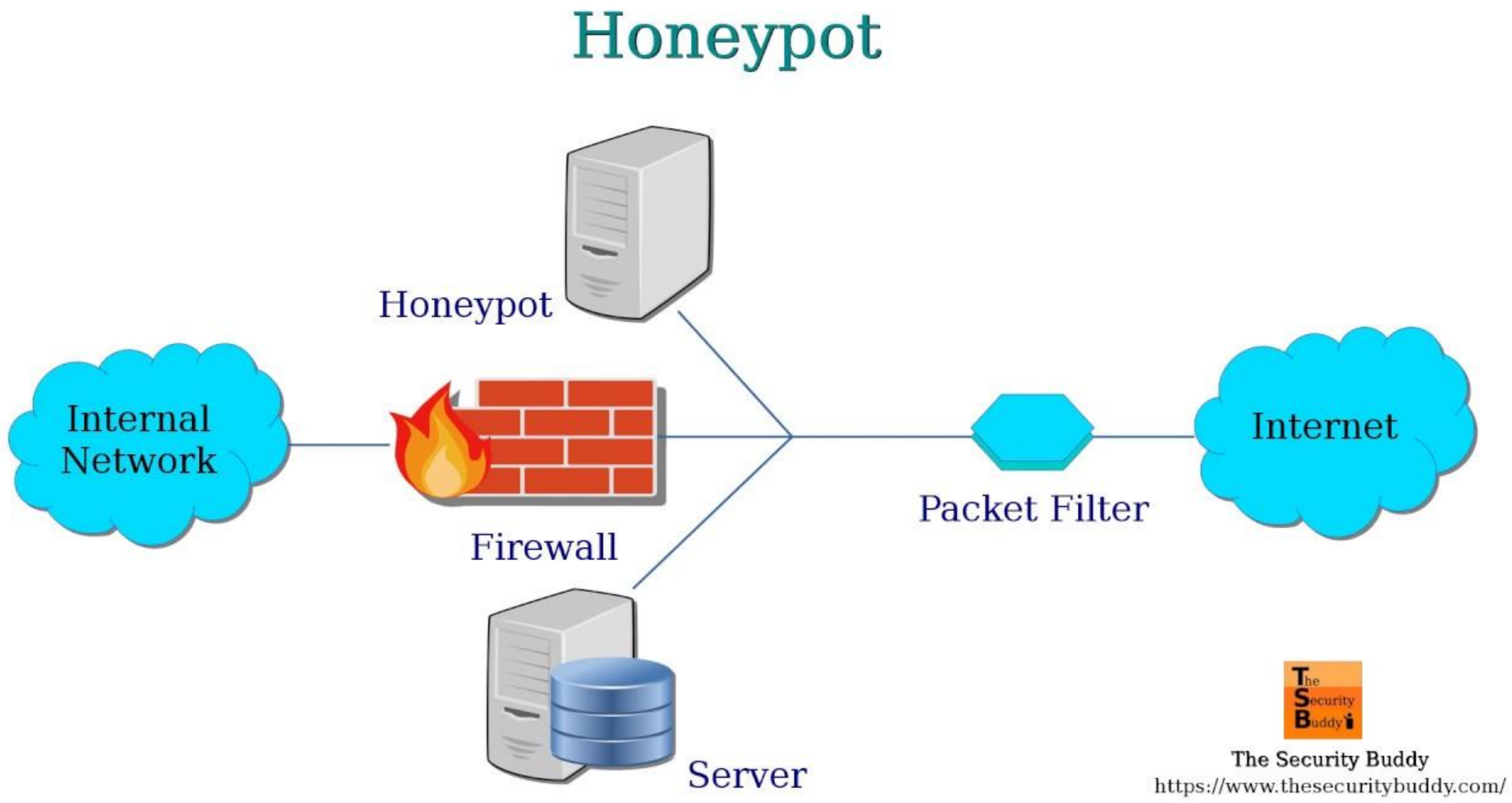
- A **honeypot** is a decoy created to look like a compromised system that will seem like an easy target for malefactors.
- Honeypots distract hackers and thus protect the real possible targets.
- Honeypots are information tools that can help you understand existing threats to your website and spot the emergence of new threats.

What Is A Honeypot?, Source: https://infatica.io/blog/honeypots-overview/

How Honeypots Work ?

- Honeypots fake **legitimate targets** such as a company's customer billing system.
- Their deliberate security vulnerabilities lure the hackers.
- Once the hackers get in the honeypot, they can be tracked, and their behavior assessed for clues on how to make the real network more secure.
- By tracking the hackers you can learn:
  - where the cybercriminals are coming from
  - the level of threat
  - what modus operandi they are using
  - what data or applications they are interested in
  - how well your security measures are working to stop cyberattacks

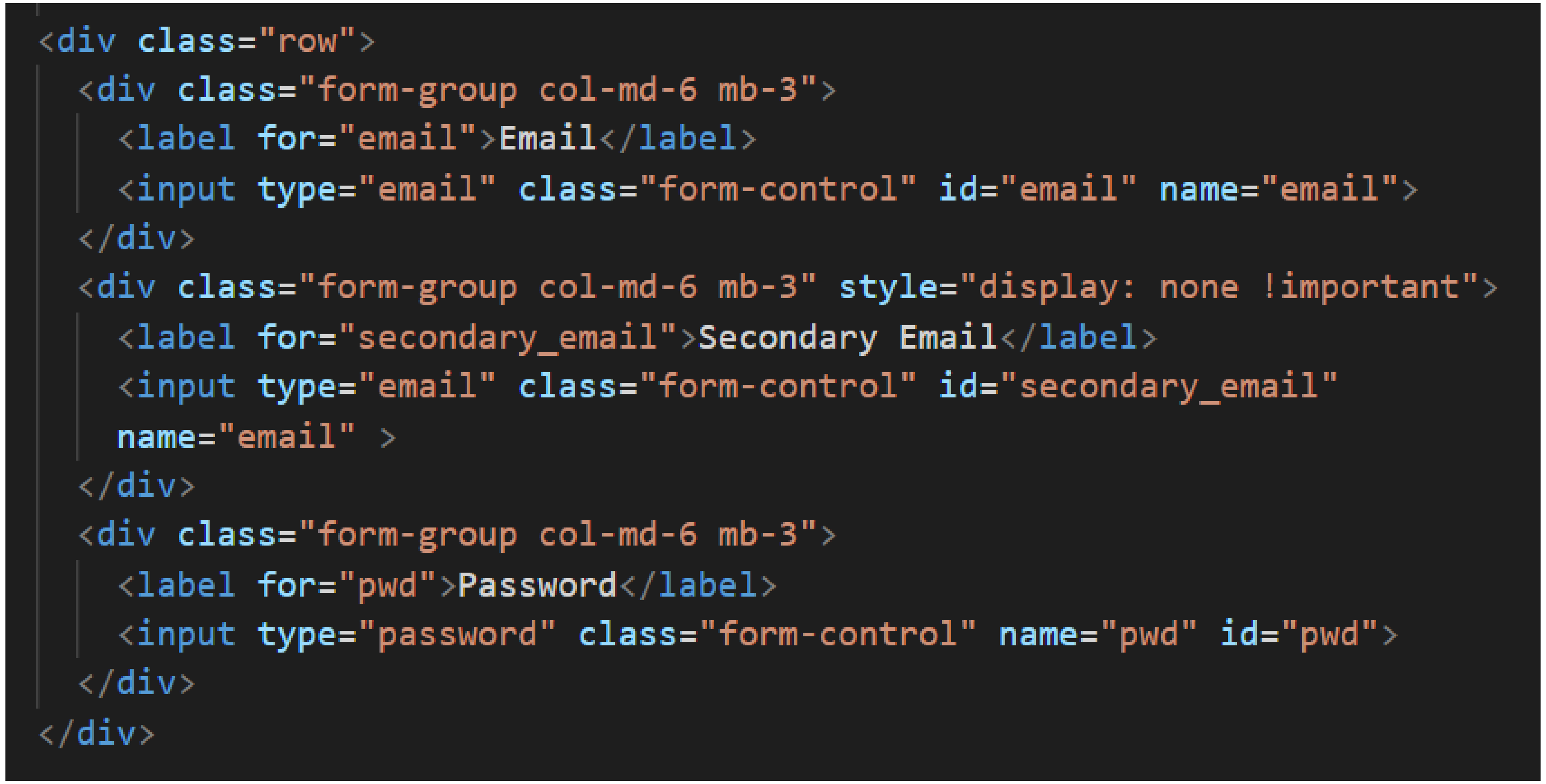
How Honeypots Work ?



Honeypot, Source: https://www.thesecuritybuddy.com/data-breaches-prevention/what-is-honeypot/

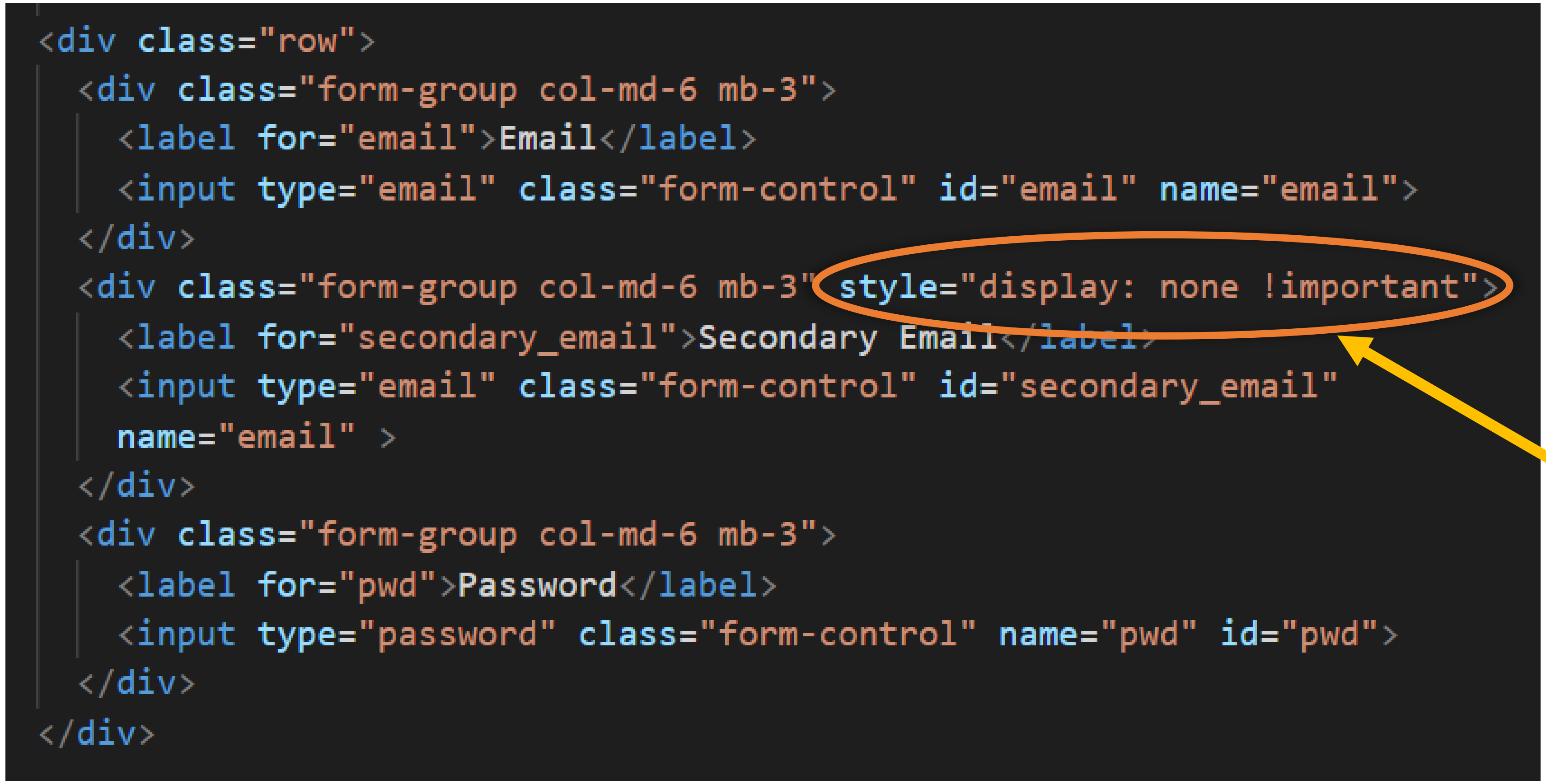
Honeypots can also be added to Forms

Honeypots can also be added to Forms



```
<div class="row">
  <div class="form-group col-md-6 mb-3">
    <label for="email">Email</label>
    <input type="email" class="form-control" id="email" name="email">
  </div>
  <div class="form-group col-md-6 mb-3" style="display: none !important">
    <label for="secondary_email">Secondary Email</label>
    <input type="email" class="form-control" id="secondary_email"
    name="email" >
  </div>
  <div class="form-group col-md-6 mb-3">
    <label for="pwd">Password</label>
    <input type="password" class="form-control" name="pwd" id="pwd">
  </div>
</div>
```

Honeypots can also be added to Forms

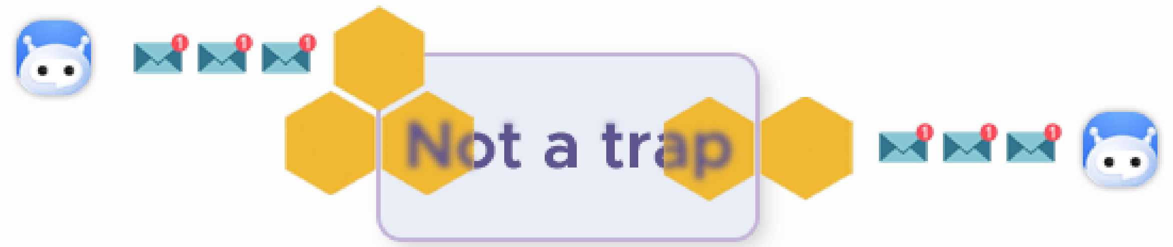


```
<div class="row">
  <div class="form-group col-md-6 mb-3">
    <label for="email">Email</label>
    <input type="email" class="form-control" id="email" name="email">
  </div>
  <div class="form-group col-md-6 mb-3" style="display: none !important">
    <label for="secondary_email">Secondary Email</label>
    <input type="email" class="form-control" id="secondary_email"
    name="email" >
  </div>
  <div class="form-group col-md-6 mb-3">
    <label for="pwd">Password</label>
    <input type="password" class="form-control" name="pwd" id="pwd">
  </div>
</div>
```

An unsophisticated bot will fill-in the form even if it is not displayed.

Usage of Honeypots

- Spam Detection
- Protection From SQL Injections
- Protection From Malware
- Searching For Malicious Servers
- Anti-Crawler



Spam Detection, Source:
https://infatica.io/blog/honeypots-overview/

Benefits and Dangers

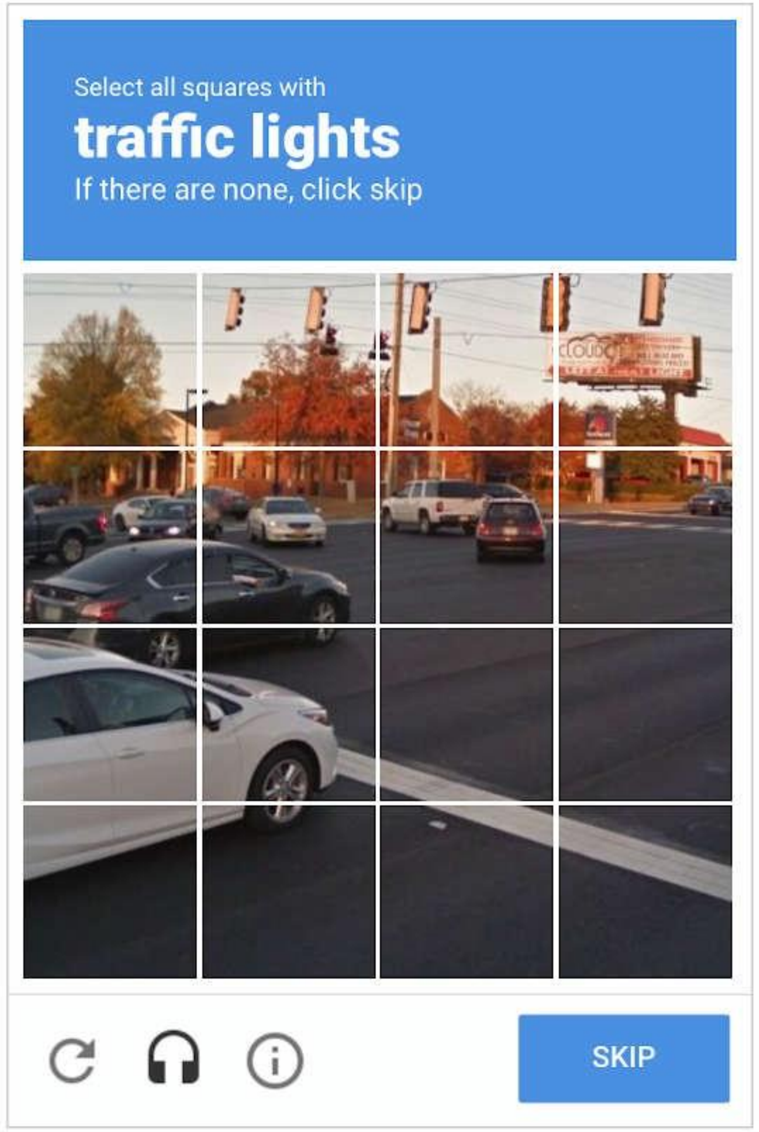
- There are several benefits of honeypots:
  - They make it much easier to spot patterns, (e.g., similar IP addresses)
  - They are resource light
  - Have a low false positive rate
  - They can give you reliable intelligence
  - Great training tools for technical security staff

- However:
  - Honeypots cannot catch everything
  - When identified, hackers can feed bad information to the honeypot
  - They can be used as a way to your real system
  - Honeypots in forms may have faulty renderings in non-standard browsers and browsers for visually impaired people

Captchas

"Completely Automated Public Turing test to tell Computers and Humans Apart" or a **CAPTCHA** is a type of challenge-response test used in computing to determine whether the user is human.

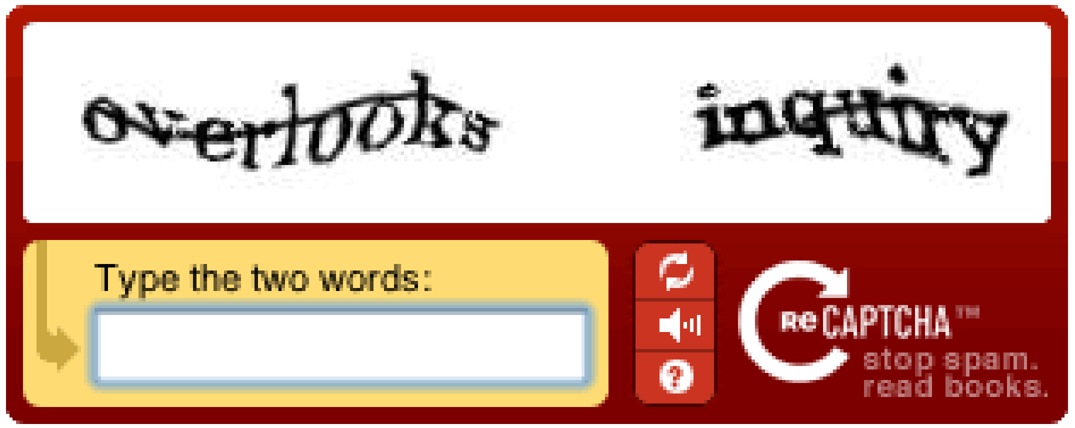
CAPTCHAs protect websites against bots by generating and grading tests that humans can pass but current computer programs cannot.


Select all squares with
traffic lights
If there are none, click skip

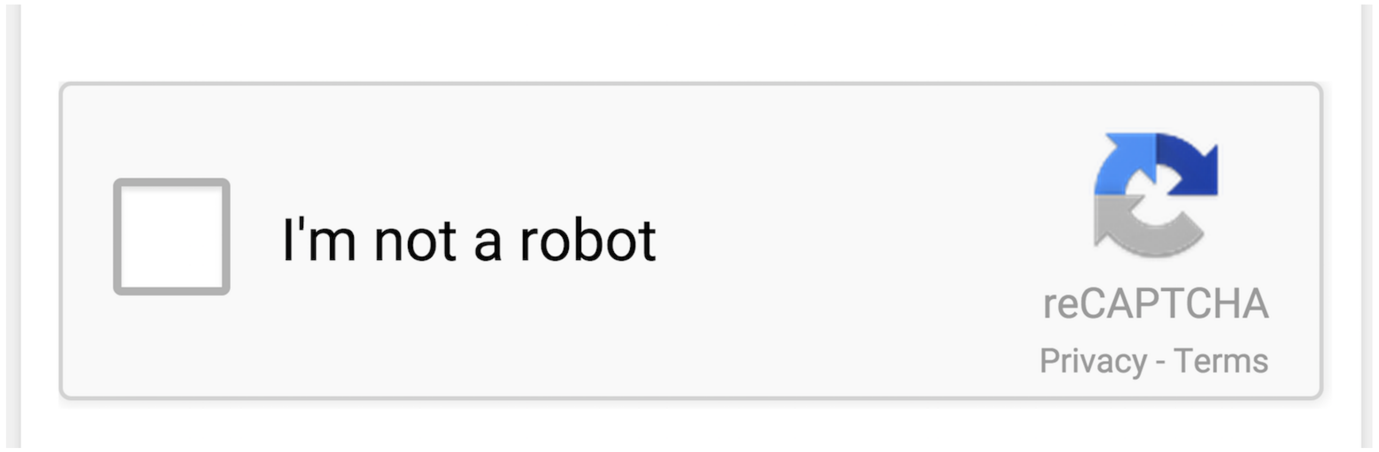
Traffic Lights Captcha, Source: https://onezero.medium.com/why-captcha-pictures-are-so-unbearably-depressing-20679b8cf84a

Types of Captchas

- Text-based CAPTCHAs
- CAPTCHA Image
- Audio CAPTCHA
- Math or Word Problems
- Social Media Sign In
- No CAPTCHA ReCAPTCHA



reCAPTCHA, Source: http://www.captcha.net/



No CAPTCHA, Source: https://de.wordpress.org/plugins/login-recaptcha/

Drawbacks of Captchas

Captchas helps to prevent bots and spams but :

- Can be disruptive and frustrating for users
- May be difficult to understand or use for some audiences
- Some CAPTCHA types do not support all browsers
- Some CAPTCHA types are not accessible to users who view a website using screen readers or assistive devices

hCaptcha

Compared to traditional reCAPTCHA:

- hCaptcha gives you more control over the difficulty level you need for your site and does a better job of protecting your users' privacy.

- hCaptcha has 100% of the features of reCAPTCHA.

- hCaptcha makes compliance with privacy rules like GDPR, LGPD, CCPA more straightforward.

- hCaptcha also pays publishers (free accounts) for the work your users are doing. Using reCAPTCHA donates that work to Google.

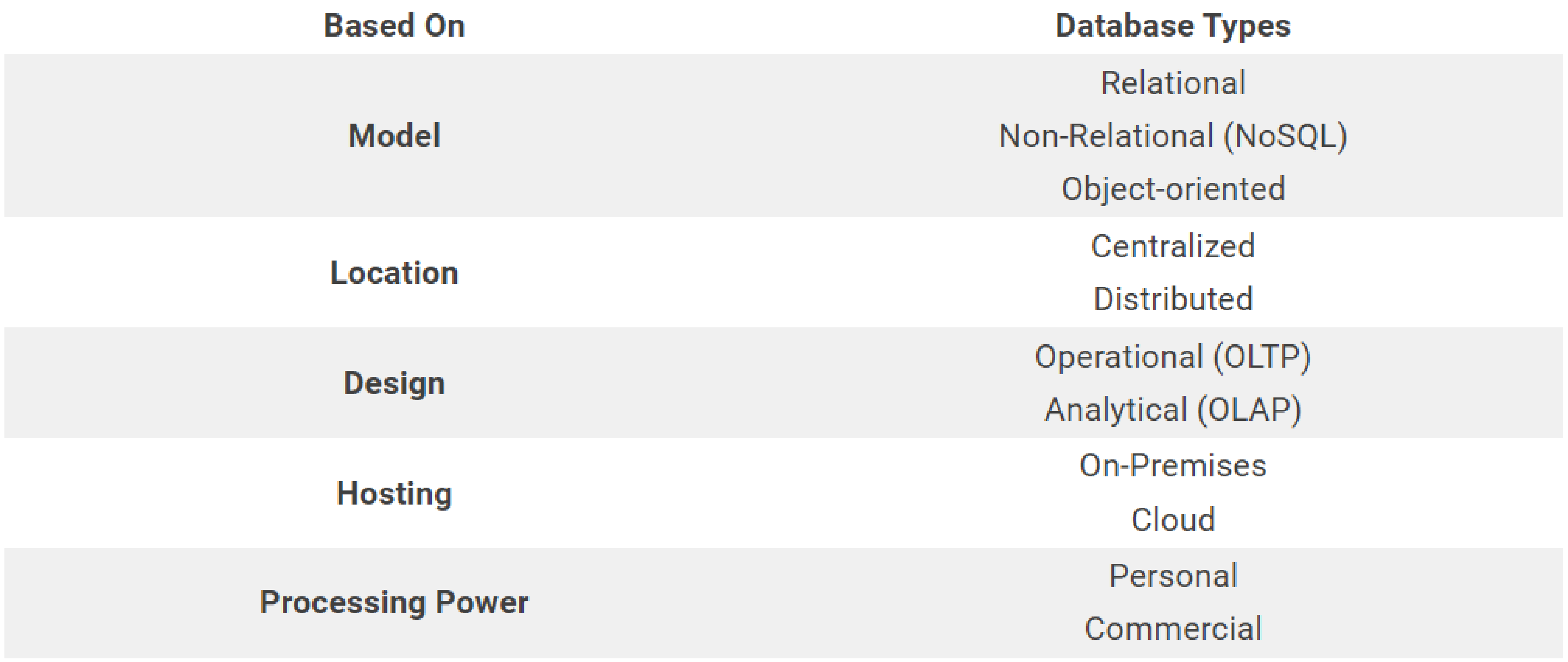
How it works ?

- The user answers an hCaptcha.
- They get a passcode from our server that is embedded in your form.
- When the user clicks Submit the passcode is sent to your server in the form.
- Your server then checks that passcode with the hCaptcha server API.
- hCaptcha says it is valid and credits your account. Your server now knows the user is not a bot and lets them log in.

Module 10: MongoDB

What is a Database?

- Databases are **organized collections** of related data records
- There are different ways of storing and organizing data in a database

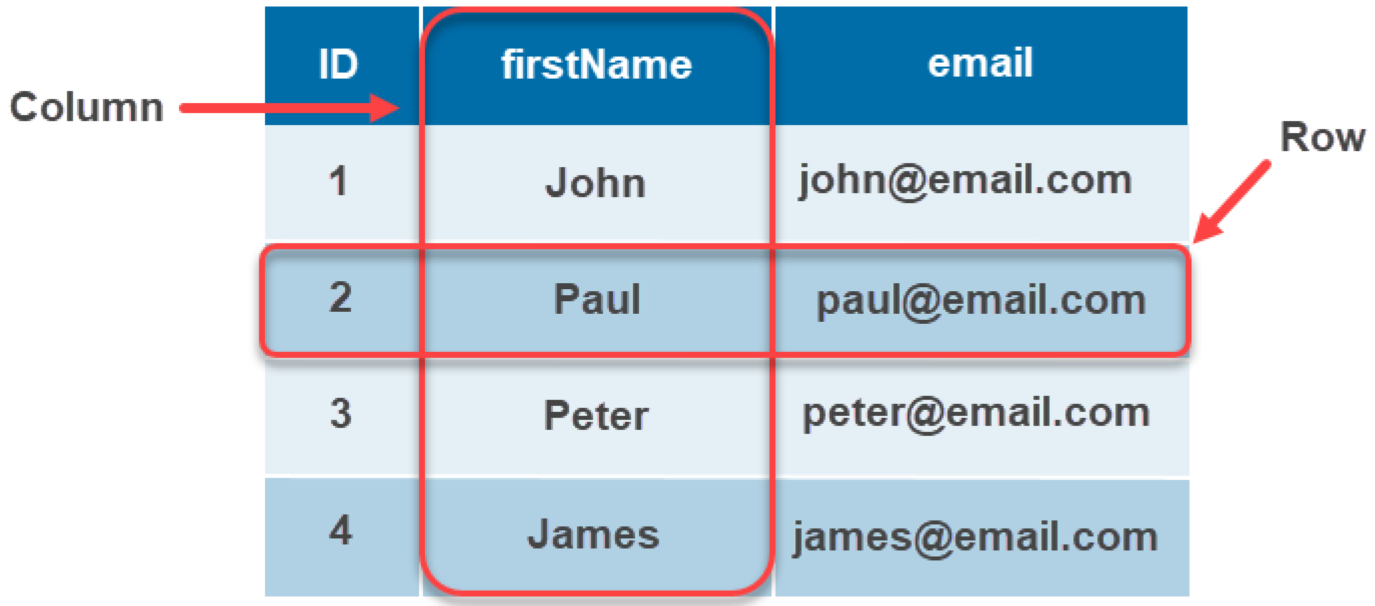


Based On	Database Types
Model	Relational Non-Relational (NoSQL) Object-oriented
Location	Centralized Distributed
Design	Operational (OLTP) Analytical (OLAP)
Hosting	On-Premises Cloud
Processing Power	Personal Commercial

https://phoenixnap.com/kb/database-types

Relational vs NoSQL

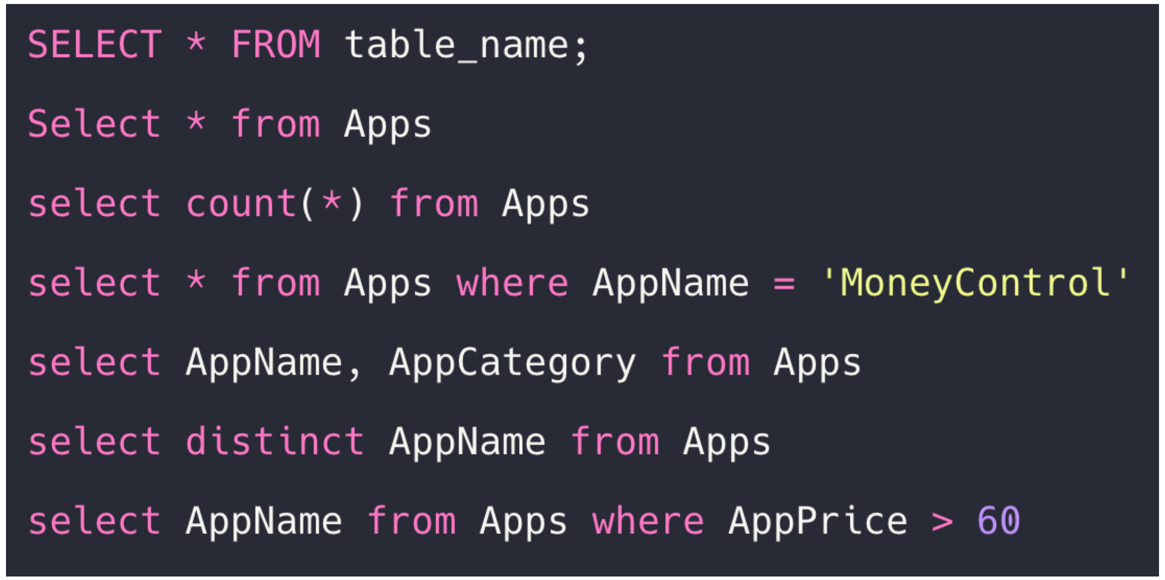
RELATIONAL (SQL)



ID	firstName	email
1	John	john@email.com
2	Paul	paul@email.com
3	Peter	peter@email.com
4	James	james@email.com

Img source

- Data is structured in **tables**
- Tables are linked via **foreign keys** (IDs)
- Keys allows for **fast** indexing and search
- Reliable and stable projects (e.g., MySQL)
- Formal language to query database

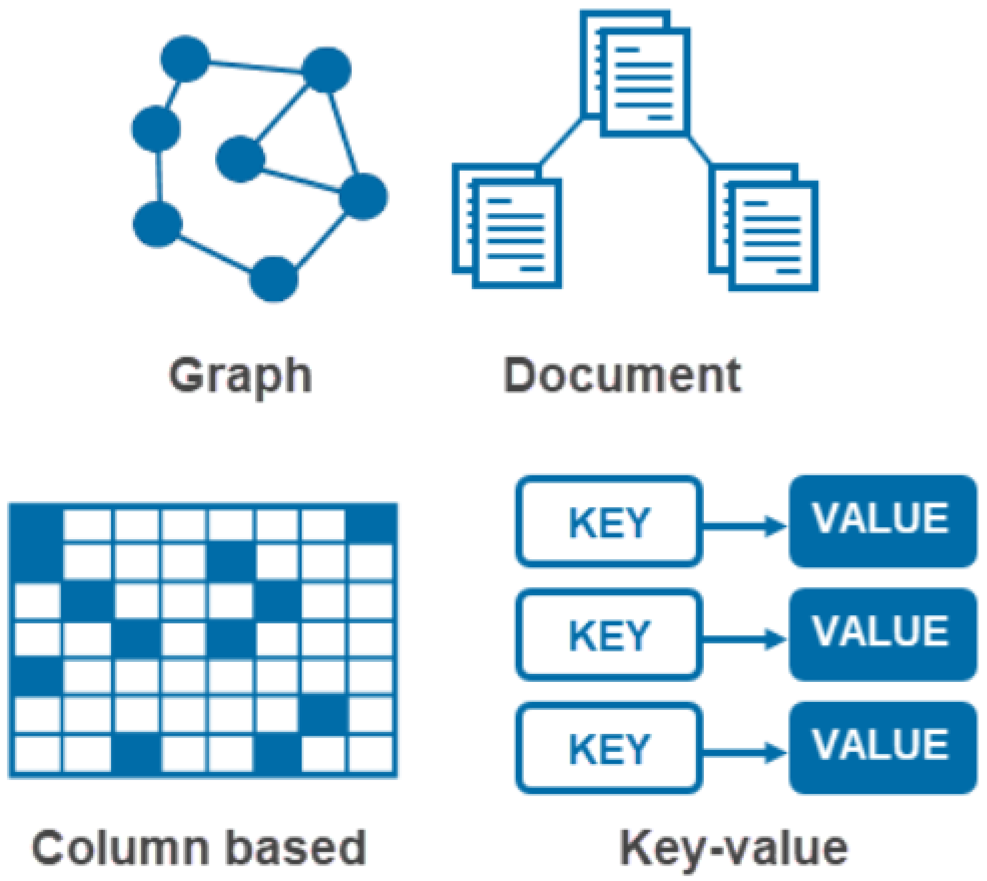


```
SELECT * FROM table_name;

Select * from Apps

select count(*) from Apps

select * from Apps where AppName = 'MoneyControl'

select AppName, AppCategory from Apps

select distinct AppName from Apps

select AppName from Apps where AppPrice > 60
```

Img source

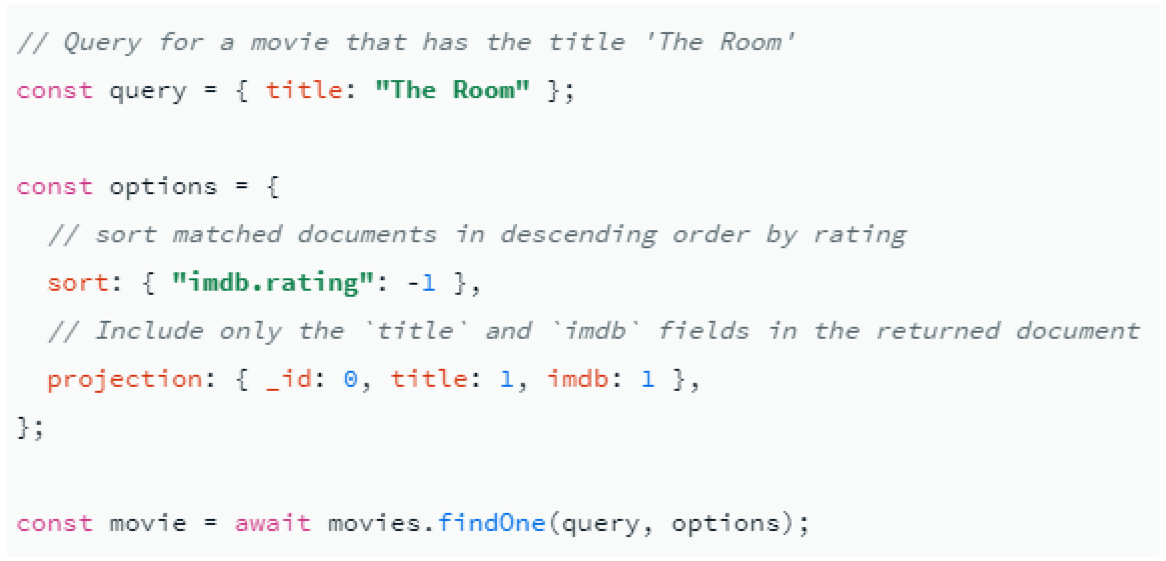
Relational vs NoSQL

NoSQL



Img source

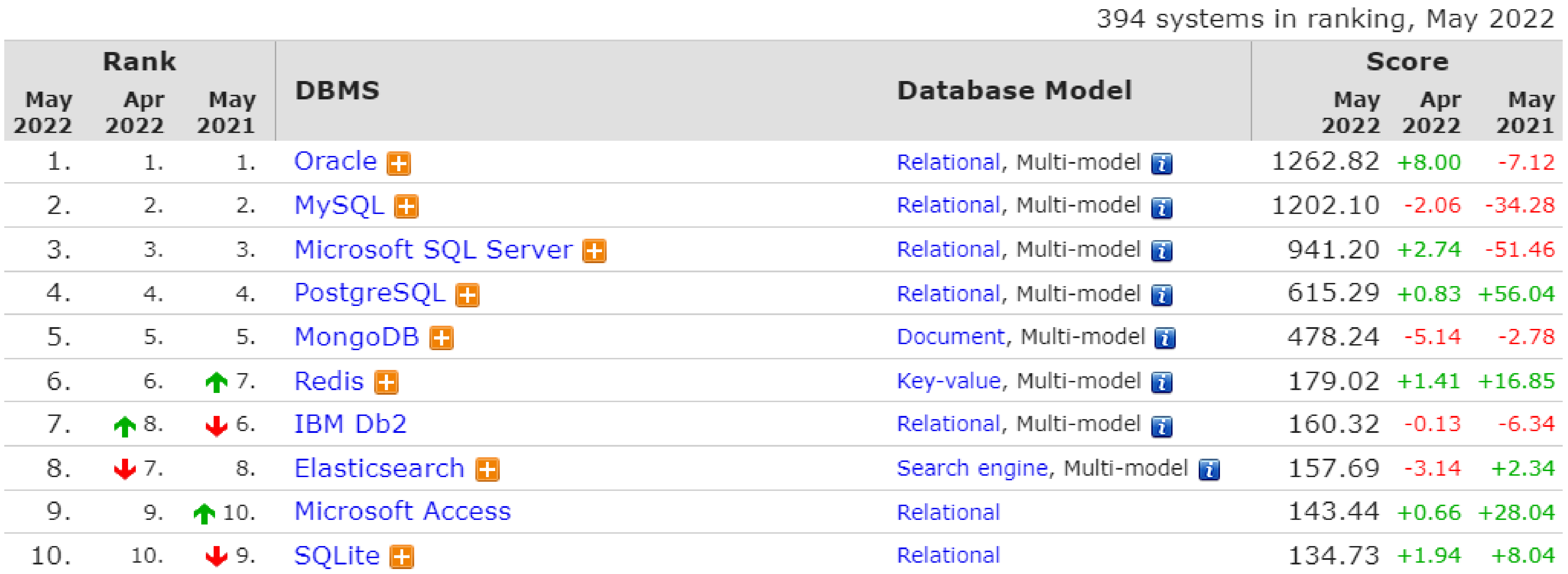
- Data is organized in various ways
- They serve specific purpose of processing the data
- **Document** databases are the most common (**MongoDB**)
- Very flexible
- Scale very well with data



```
// Query for a movie that has the title 'The Room'
const query = { title: "The Room" };

const options = {
  // sort matched documents in descending order by rating
  sort: { "imdb.rating": -1 },
  // Include only the `title` and `imdb` fields in the returned document
  projection: { _id: 0, title: 1, imdb: 1 },
};

const movie = await movies.findOne(query, options);
```

https://www.mongodb.com/docs/drivers/node/v4.6/usage-examples/findOne/

Rankings



394 systems in ranking, May 2022

Rank			DBMS	Database Model	Score		
May 2022	Apr 2022	May 2021			May 2022	Apr 2022	May 2021
1.	1.	1.	Oracle	Relational, Multi-model	1262.82	+8.00	-7.12
2.	2.	2.	MySQL	Relational, Multi-model	1202.10	-2.06	-34.28
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	941.20	+2.74	-51.46
4.	4.	4.	PostgreSQL	Relational, Multi-model	615.29	+0.83	+56.04
5.	5.	5.	MongoDB	Document, Multi-model	478.24	-5.14	-2.78
6.	6.	↑ 7.	Redis	Key-value, Multi-model	179.02	+1.41	+16.85
7.	↑ 8.	↓ 6.	IBM Db2	Relational, Multi-model	160.32	-0.13	-6.34
8.	↓ 7.	8.	Elasticsearch	Search engine, Multi-model	157.69	-3.14	+2.34
9.	9.	↑ 10.	Microsoft Access	Relational	143.44	+0.66	+28.04
10.	10.	↓ 9.	SQLite	Relational	134.73	+1.94	+8.04

https://db-engines.com/en/ranking

MongoDB Documentation

- Free, open source
- Can be installed locally or used on in the cloud (Atlas)
- Great documentation and a large community
- Why using MongoDB? https://www.mongodb.com/why-use-mongodb
- 4 Reasons to use MongoDB: https://www.mongodb.com/developer/article/top-4-reasons-to-use-mongodb/

MongoDB Documentation

- You can access MongoDB via a MongoDB shell, e.g., **mongosh**

- In mongosh, you type command as in a terminal

- Documentation for shell access is here:
https://www.mongodb.com/docs/manual/

MongoDB Documentation

- You can access MongoDB via a MongoDB shell, e.g., **mongosh**
- In mongosh, you type command as in a terminal
- Documentation for shell access is here: https://www.mongodb.com/docs/manual/

- In your app, you need to use **MongoDB driver**, e.g., the driver for Node.JS
- The commands are slightly different from the shell, the most important difference is you need to account for **asynchronous interaction**
- Documentation for Node.JS is here: https://www.mongodb.com/docs/drivers/node/current/quick-start/