



POLITECNICO
MILANO 1863



Welcome!

Web Security: HTTP and Injections

Who am I?

- Juri Sacchetta
- PhD candidate, NECSTLab — Politecnico di Milano
- juri.sacchetta@polimi.it



The range

A fictional retail bank, deliberately broken, running for this session only

Platform bocconi.necst.it — register, pick a challenge, read the brief

Systems each challenge links to its own host under meridian.necst.it

Scoring every challenge hides a string shaped like `flag{...}`; paste it into the platform

Tools your browser is enough. Right-click, then Inspect

Rules attack only the hosts the platform links to. NeoBank is invented: no real institution, customer or account appears anywhere in here



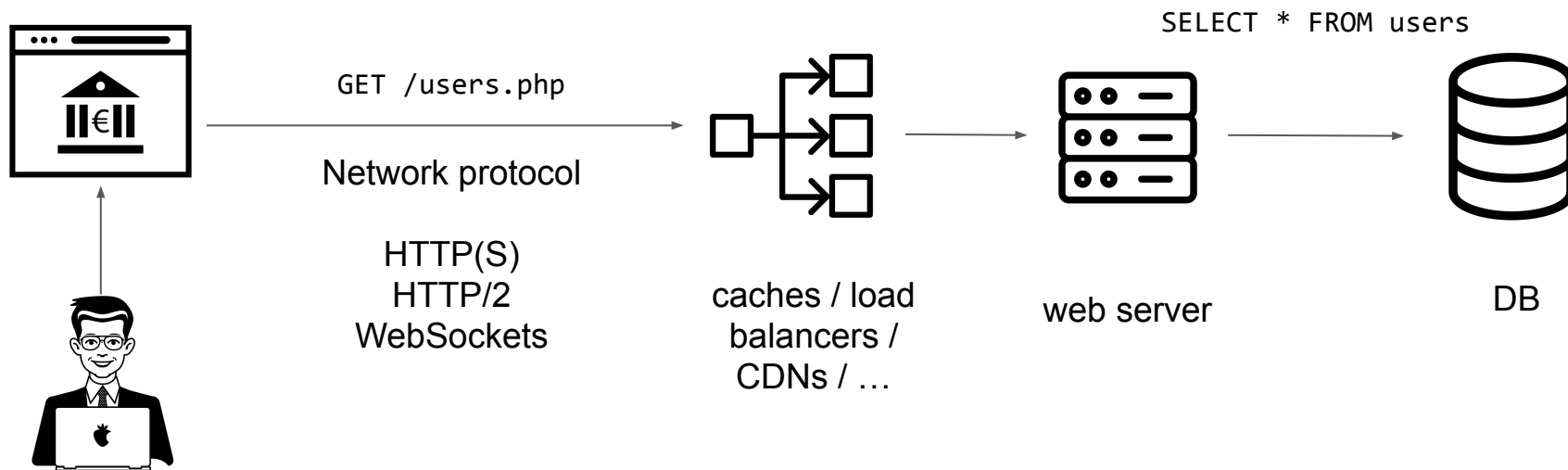
POLITECNICO
MILANO 1863



Web 101

How to Load a Webpage!

`https://bank.com/users.php`



Resources are identified by a URL

What's in a URL (Uniform Resource Locator)?

`http://user:pass@www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#Somewhere`

scheme	auth	host	port	path	parameters (query string)	anchor (fragment) client-side!
--------	------	------	------	------	------------------------------	--------------------------------------

1: <https://tools.ietf.org/html/rfc1738>

2: <https://tools.ietf.org/html/rfc3986>

https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL

How is a Website Made?

The main resources returned by the server following a request

When we make a request, the server responds by returning resources. On average, the structure of a website consists of **three fundamental macro-categories**:

HTML

Structure and Content

Represents the skeleton of the page. It defines the organization of text, links, images, and all structural elements that make up the document.

CSS

Style and Presentation

Manages the visual appearance. It determines colors, fonts, layouts, spacing, and ensures the page adapts correctly to any screen size (responsive).

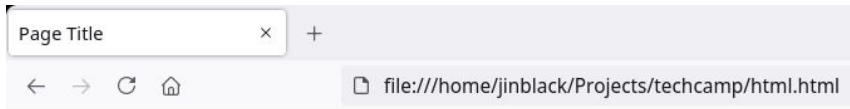
JavaScript

Logic and Interactivity

Makes the page dynamic. It handles complex behaviors, user interaction, animations, asynchronous data updates, and responses to events.

Drawing a WebPage! (HTML)

```
1.  <html>
2.    <head>
3.      <title>Page Title</title>
4.    </head>
5.    <body>
6.      <h1>This is an HTML Page!</h1>
7.      <p>This is a paragraph</p>
8.      <a href="https://www.w3schools.com/html/">
9.        more about html
10.     </a>
11.    </body>
12. </html>
```



This is an HTML Page!

This is a paragraph

[more about html](https://www.w3schools.com/html/)

Drawing a Pretty WebPage! (CSS)

```
1. <html>
2.   <head>
3.     <title>Page Title</title>
4.     <link rel="stylesheet" href="css.css">
5.   </head>
6.   <body>
7.     <h1>This is an HTML Page!</h1>
8.     <p>This is a paragraph</p>
9.     <a href="https://www.w3schools.com/css/">
10.      more about css
11.    </a>
12.  </body>
13. </html>
```

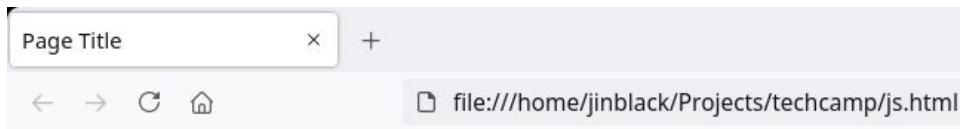
```
1. body {
2.   background-color: lightblue;
3. }
4.
5. h1 {
6.   color: white;
7.   text-align: center;
8. }
9.
10. p {
11.   font-family: verdana;
12.   font-size: 20px;
13. }
```



Dynamic WebPage! (JS)

```
1. <html>
2.   <head>
3.     <title>Page Title</title>
4.     <script src="js.js"></script>
5.   </head>
6.   <body>
7.     <h1>This is an HTML Page!</h1>
8.     <p id="demo">This is a paragraph</p>
9.     <button type="button" onclick="changeColor()">This is a Button</button>
10.    <br>
11.    <a href="https://www.w3schools.com/js/">
12.      more about js
13.    </a>
14.  </body>
15. </html>
```

```
1. function changeColor(){
2.   document.getElementById('demo').style.color='red';
3. }
```



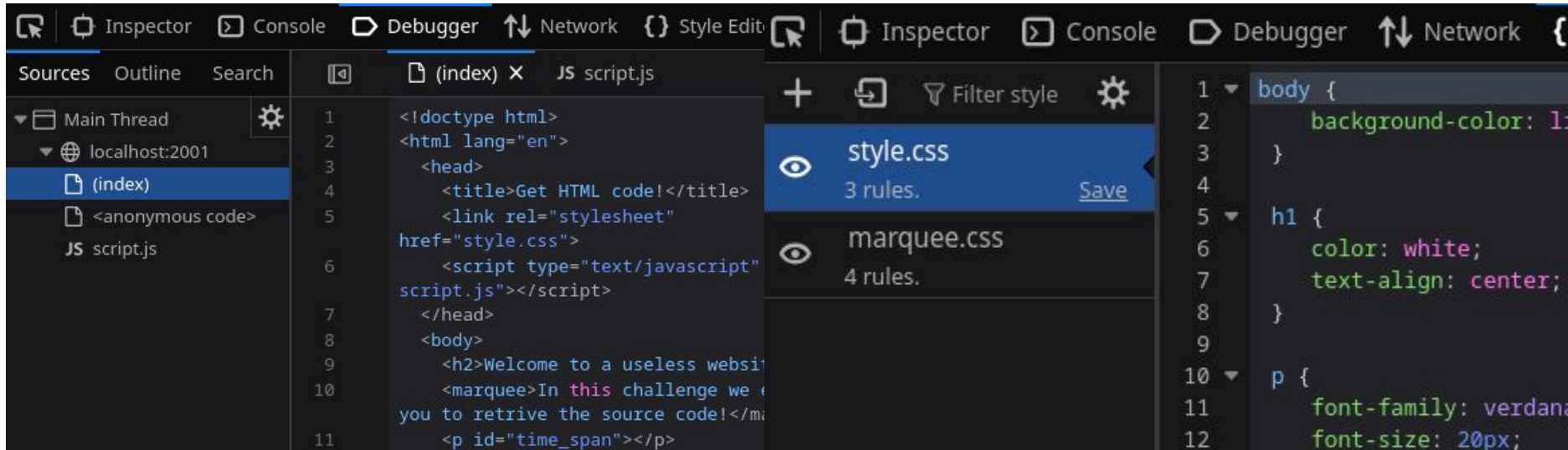
This is an HTML Page!

This is a paragraph

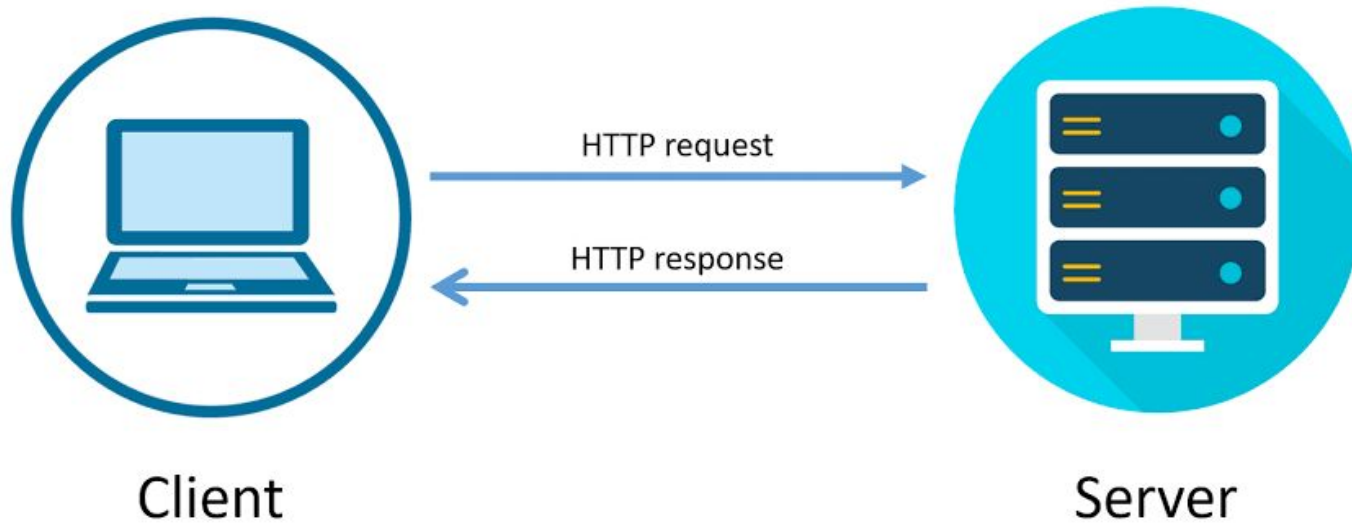
This is a Button

[more about js](https://www.w3schools.com/js/)

Browser Developer Tools - Source Code



How the language works? HTTP!



Getting the Page! (HTTP - Request)

GET / HTTP/2 Host: necst.it User-Agent: curl/7.54.0 Accept: */*	HEADERS
	BODY

Methods

GET used for loading a page and for idempotent actions

POST used to submit forms and for state-changing actions

HEAD, PUT, DELETE, OPTIONS, ...

Request Headers

Host: The hostname. Useful for multiples website on the same IP address. (virtual Hosting)

User-Agent: The Browser type and version

Accept: what MIME type the client expect

Request Body (Only for POST, PUT, and other Methods)

Contains the data (username, password, and more...)

GET Requests do not usually have a body

Getting the Page! (HTTP - Response)

GET / HTTP/2 Host: necst.it User-Agent: curl/7.54.0 Accept: */*	HEADERS	HTTP/2 200 date: Sun, 04 Mar 2018 06:22:06 GMT expires: Thu, 19 Nov 1981 08:52:00 GMT set-cookie: PHPSESSID=AAAAA; path=/; secure; HttpOnly user-agent: Google Chrome content-length: 28046 content-type: text/html; charset=UTF-8	HEADERS
	BODY	<html> <head> <title>Website Title</title> </head> ...	BODY

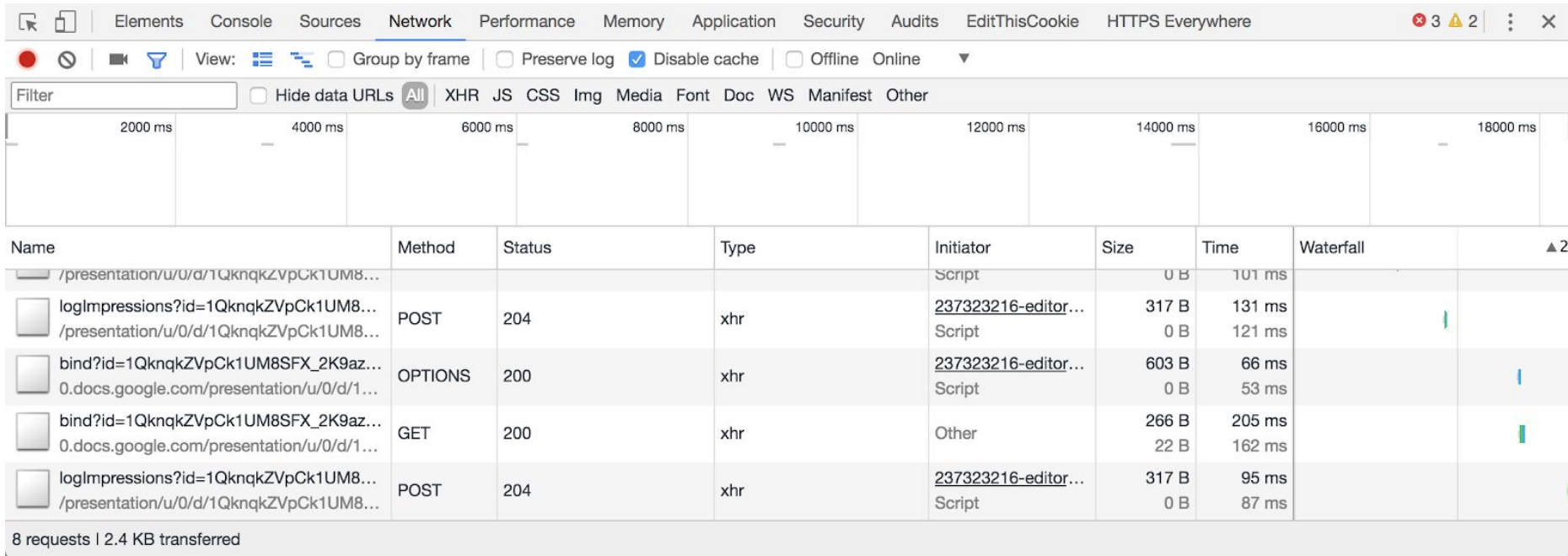
Getting the Page! (HTTP - Response)

Response Codes

- 1xx** Information response
(e.g., 101 Switching Protocol)
- 2xx** Successful response
(e.g., 200 OK)
- 3xx** Redirection messages
(e.g., 302 Found)
- 4xx** Client error responses
(e.g., 404 Not Found)
- 5xx** Server error responses
(e.g., 500 Internal Server Error)

HTTP/2 200 date: Sun, 04 Mar 2018 06:22:06 GMT expires: Thu, 19 Nov 1981 08:52:00 GMT set-cookie: PHPSESSID=AAAAA; path=/; secure; HttpOnly user-agent: Google Chrome content-length: 28046 content-type: text/html; charset=UTF-8	HEADERS
<html> <head> <title>Website Title</title> </head> ...	BODY

In Browser Developer Tools



Getting the Page! (HTTP - Request)

GET / HTTP/2 Host: necst.it User-Agent: curl/7.54.0 Accept: */* Cookie: PHPSESSID=AAAAAA	HEADERS
	BODY

Cookies

- Browsers save cookies
- Send them on further interactions with a server in the Request Headers



Getting the Page! (HTTP - Response)

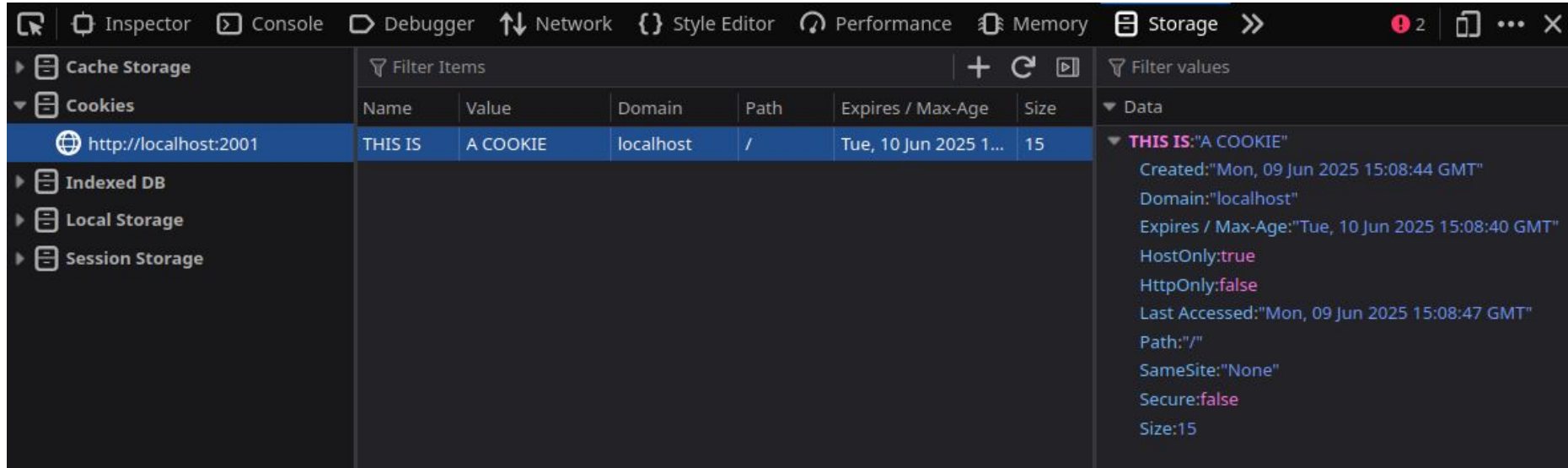
Cookies

- HTTP is a stateless protocol
- Unique strings assigned by the server
- Used to "identify" a user over multiple connections
- User profiling
- Authentication handling



HTTP/2 200 date: Sun, 04 Mar 2018 06:22:06 GMT expires: Thu, 19 Nov 1981 08:52:00 GMT set-cookie: PHPSESSID=AAAAA; path=/; secure; HttpOnly x-frame-options: SAMEORIGIN content-length: 28046 content-type: text/html; charset=UTF-8	HEADERS
<html> <head> <title>Website Title</title> </head> ...	BODY

Browser Developer Tools - Console



The screenshot shows the Chrome DevTools Storage panel. The left sidebar lists storage types: Cache Storage, Cookies, Indexed DB, Local Storage, and Session Storage. The Cookies section is expanded, showing a table of cookies for the domain http://localhost:2001. The table has columns for Name, Value, Domain, Path, Expires / Max-Age, and Size. One cookie is listed with the name 'THIS IS', value 'A COOKIE', domain 'localhost', path '/', and expires on 'Tue, 10 Jun 2025 1...'. The right pane shows the details for the selected cookie, including its creation time, domain, expiration, and various attributes like HostOnly, HttpOnly, Last Accessed, Path, SameSite, Secure, and Size.

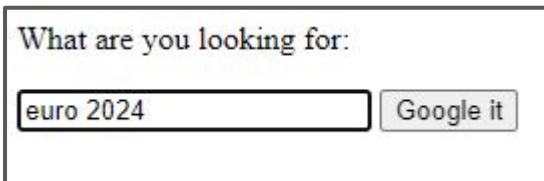
Name	Value	Domain	Path	Expires / Max-Age	Size
THIS IS	A COOKIE	localhost	/	Tue, 10 Jun 2025 1...	15

THIS IS: "A COOKIE"

- Created: "Mon, 09 Jun 2025 15:08:44 GMT"
- Domain: "localhost"
- Expires / Max-Age: "Tue, 10 Jun 2025 15:08:40 GMT"
- HostOnly: true
- HttpOnly: false
- Last Accessed: "Mon, 09 Jun 2025 15:08:47 GMT"
- Path: "/"
- SameSite: "None"
- Secure: false
- Size: 15

How can we easily allow user to send data?

```
1.  <html>
2.    <head>
3.      <title>Google Page</title>
4.    </head>
5.    <body>
6.      <form method="GET" action="https://google.com/search" >
7.        <p>What are you looking for:</p>
8.        <input type="text" name="q">
9.        <input type="submit" value="Google it">
10.      </form>
11.    </body>
12. </html>
```



What are you looking for:

<https://google.com/search?q=euro+2024>

The browser is the tool

Three panels do everything this session needs

Inspector reads and edits the page you were served, hidden fields included

Network shows every request and response, and resends one you have edited

Console sends requests of your own, with any method you like

```
fetch('/', {method: 'POST'}).then(r => r.text()).then(console.log)
fetch('/', {method: 'HEAD'}).then(r => console.log(r.headers.get('Flag')))
document.cookie
```

No terminal, no proxy, nothing to install.

Reqbin.com

Online REST & SOAP API Testing Tool

ReqBin is an online API testing tool for REST and SOAP APIs. Test API endpoints by making API requests directly from your browser. Test API responses with built-in JSON and XML validators. Load test your API with hundreds of simulated concurrent connections. Generate code snippets for API automation testing frameworks. Share and discuss your API requests online.

GET US Send

Content Authorization Headers Raw

JSON (application/json)

```
{"key": "value"}
```

Webhook.site

Webhook.site

Docs & API

Features & Pricing

Terms, Privacy & Security

Support

Copy

Edit

+ New

Login

Sign Up Now

f5a50e98

Share

Schedule

Form Builder

CSV Export

Custom Actions

Replay

XHR Redirect

Redirect Now

More

INBOX (0/100) Newest First

Search Query

Waiting for first request

Your unique URL

Copy to clipboard

`https://webhook.site/f5a50e98-1148-4266-8c63-a3a6e9c1ac80`

Open in new tab

Examples

Your unique email address

`f5a50e98-1148-4266-8c63-a3a6e9c1ac80@emailhook.site`

Open in mail client

Your unique DNS name

`*.f5a50e98-1148-4266-8c63-a3a6e9c1ac80.dnshook.site`

About DNSHook

Proxy bidirectionally with Webhook.site CLI

```
$ whcli forward --token=f5a50e98-1148-4266-8c63-a3a6e9c1ac80 --target=https://localhost
```

Info & installation

Star on GitHub

Follow @webhooksite on X

What is Webhook.site?

Webhook.site generates free, unique URLs and e-mail addresses and lets you see everything that's sent there instantly.

Why upgrade to a Webhook.site account?

Unlimited requests, emails, DNSHooks

Receive endless webhooks with addresses that can be managed in your account and never expire. View a history of the latest 10,000 items.

Forward and Transform

Transform data and resend to different URLs or emails – with retry and error notifications. Fetch data with our easy to use API.

Workflows and Automations

Click, drag and drop or use AI to create workflows that run on each incoming request. Go back and see what happened and run again with Error Log, Replay and notifications.

Schedules

Set your workflows to run every minute, day, week. Use as an uptime & SSL monitor. Cronjob syntax can be used. [More info](#)

Native integrations including Google Sheets, Excel, Slack, S3, Dropbox, SFTP, push notifications, databases, and much more.



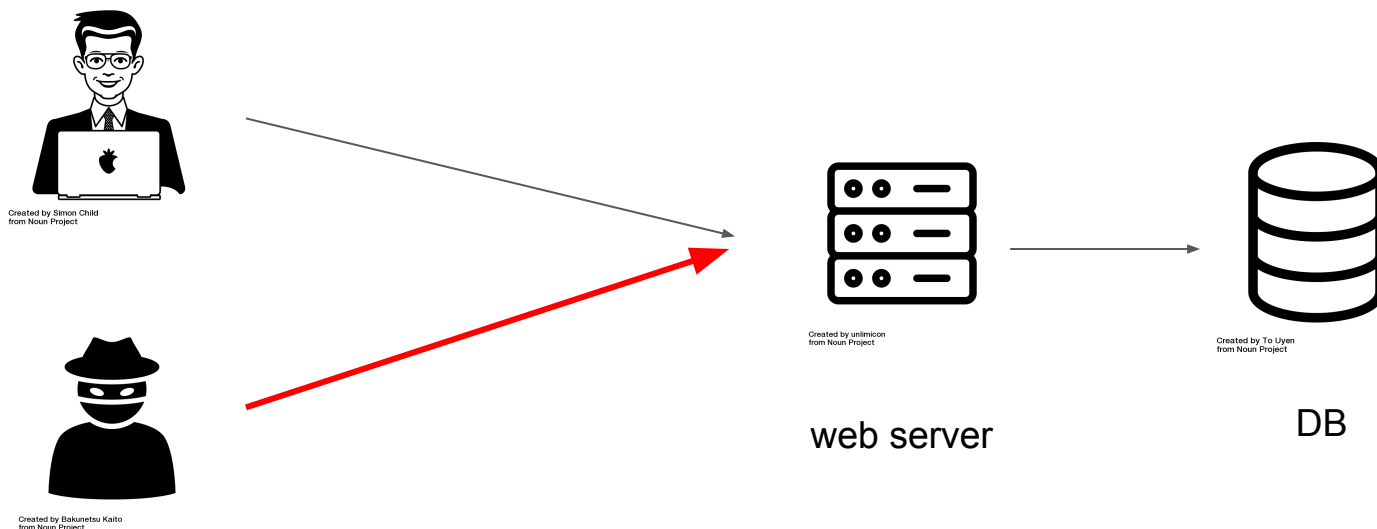
POLITECNICO
MILANO 1863



Web 101: Server Attacks

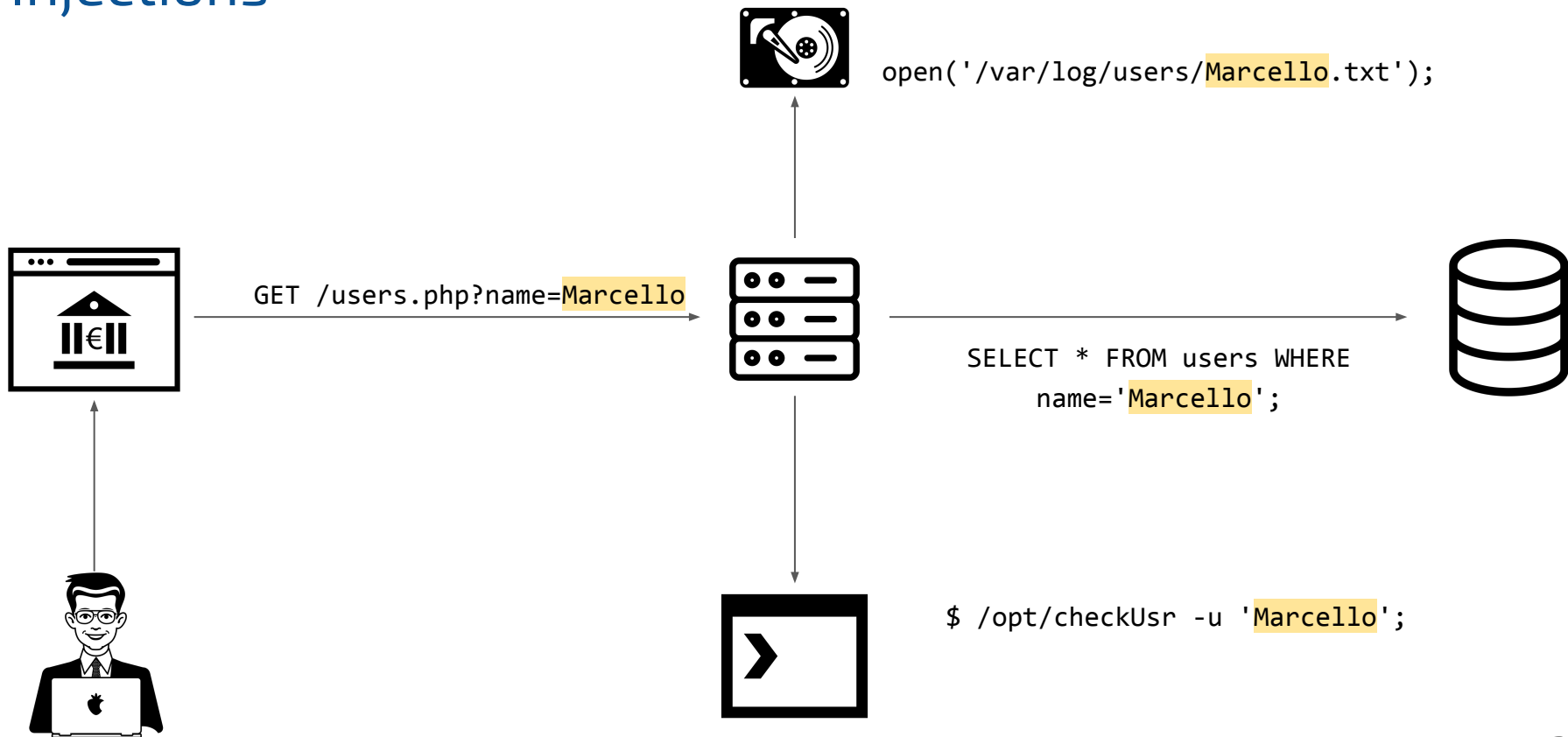
Injectons

Malicious user



Goal: compromise, directly, the server(s)

Injections



Never trust the user's input!

GET\POST data, headers (Referer), cookies

Data that we fetch from other servers

Data that we previously stored (where did it come?)

**... needs to be checked and filtered always and properly,
otherwise bad things can (and will) happen**

DEMO

Instant Transfers

Questions?