6. Sicurezza delle Applicazioni Web

Sicurezza dell'Informazione

URL (Uniform Resource Locator)

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

Schema Autenticazione Dominio Porta Percorso Parametri Ancora

Cosa succede quando ci si connette ad un sito web

- Risoluzione DNS
- Connessione al server (un computer)
- Richiesta HTTP(s)
- Risposta HTTP(s)
- La pagina viene mostrata

Risoluzione DNS

Dove abita Mario Rossi?

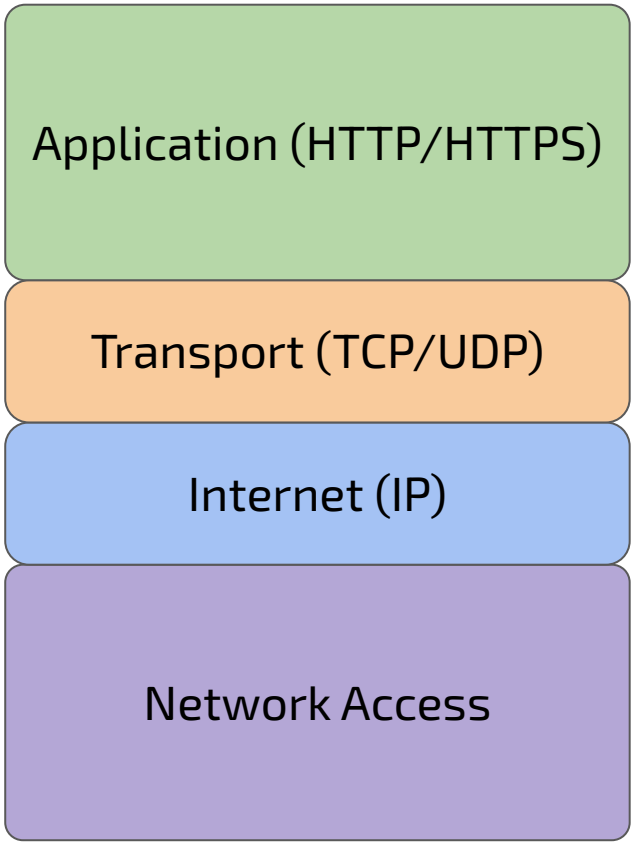
Via Roma 3, MI, Milano

Dov'è google.com?

8.8.12.132

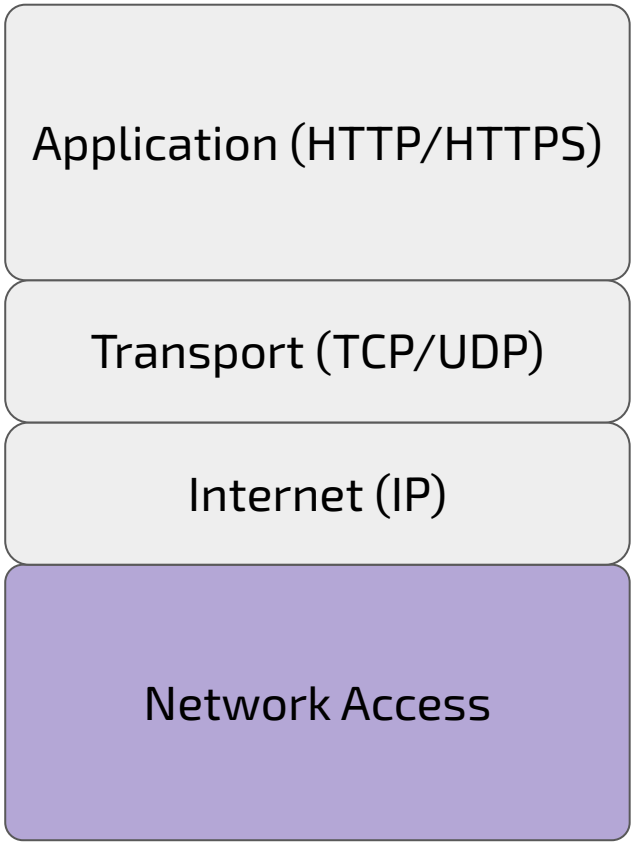
Connessione al server

Stack TCP/IP



Include tutti i protocolli che permettono a due computer di mandare e ricevere messaggi tramite internet.

Livello Network Access



Protocolli che gestiscono la trasmissione fisica dei dati sulla rete.

Include gli standard che definiscono come i pacchetti IP vengono inviati sul mezzo fisico (cavo, Wi-Fi, fibra, ecc.).

Indirizzo fisico MAC

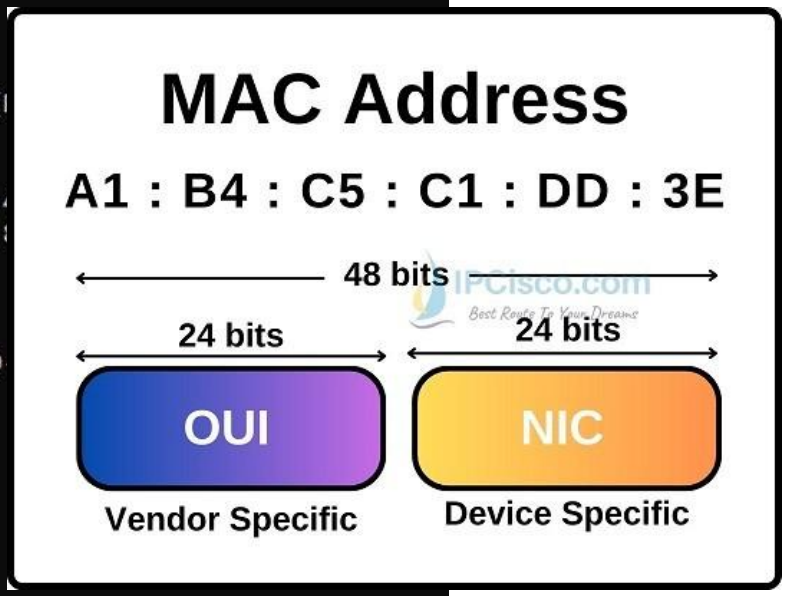
Media **A**ccess **C**ontrol Address

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
   Physical Address. . . . . . . . . : 98-43-FA-28-3C-14
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::c41f:eaf3:8e1a:c2d7%21(
   IPv4 Address. . . . . . . . . . . : 10.0.0.32(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Monday, September 19, 2022 8:
   Lease Expires . . . . . . . . . . : Saturday, September 24, 2022
   Default Gateway . . . . . . . . . : 10.0.0.1
   DHCP Server . . . . . . . . . . . : 10.0.0.1
   DHCPv6 IAID . . . . . . . . . . . : 362300410
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-29-01-52-96-A0-29
   DNS Servers . . . . . . . . . . . : 10.0.0.1
   NetBIOS over Tcpip. . . . . . . . : Enabled

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Bluetooth Device (Personal Area Network)
   Physical Address. . . . . . . . . : 98-43-FA-28-3C-18
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
```



Livello Internet

Application (HTTP/HTTPS)

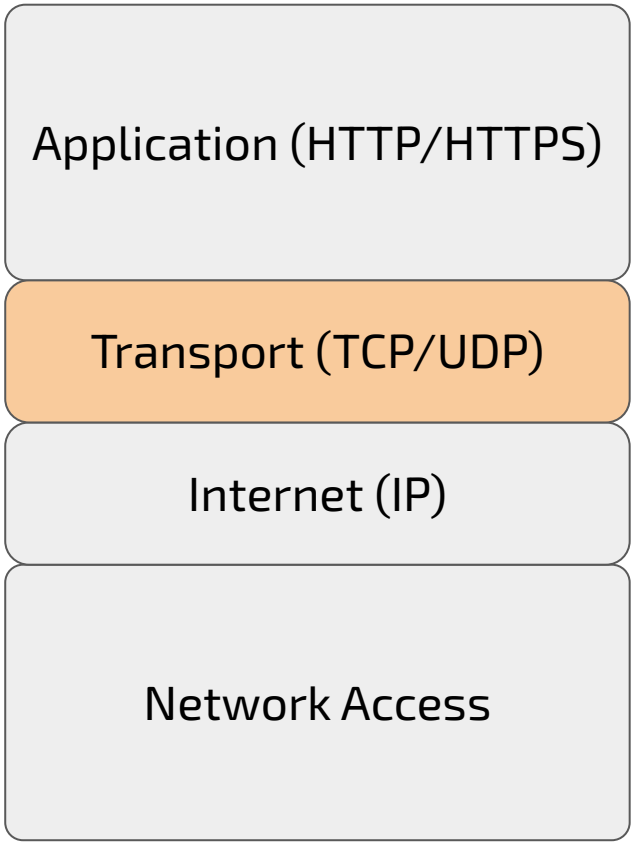
Transport (TCP/UDP)

Internet (IP)

Network Access

Protocollo che gestisce l'instradamento dei pacchetti tra reti internet diverse.

Livello di Trasporto



Protocolli che gestiscono la comunicazione logica tra applicazioni.

TCP: protocollo affidabile con controllo degli errori.

UDP: protocollo più veloce, ma non garantisce l'ordine o la consegna dei pacchetti.

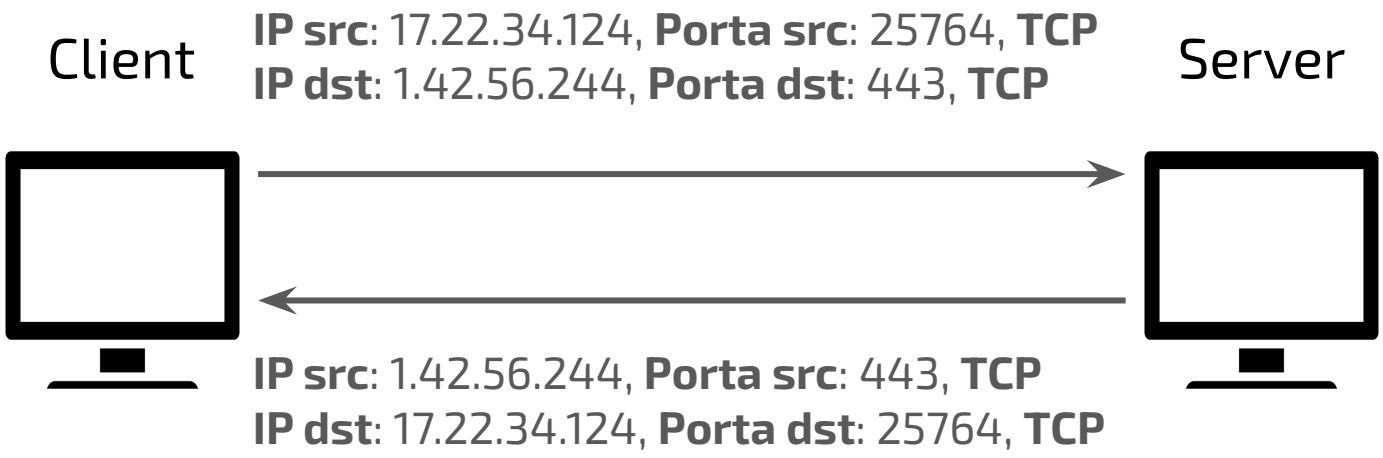
Porte del livello di trasporto

Ogni computer/smartphone può e deve aprire più di una connessione a diversi server. Per evitare logiche complesse, ogni connessione parte da una porta “logica”; un numero che identifica la specifica connessione.

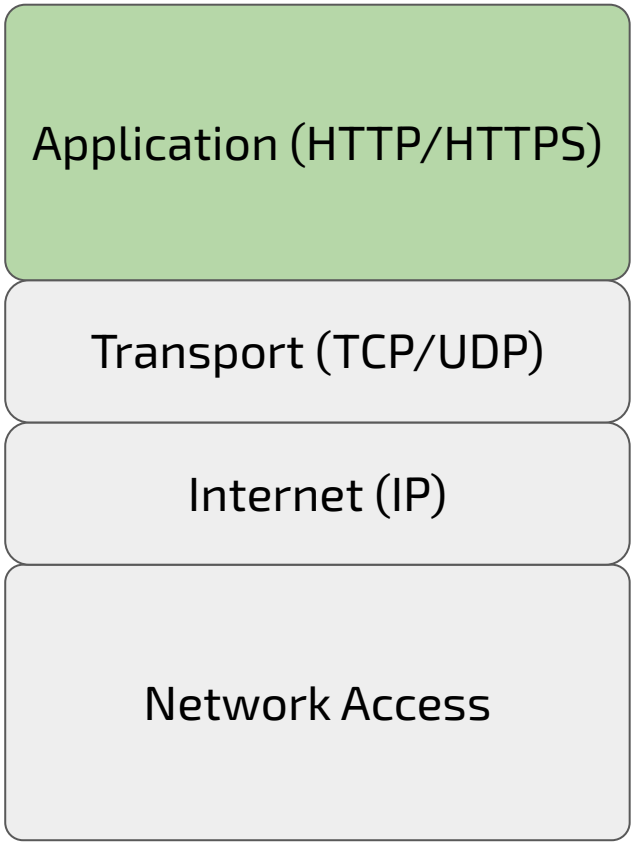
Il server che riceve il messaggio, lo riceve ad una specifica porta:

- HTTP: 80
- HTTPS: 443
- SSH: 22
- DNS: 53

IP e porta


Client
IP src: 17.22.34.124, **Porta src**: 25764, **TCP**
IP dst: 1.42.56.244, **Porta dst**: 443, **TCP**
Server
IP src: 1.42.56.244, **Porta src**: 443, **TCP**
IP dst: 17.22.34.124, **Porta dst**: 25764, **TCP**

Livello Applicativo



Protocolli che definiscono come le applicazioni scambiano dati tra computer e dispositivi.

Protocollo HTTP (HyperText Transfer Protocol)

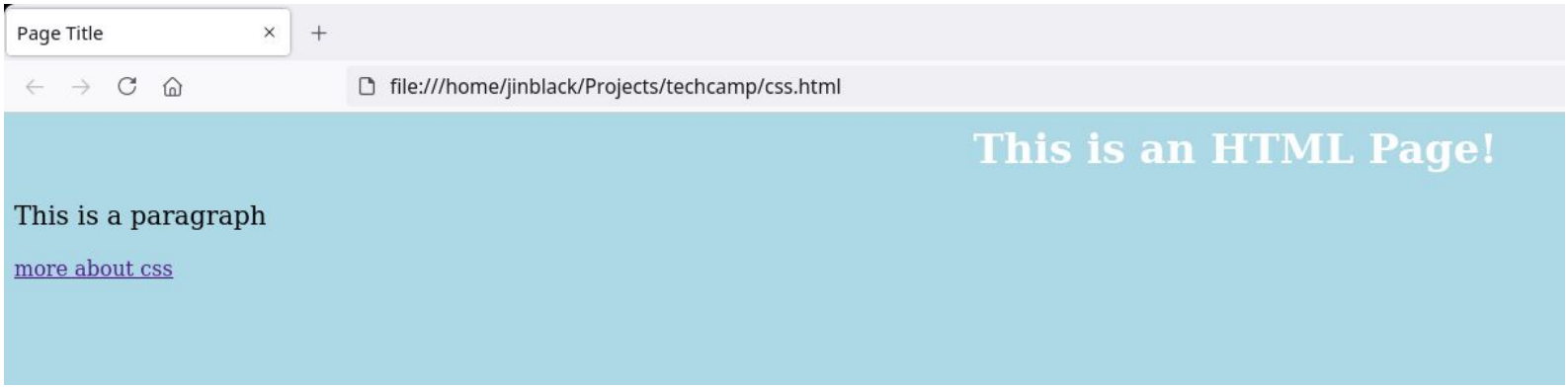
Quello che riceviamo quando apriamo una connessione!

Codice HTML (HyperText Markup Language

```
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>This is an HTML Page!</h1>
        <p>This is a paragraph</p>
        <a href="https://www.w3schools.com/html/">
            more about html
        </a>
    </body>
</html>
```

Abbellire una pagina HTML

```
1.  <html>
2.      <head>
3.          <title>Page Title</title>
4.      </head>
5.      <body style="background-color: lightblue;">
6.          <h1 style="color: white; text-align: center;">This is an HTML
    Page!</h1>
7.          <p style="font-family: verdana; font-size: 20px;">This is a
    paragraph</p>
8.          <a href="https://www.w3schools.com/css/">
9.              more about css
10.         </a>
11.     </body>
12. </html>
```



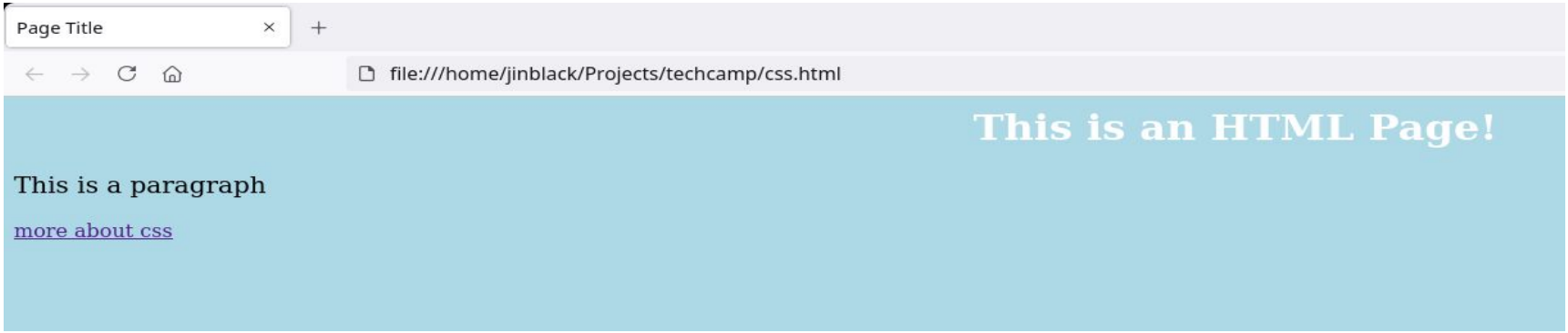
Abbellire una pagina HTML

```html
<html>
    <head>
        <title>Page Title</title>
        <link rel="stylesheet" href="css.css">
    </head>
    <body>
        <h1>This is an HTML Page!</h1>
        <p>This is a paragraph</p>
        <a href="https://www.w3schools.com/css/">
            more about css
        </a>
    </body>
</html>
```

```css
body {
    background-color: lightblue;
}

h1 {
    color: white;
    text-align: center;
}

p {
    font-family: verdana;
    font-size: 20px;
}
```



Pagina HTML dinamica (JavaScript)

```html
<html>
    <head>
        <title>Page Title</title>
        <script src="js.js"></script>
    </head>
    <body>
        <h1>This is an HTML Page!</h1>
        <p id="demo">This is a paragraph</p>
        <button type="button" onclick="changeColor()">This is a Button</button>
        </br>
        <a href="https://www.w3schools.com/js/">
            more about js
        </a>
    </body>
</html>
```

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

Pagina HTML dinamica (JavaScript)

```html
<html>
    <head>
        <title>Page Title</title>
        <script src="js.js"></script>
    </head>
    <body>
        <h1>This is an HTML Page!</h1>
        <p id="demo">This is a paragraph</p>
        <button type="button" onclick="changeColor()">This is a Button</button>
        </br>
        <a href="https://www.w3schools.com/js/">
            more about js
        </a>
    </body>
</html>
```

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

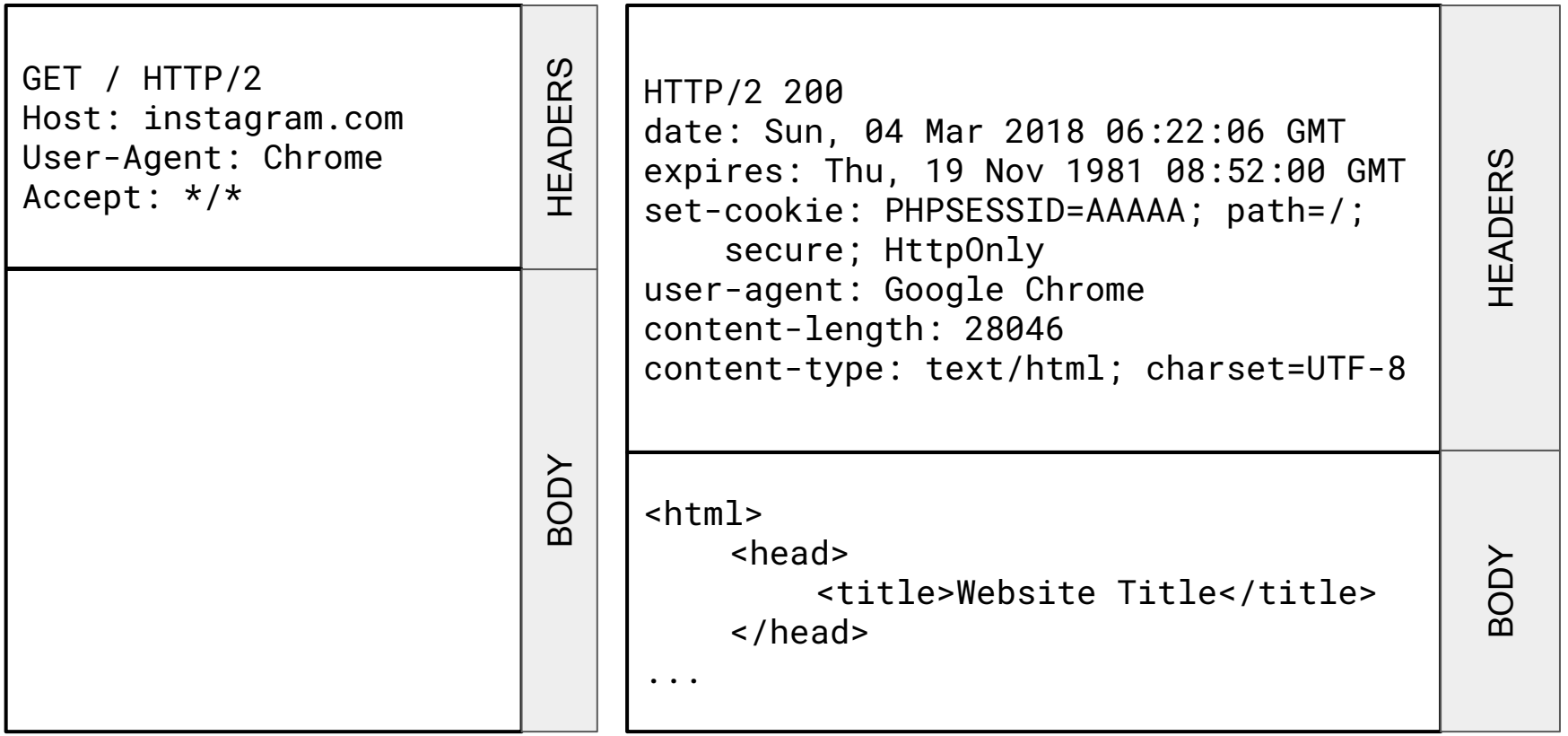
Richiesta HTTP

Metodi
GET utilizzato per richiedere una pagina.
POST utilizzato per inviare dati.
HEAD, PUT, DELETE, OPTIONS, ...

Intestazione HTTP
Host : Il nome del sito web.
User-Agent: Il browser utilizzato.
Cookie: Utilizzati per sessioni web e tracciamenti commerciali.

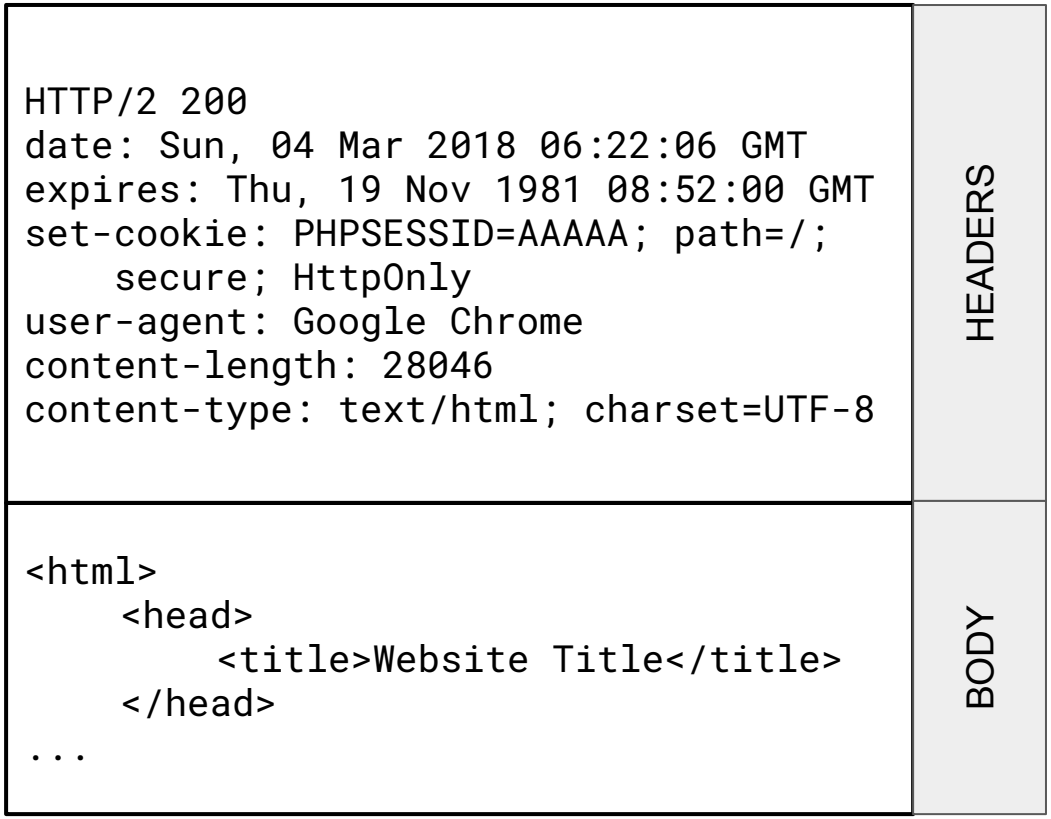
Corpo HTTP (Solo POST, PUT, e altri Medie)
Contiene le informazioni (username, password, e altro…).
Richieste GET in genere non hanno il corpo.

Risposta HTTP



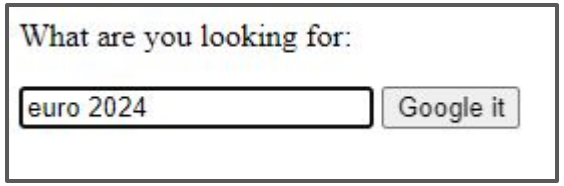
Risposta HTTP

Codici di Risposta
1xx Informazione Generica
(es, 101 Switching Protocol)
2xx Risposta Affermativa
(es, 200 OK)
3xx Messaggi di Ridirezione
(es, 302 Found)
4xx Errori nella richiesta HTTP
(es, 404 Not Found)
5xx Errori del Server
(es, 500 Internal Server Error)



Mandare dati ad un server

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



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

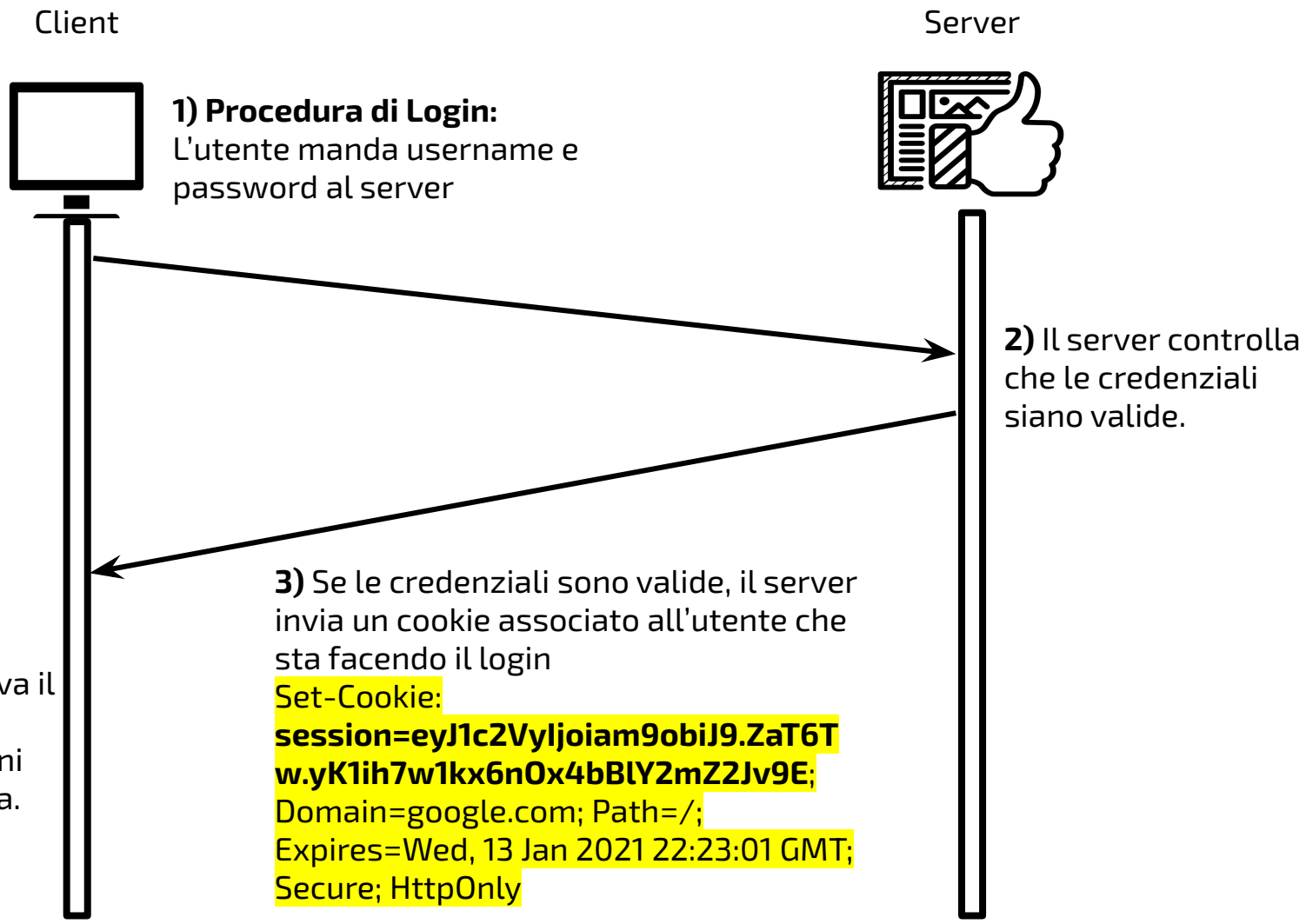
Codifica URL

- Alcuni caratteri nell'URL sono **riservati.**
  - Come facciamo se vogliamo mettere un "#" nei parametri di un URL?
- Codifica URL: % seguito dai due valori esadecimali (nella codifica ASCII) del carattere scelto.
  - Per esempio, il carattere "#" si scriverebbe %23.
- I caratteri riservati vanno obbligatoriamente codificati con la codifica URL.
- Gli spazi possono essere rappresentati con %20 o il segno più(+)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Cookie


Client
Server
1) Procedura di Login:
L'utente manda username e password al server
2) Il server controlla che le credenziali siano valide.
3) Se le credenziali sono valide, il server invia un cookie associato all'utente che sta facendo il login
Set-Cookie:
session=eyJ1c2VyIjoiam9obiJ9.ZaT6Tw.yK1ih7w1kx6nOx4bBlY2mZ2Jv9E;
Domain=google.com; Path=/;
Expires=Wed, 13 Jan 2021 22:23:01 GMT;
Secure; HttpOnly
4) Il browser dell'utente salva il cookie e lo rimanda ad ogni richiesta futura.

Base64 encoding

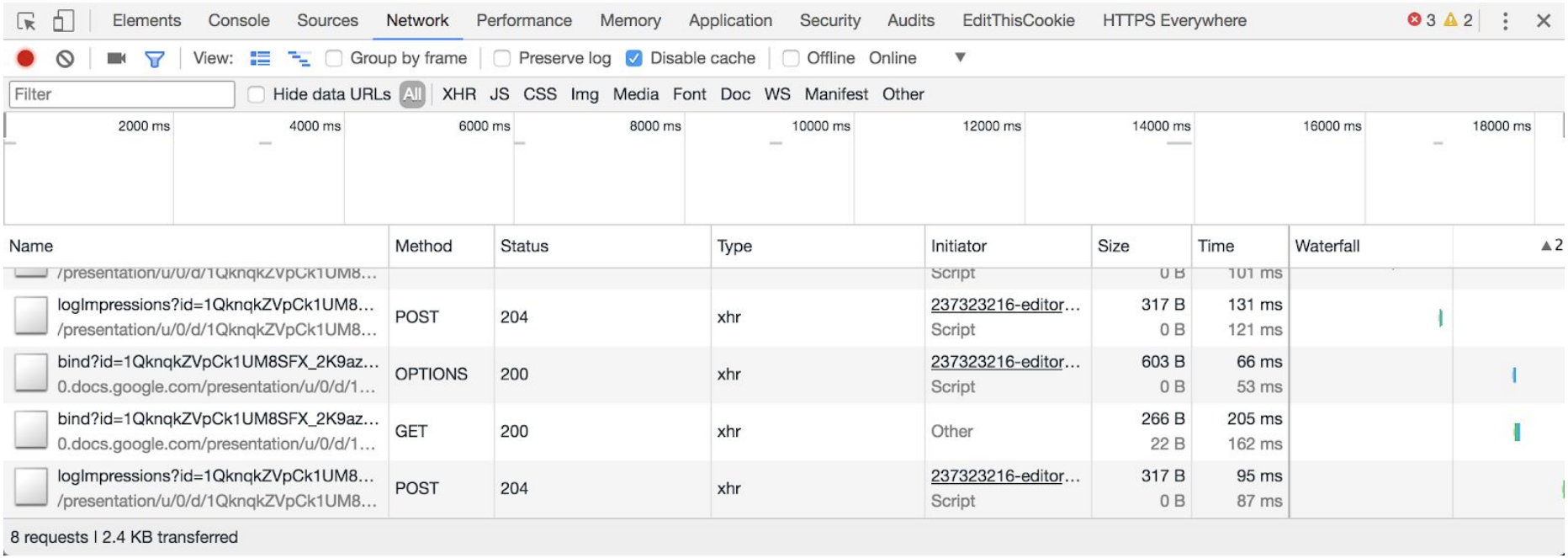
Valore	ASCII
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H
8	I
9	J
10	K
11	L
12	M
13	N
14	O
15	P

Valore	ASCII
16	Q
17	R
18	S
19	T
20	U
21	V
22	W
23	X
24	Y
25	Z
26	a
27	b
28	c
29	d
30	e
31	f

Valore	ASCII
32	g
33	h
34	i
35	j
36	k
37	l
38	m
39	n
40	o
41	p
42	q
43	r
44	s
45	t
46	u
47	v

Valore	ASCII
48	w
49	x
50	y
51	z
52	0
53	1
54	2
55	3
56	4
57	5
58	6
59	7
60	8
61	9
62	+
63	/

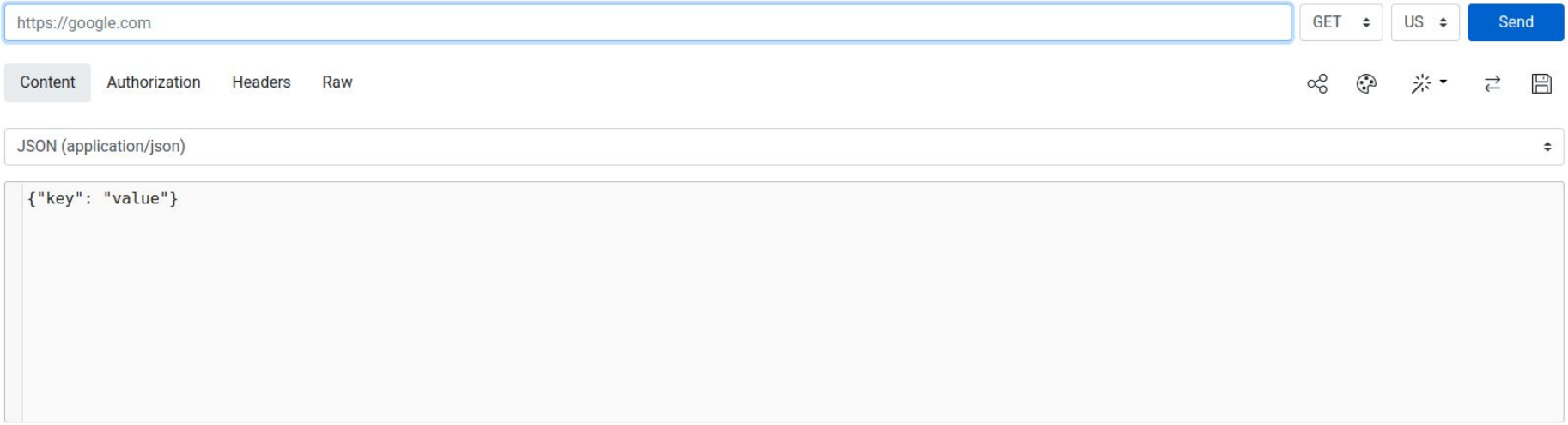
In Browser Developer Tools



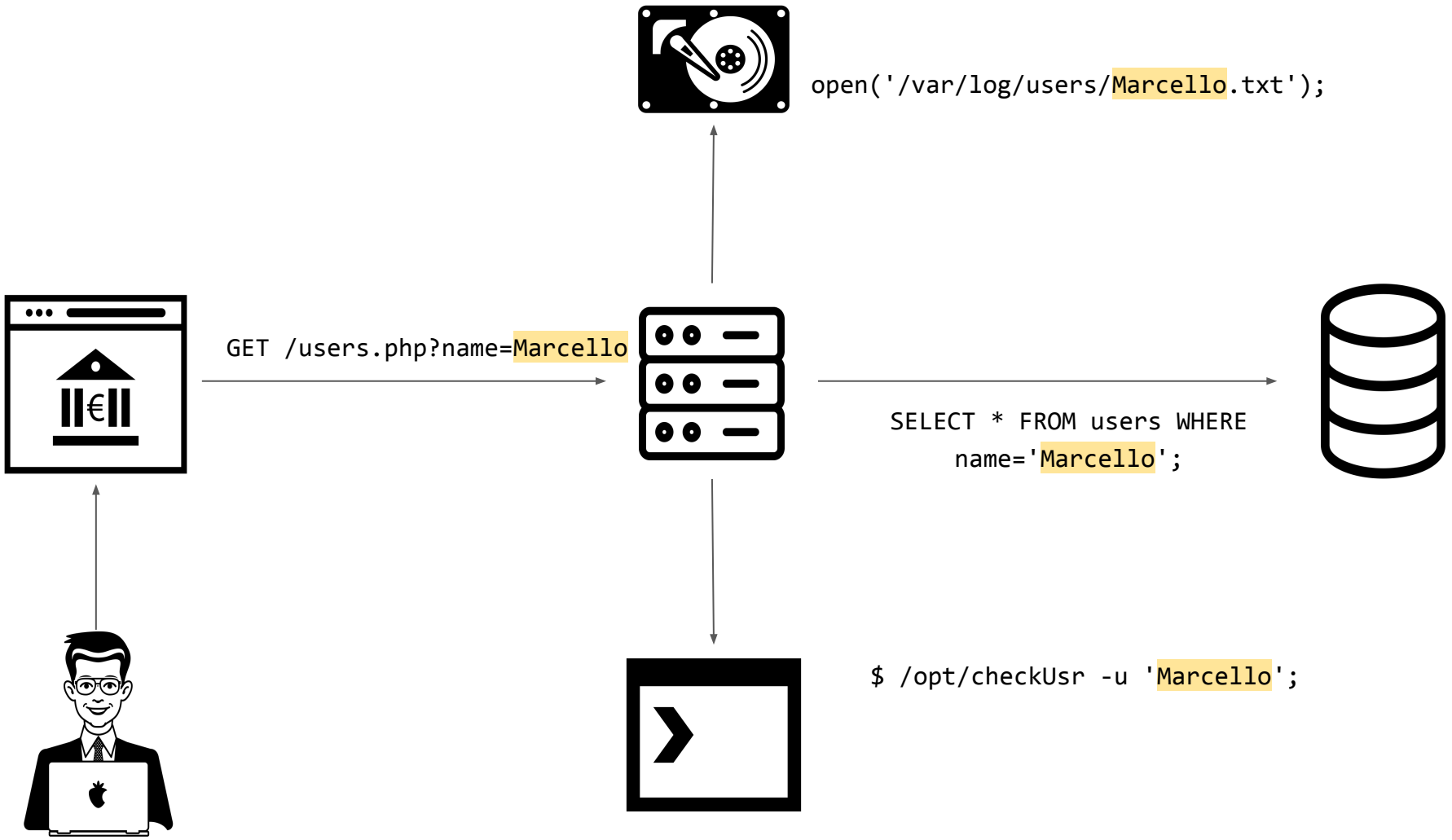
Reqbin.com

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.



Injections


open('/var/log/users/Marcello.txt');
GET /users.php?name=Marcello
SELECT * FROM users WHERE
name='Marcello';
$ /opt/checkUsr -u 'Marcello';

Mai fidarsi dell'input dell'utente!

I dati provenienti da richieste GET/POST, intestazioni (Referer), cookie,

dati che recuperiamo da altri server,

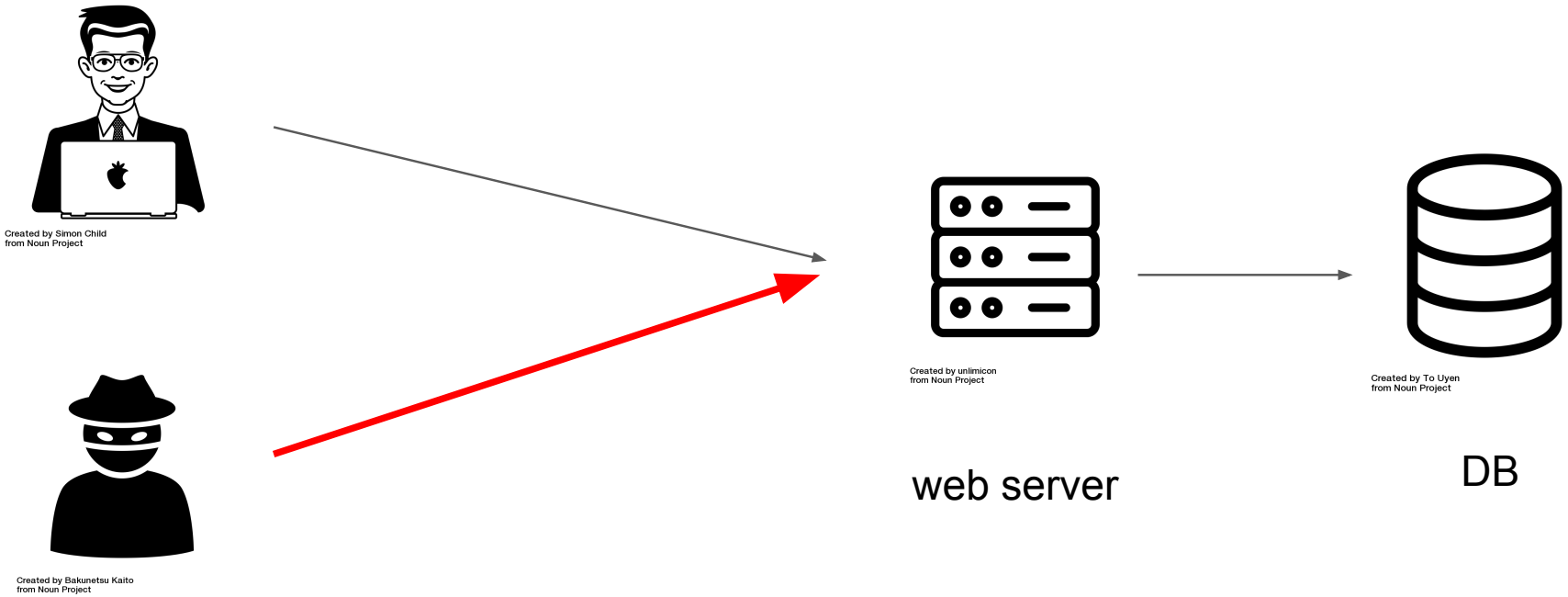
dati che abbiamo salvato in precedenza (ma da dove provenivano?)

… devono sempre essere controllati e filtrati in modo corretto,

altrimenti possono (e sicuramente accadranno) conseguenze indesiderate.

Attacchi Lato Server

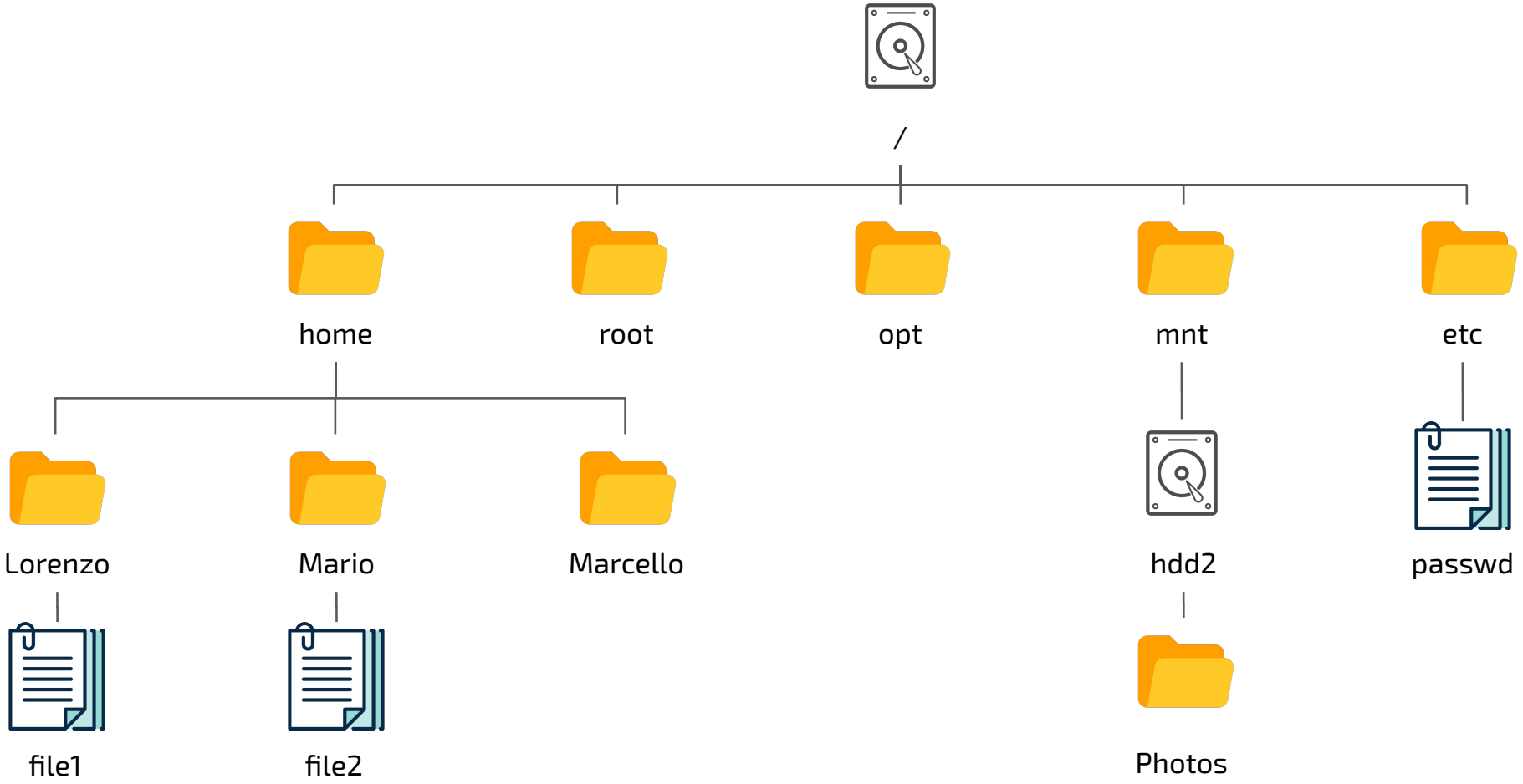
Attacchi Lato Server



Goal: compromettere/attaccare direttamente il server.

Path Traversal

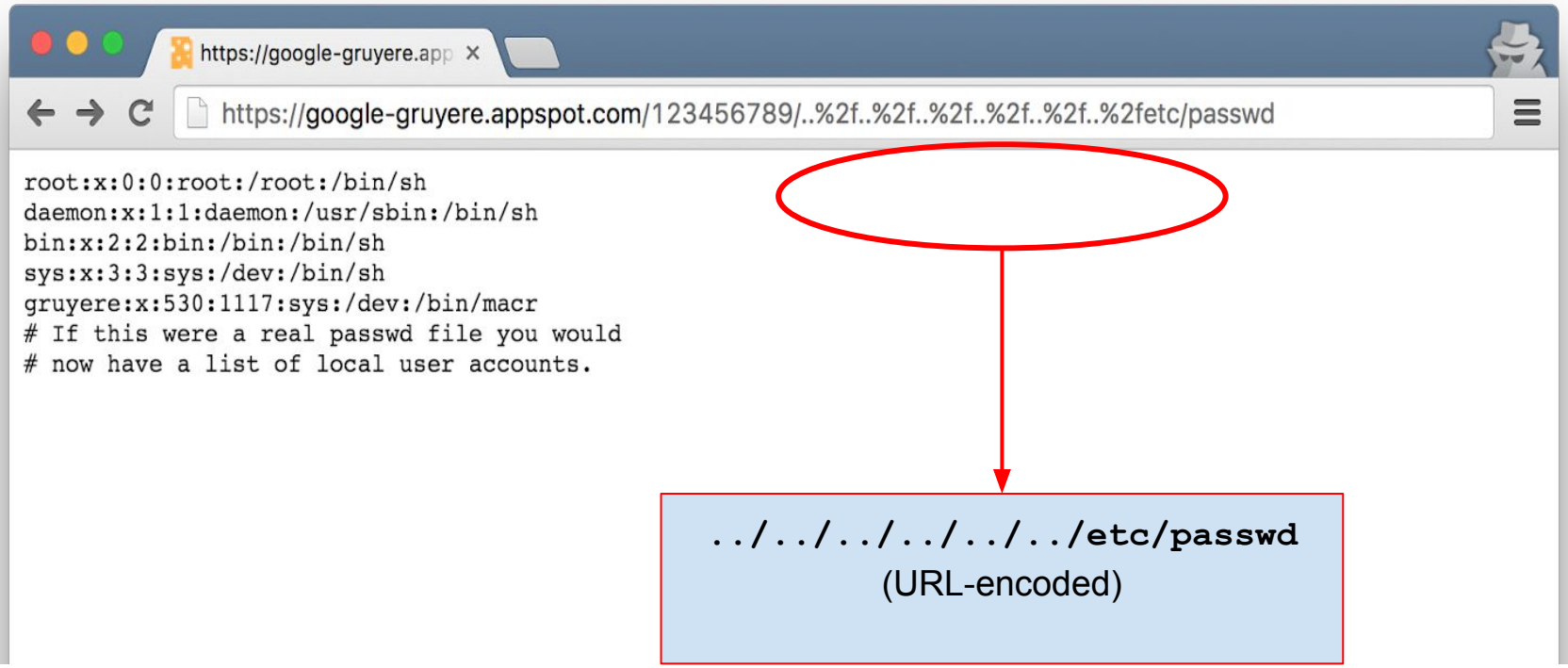
Recall: Percorsi relativi



Dalla cartella **Mario**: `../../etc/passwd` (Relativo)

Equivalente a: `/etc/passwd` (Assoluto)

Path Traversal



L'attaccante sfrutta un mancato controllo dell'input per accedere ad un file sul computer.

Database & SQL-Injections

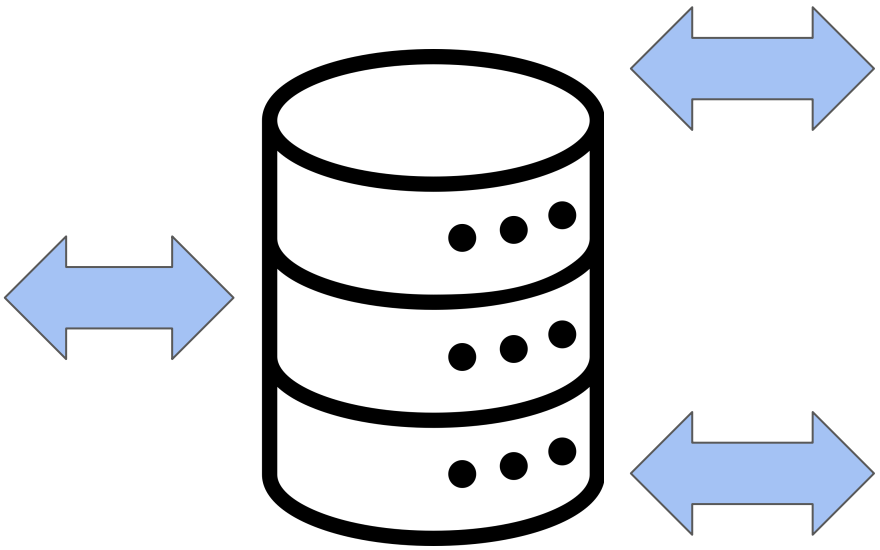
Dove vengono salvati i dati?

amazon

TikTok

Cos'è un Database?

DATI



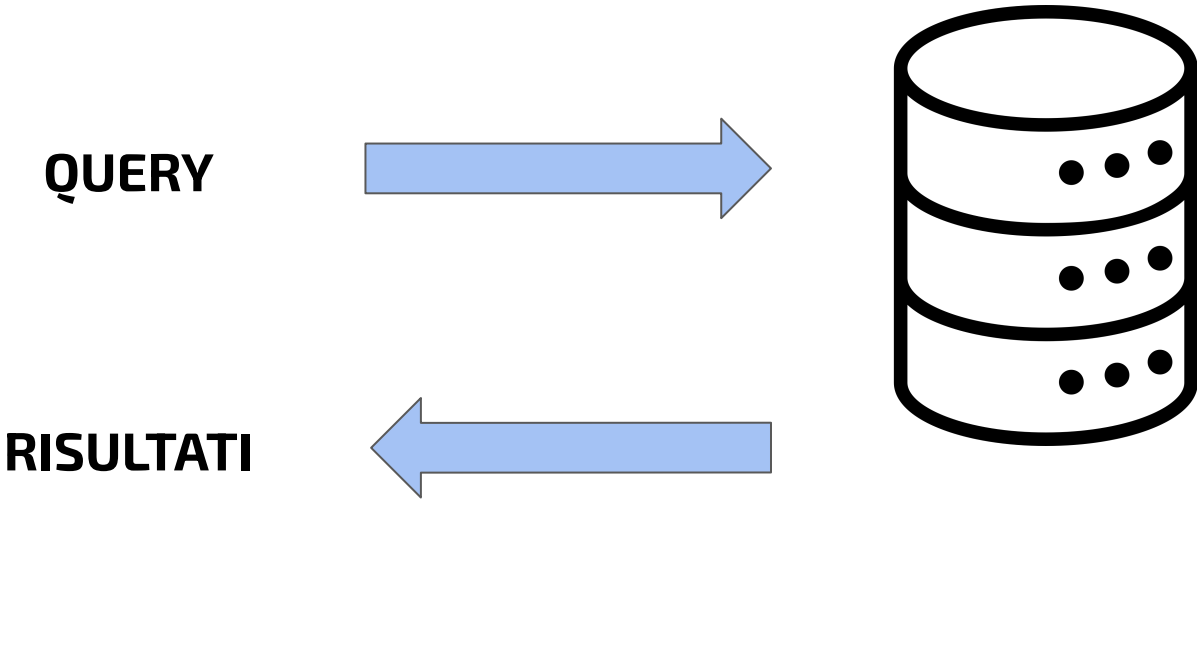
UTENTI			
User_id	**Username**	**Email**	**Password**
001	Law	…	…
002	Bertados	…	…

PRODOTTI		
Prod_id	**Nome**	**Descrizione**
001	iPhone 14	…
002	Playstation 5	…

Come comunicare con un database?

Possiamo **interrogare** un database con un **linguaggio di interrogazione**.

Esempio: Datalog, **SQL**, …



SQL

SQL sta per **Structured Query Language** (linguaggio di interrogazione strutturata).

Cosa può fare?

1. Creare e modificare database
2. Inserire e modificare i dati
3. **Interrogare** il database
4. ...

La più semplice query SQL

```
SELECT colonna/e

FROM tabella;
```

La più semplice query SQL - Esempio

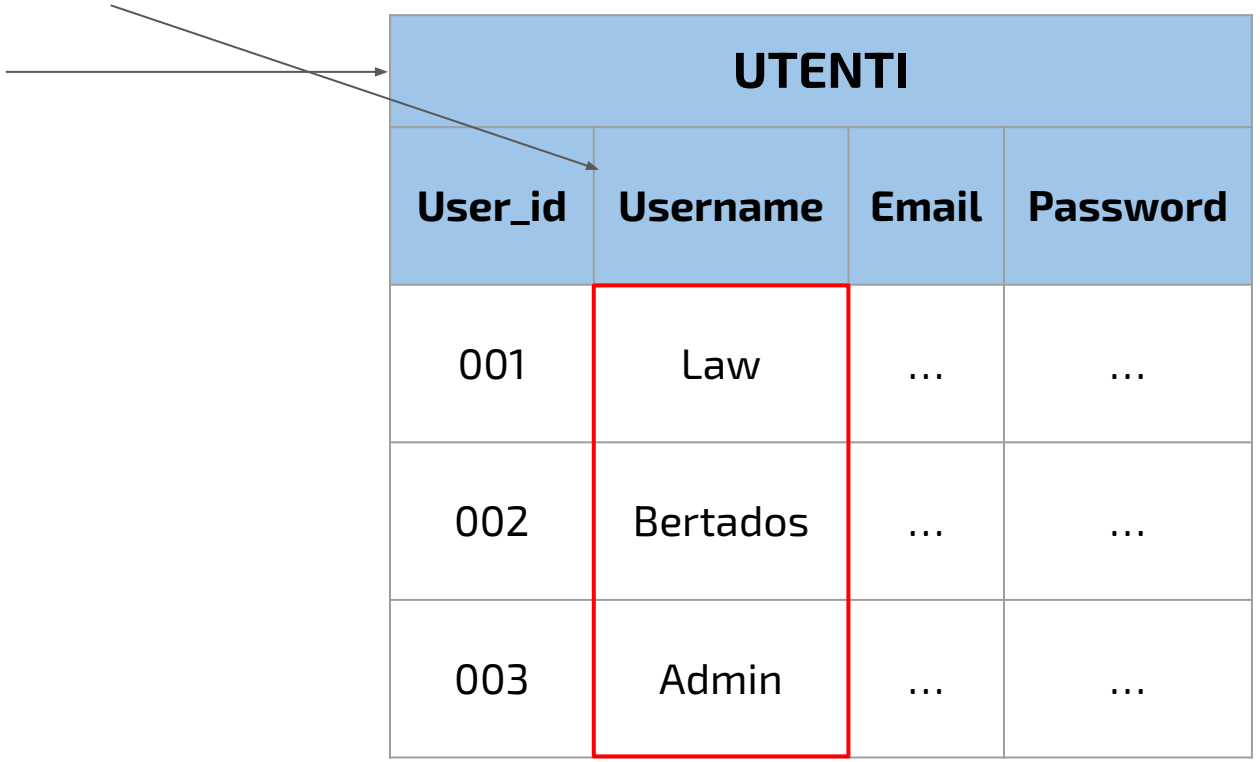
```
SELECT Username
FROM UTENTI;
```

Risultato?

Law

Bertados

Admin



UTENTI			
User_id	**Username**	**Email**	**Password**
001	Law	…	…
002	Bertados	…	…
003	Admin	…	…

Come fare una ricerca più precisa?

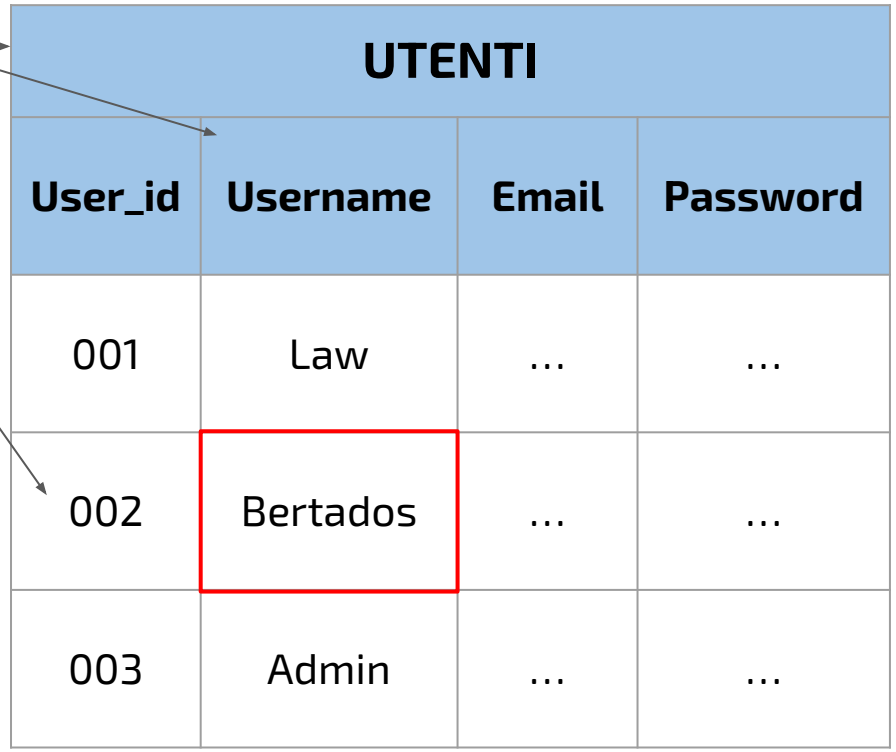
```
SELECT colonna/e

FROM tabella

WHERE condizione
```

La più semplice query SQL - Esempio

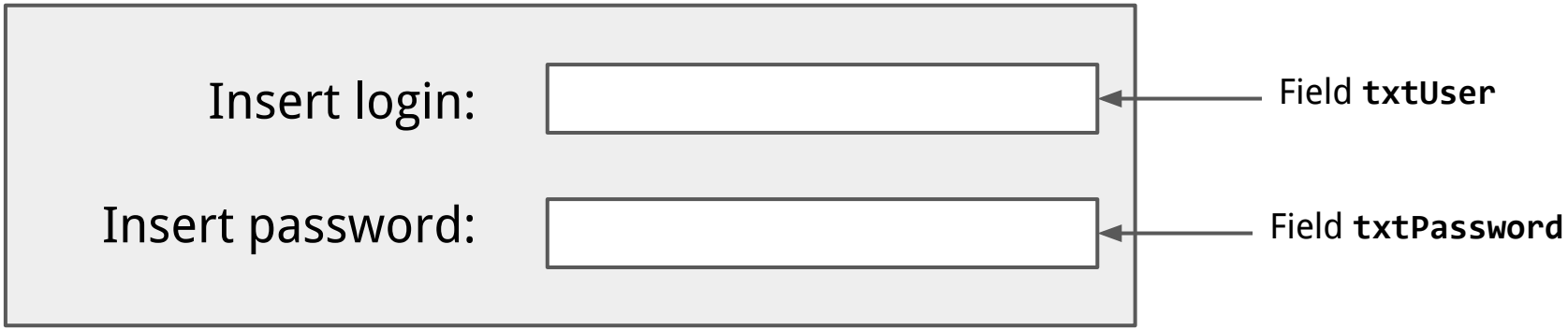
```
SELECT Username
FROM UTENTI
WHERE User_id = 002;
```



UTENTI			
User_id	**Username**	**Email**	**Password**
001	Law	…	…
002	Bertados	…	…
003	Admin	…	…

Risultato?
Bertados

SQL Injection



Diamo un'occhiata a cosa fa il server

```
username = request.form['username']
password = request.form['password']

db_query = "SELECT * FROM users WHERE username='"+username+"' AND
password='"+password+"'"

db_conn = get_db_conn()
cur = db_conn.cursor()
cur.execute(db_query)
res = cur.fetchall()

if res:
   do_login()
```

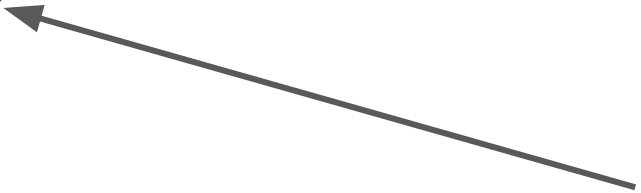
Diamo un'occhiata a cosa fa il server

```
username = request.form['username']
password = request.form['password']

db_query = "SELECT * FROM users WHERE username='"+username+"' AND
password='"+password+"'"

db_conn = get_db_conn()
cur = db_conn.cursor()
cur.execute(db_query)
res = cur.fetchall()

if res:
   do_login()
```



Attenzione: MAI salvare o controllare le password in questo modo, ci sono modi più sicuri per farlo!

Concentriamoci sull'interrogazione

```
SELECT *
FROM users
WHERE username='Law' AND password='pippo';
```

Qui finisce il nostro nome username

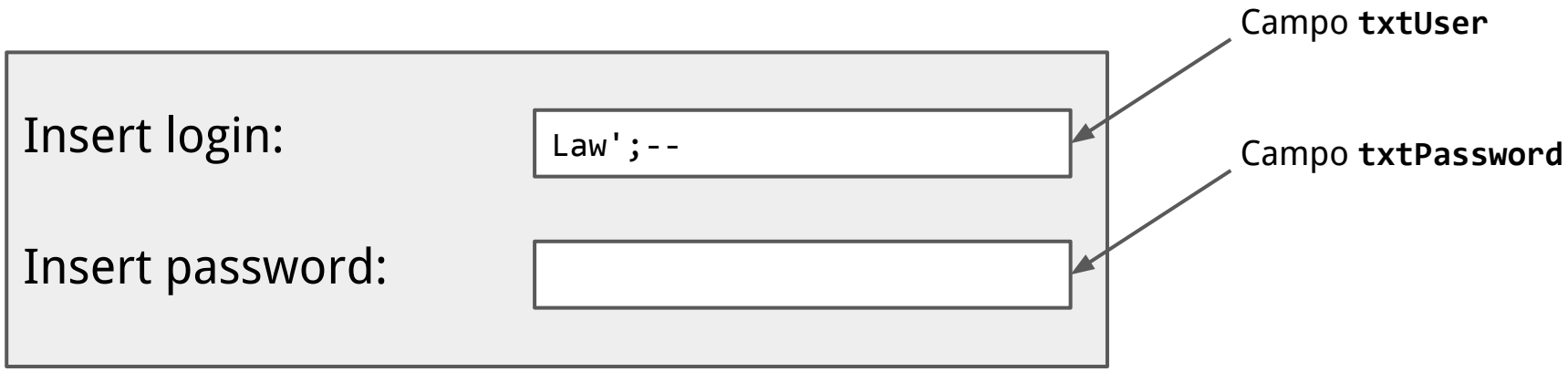
Qui finisce la nostra password

Attenzione: C'è un carattere che crea problemi alla struttura dell'interrogazione. Quale?

Concentriamoci sull'interrogazione

```
SELECT *
FROM users
WHERE username='Law' AND
password='pippo' OR '1'='1';
```

Metodi alternativi



```
SELECT * FROM users
WHERE username='Law'; -- ' AND password='';
```

Inserimenti (stesso problema)

```
INSERT INTO results
       VALUES (NULL, 'm.pogliani', '30L') -- ', '18');
```

```
INSERT INTO results VALUES (NULL, 'm.pogliani', '30L'),
(NULL, 'j.doe', '30L') -- ', '18');
```

```
INSERT INTO results VALUES (NULL, 'm.pogliani', (SELECT
password from USERS where user='admin')) -- ', '18');
```

E se volessimo leggere i dati di un'altra tabella?

```
SELECT name, phone, address FROM users WHERE
id='<user-input>';
```

UNIONs to the rescue!

- ```
  SELECT name, phone, address FROM users
  WHERE Id='' UNION ALL SELECT name, creditCardNumber,CCV2
  from CreditCardTable;-- ';
  ```

NOTE: column data types must be the same
  ```

E se non sapessimo i nomi delle altre tabelle?

Ogni database ha tabelle speciali per il tracciamento delle altre tabelle nel database:

```
SELECT schema_name FROM information_schema.schemata;

SELECT table_name FROM information_schema.tables;

SELECT column_name, data_type FROM
information_schema.columns WHERE table_name = <table>;
```

Ulteriori dettagli:
https://dev.mysql.com/doc/refman/5.7/en/information-schema.html

One Injection to rule them all (Blind)

Le **query SQL che non mostrano direttamente i valori** restituiti eseguono — o non eseguono — determinate azioni in base al risultato della query.

Possiamo sfruttarne il comportamento per **dedurre dati** che non possiamo visualizzare direttamente.

Esempio di Blind Injection

Supponiamo un login:

```
SELECT *
FROM users
WHERE username='Law'
AND password='password sbagliata'
OR username='Law'
AND password LIKE 'a%';
```

Se il login va a buon fine, la password inizia con “a”.

Prevenire le SQL-Injection

Fare la cosa giusta: **query parametrizzate**

- Prima si definisce la struttura dell'interrogazione:
  - SELECT * FROM users WHERE username=? AND password=?;
- Poi si sostituiscono le variabili con l'input dell'utente

→ Nessuna confusione tra codice e dati!

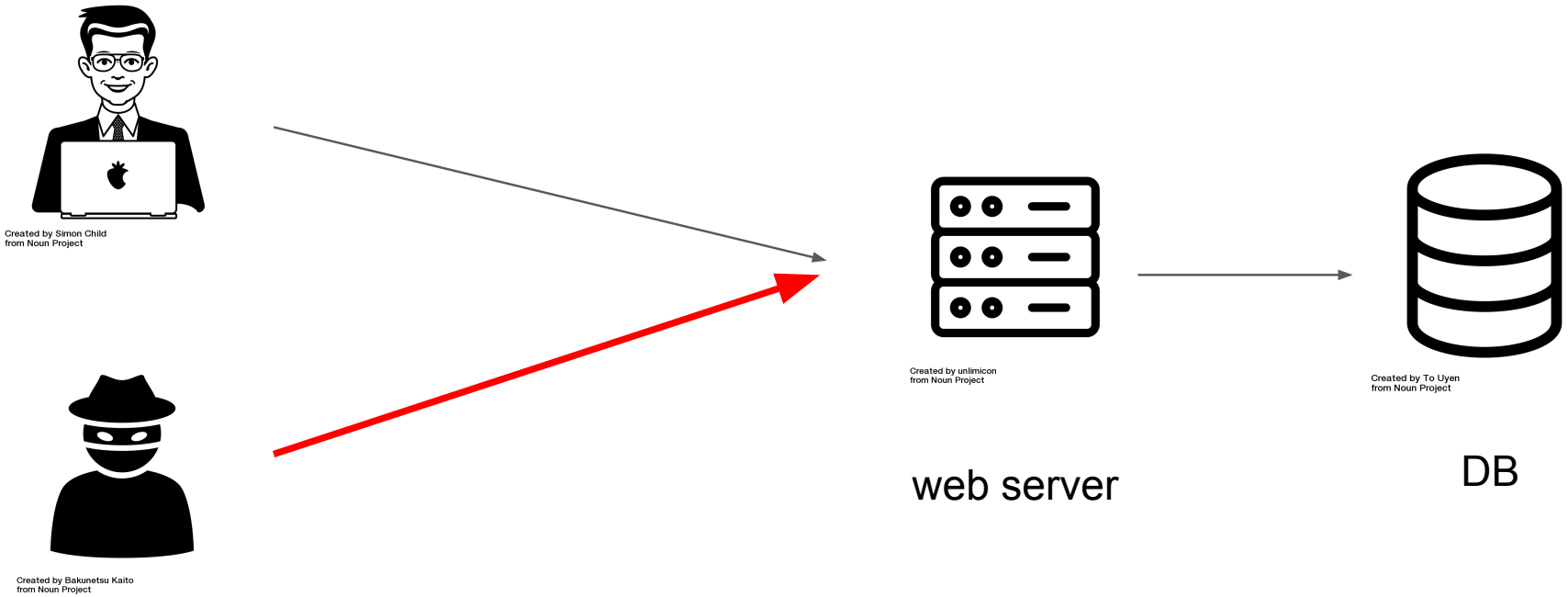
La mancanza di query parametrizzate

e la presenza di filtri o espressioni regolari personalizzate

→ **red flag** ←

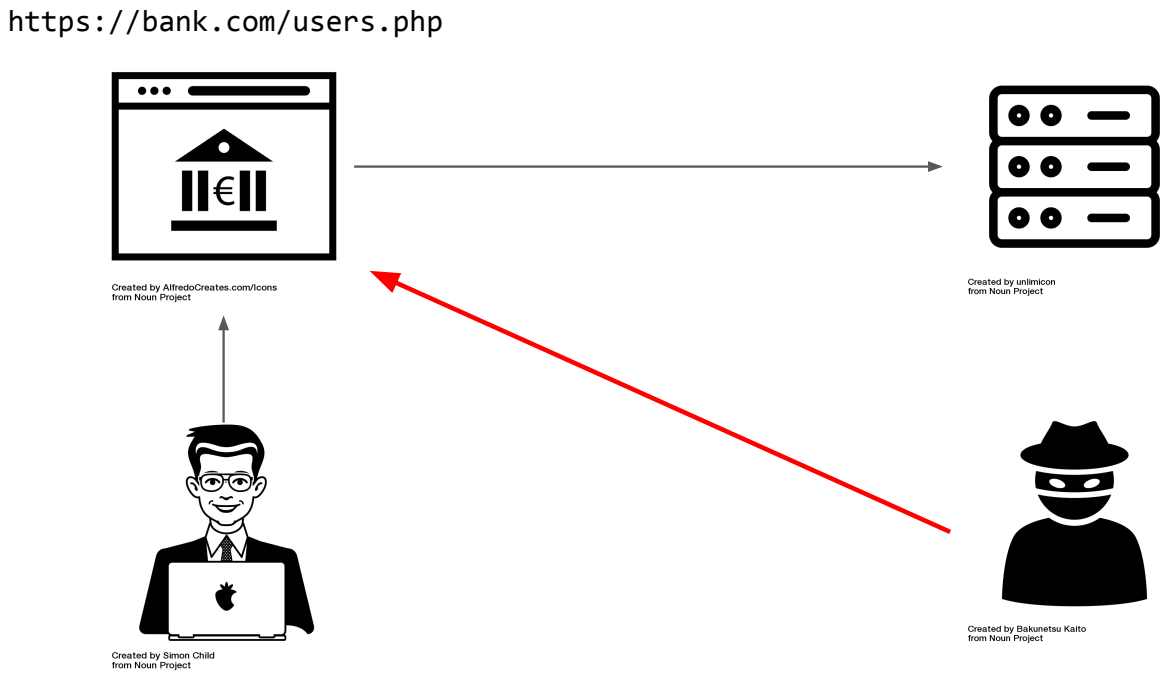
Attacchi Lato Client

Fin'ora



Goal: compromettere/attaccare direttamente il server.

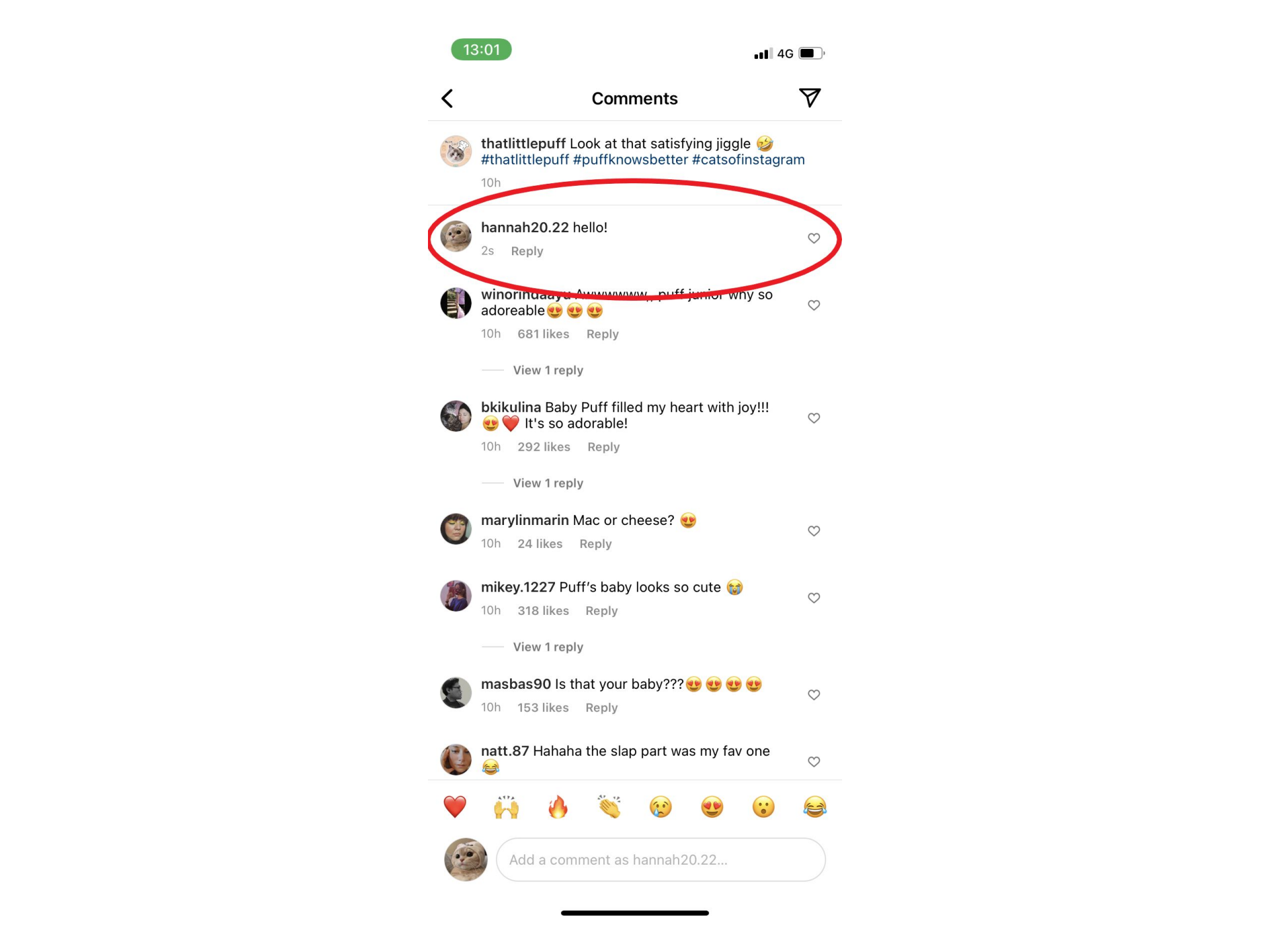
Attacchi Lato Client



Obiettivo: compromettere l'utente che usa l'applicazione web.

Il server, in genere, potrebbe avere un vulnerabilità che permette questo scenario.

XSS: Cross Site Scripting


13:01
4G
Comments
thatlittlepuff Look at that satisfying jiggle 🤣
#thatlittlepuff #puffknowsbetter #catsofinstagram
10h
hannah20.22 hello!
2s Reply
adoreable😍😍😍
10h 681 likes Reply
View 1 reply
bkikulina Baby Puff filled my heart with joy!!!
😍❤️ It's so adorable!
10h 292 likes Reply
View 1 reply
marylinmarin Mac or cheese? 😍
10h 24 likes Reply
mikey.1227 Puff's baby looks so cute 😭
10h 318 likes Reply
View 1 reply
masbas90 Is that your baby???😍😍😍😍
10h 153 likes Reply
natt.87 Hahaha the slap part was my fav one
😂
❤️ 🙌 🔥 👏 😢 😍 😮 😂
Add a comment as hannah20.22...

XSS: Cross Site Scripting

Al sito vengono mandati dati/informazioni che possono essere interpretati come codice HTML.

- Per esempio, un commento potrebbe contenere <IMG src="...”></IMG>, o <SCRIPT>... codice JavaScript…</SCRIPT

Il threat model è molto ampio. E’ possibile:

- Ottenere le informazioni confidenziali protette dal sito web (dati personali, carte di credito, …).
- Inviare messaggi ai dispositivi nella rete locale del browser che carica la pagina.
- Ottenere i cookie (non HTTPOnly o sessione).
- Aggiungere delle componenti per phishing.
- Scaricare un file sul PC.
- Ottenere controllo del computer…

XSS

Stored	Reflected	DOM-based
Payload salvato nel server (DB, disk, email, …).	Payload inviato nella richiesta e non salvato nel server, ma riportato nella risposta.	Payload legato alla pagina web (di solito viene aggiunto nell'ancora).
server-side		**client-side**

Sanitizzare l'input non è banale

Come filtrare dati non attendibili?

- Non è facile (il filtro deve essere "correttamente paranoico").

La **sequenza di validazione**:

- **Allowlisting**: consentire solo ciò che ci si aspetta.
- **Blocklisting**: in aggiunta, scartare ciò che è noto come dannoso.
- **Escaping**: trasformare i caratteri speciali in qualcos'altro di meno pericoloso.

Regola di base: l'allowlisting è più sicuro del blocklisting.

Esempio 1: Numero di Telefono

- Un campo di input che riceve un numero di telefono.
- Iniziamo progettando una allowlist:

Esempio 1: Numero di Telefono

- Un campo di input che riceve un numero di telefono.
- Iniziamo progettando una allowlist:
  - Solo numeri:
    - +39 non passa.

Esempio 1: Numero di Telefono

- Un campo di input che riceve un numero di telefono.
- Iniziamo progettando una allowlist:
  - Solo numeri:
    - +39 non passa.
  - Numeri e simboli +, -, spazio:
    - Più corretto, la maggior parte degli input verrà accettata.

Esempio 1: Numero di Telefono

- Un campo di input che riceve un numero di telefono.
- Iniziamo progettando una allowlist:
  - Solo numeri:
    - +39 non passa.
  - Numeri e simboli +, -, spazio:
    - Più corretto, la maggior parte degli input verrà accettata.
- Dobbiamo creare una blocklist per qualcosa?
  - Dipende dal back-end, ma probabilmente no.

Esempio 1: Numero di Telefono

- Un campo di input che riceve un numero di telefono.
- Iniziamo progettando una allowlist:
  - Solo numeri:
    - +39 non passa.
  - Numeri e simboli +, -, spazio:
    - Più corretto, la maggior parte degli input verrà accettata.
- Dobbiamo creare una blocklist per qualcosa?
  - Dipende dal back-end, ma probabilmente no.
- Dobbiamo fare escaping di qualcosa?
  - Allo stesso modo, probabilmente no.

Esempio 2: No Volpi

E’ un esempio giocattolo!

Il proprietario di doglover.it odia le volpi e non vuole la parola “volpe” in nessun post del sito…

Esempio 2: No Volpi

Tuttavia l'attaccante vuole ad ogni costo postare la frase:

La rapida volpe marrone salta sopra il cane pigro.

Esempio 2: No Volpi

Prima blocklist: la parola “volpe” non è consentita.

L’attaccante posta:

La rapida VOLPE marrone salta sopra il cane pigro.

Il post è valido.

Esempio 2: No Volpi

Seconda blocklist: le parole “volpe” e “VOLPE” non è consentita.

L’attaccante posta:

La rapida Volpe marrone salta sopra il cane pigro.

Il post è valido.

Esempio 2: No Volpi

Terza blocklist: Tutte le combinazioni di caratteri maiuscoli e minuscoli che formano la parola volpe non sono consentiti.

L'attaccante posta:

La rapida v0lp3 marrone salta sopra il cane pigro.

Il post è valido.

Il gatto e il topo

Questo è un “gioco del gatto e del topo” tipico della sicurezza informatica.

Fare troppo affidamento sulle blocklist può trasformarsi in un inseguimento senza fine.

Blog webapp

- Supponiamo ora di avere una semplice applicazione di blog:
  - consente a chiunque di pubblicare un commento,
  - tramite un semplice campo di testo compilato dal visitatore,
  - il cui contenuto viene poi mostrato ai visitatori successivi.
- Se non applichiamo alcun filtro a ciò che viene inserito, un attaccante potrebbe digitare:

```
<SCRIPT>
   alert('JavaScript Executed');
</SCRIPT>
```

- Un **popup** apparirebbe sullo schermo dei visitatori successivi.

Filtri XSS

- Bloccare **<SCRIPT>**!
- No, non è la soluzione.
- Prima di tutto, non è solo **<SCRIPT>**:
- ci sono anche, per es., **<FRAME>** e **<IFRAME>**, **<OBJECT>**, **<EMBED>**, …
- Ma anche in quel caso …

E questi?

```
<iframe src="javascript:alert(document.domain);">
<a href="jAvAsCrIpT:alert('JS Executed');">click here!</a>
<img src=x onerror="alert('JS!');">
<svg onload=alert('JS Executed');">
```

- Possiamo includere nella blocklist “**javascript**:”…
- Ma poi…

Spazi: un’altra risorsa

Poi qualcuno scrive:

```
<IFRAME SRC="javas

cript:alert('JavaScript Executed');">
```

E il browser rimuove i caratteri “spazi” e “a capo” ed esegue comunque il codice, mentre la nostra blocklist non lo intercetta.

Allora ampliamo la blocklist per tenerne conto...

Ma poi...

Entità HTML (Paranormali)

Cosa succede se aggiungo una entità HTML nulla (09-12) ?

```
<IFRAME SRC="javasc&#09;ript:alert('JavaScript Executed');">
```

… funziona ANCORA :(

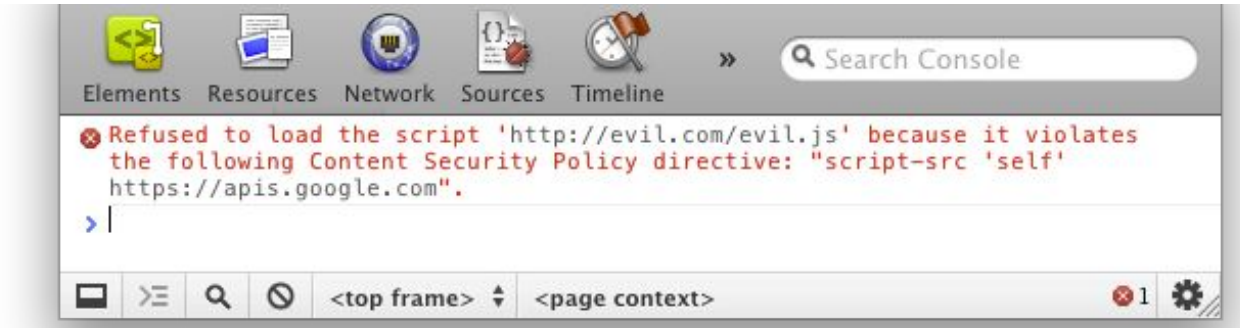
Filtro entità nulle, e poi applico i filtri precedenti…

Ma posso farlo con i valori esadecimali, posso aggiungere zeri… divertiti a scrivere quel filtro!

```
<IFRAME SRC="javasc&#X0A;ript:alert('JavaScript Executed');">
<IFRAME SRC=javasc&#000010;ript:alert('JavaScript Executed');>
```

Prevenire le XSS: Content Security Policy (CSP)

- Si informa il browser su cosa è attendibile e cosa no.
- Nello specifico, si parla di script, immagini, stili…
- Un **insieme di direttive** aggiunte agli header di ogni riposta HTTP(s):
  - `Content-Security-Policy: script-src 'self' http://google.com`
  - Ovvero, la pagina può eseguire script solo se provengono da file dello stesso dominio o da Google.
  - Non possono essere eseguiti script direttamente nel codice HTML.



Prevenire le XSS: Content Security Policy (CSP)

Perché una CSP sia efficace:

- Nessuno script inline (unsafe-inline rende la policy quasi del tutto inutile)
- Allowlist rigorosa, con origini strettamente controllate che espongono solo ciò che è necessario

Tendenzialmente è difficile!

- Scrivere le allowlist e mantenerle aggiornate
- Le Rete di distribuzione dei contenuti ospitano molti script, alcuni dei quali possono essere utilizzati per aggirare la CSP.

CSP2 e CSP3

CSP2: hash e nonce

- Metti in allowlist script specifici tramite hash
- Includi un attributo nonce <script nonce="r4nd0m"> per “marcare” uno script come “sicuro da eseguire”

CSP3: Propagazione dinamica della fiducia (strict-dynamic)

- Come propagare la fiducia (il nonce) quando uno script carica qualcos’altro, che a sua volta carica qualcos’altro, e così via…

CSRF: Cross-Site Request Forgery

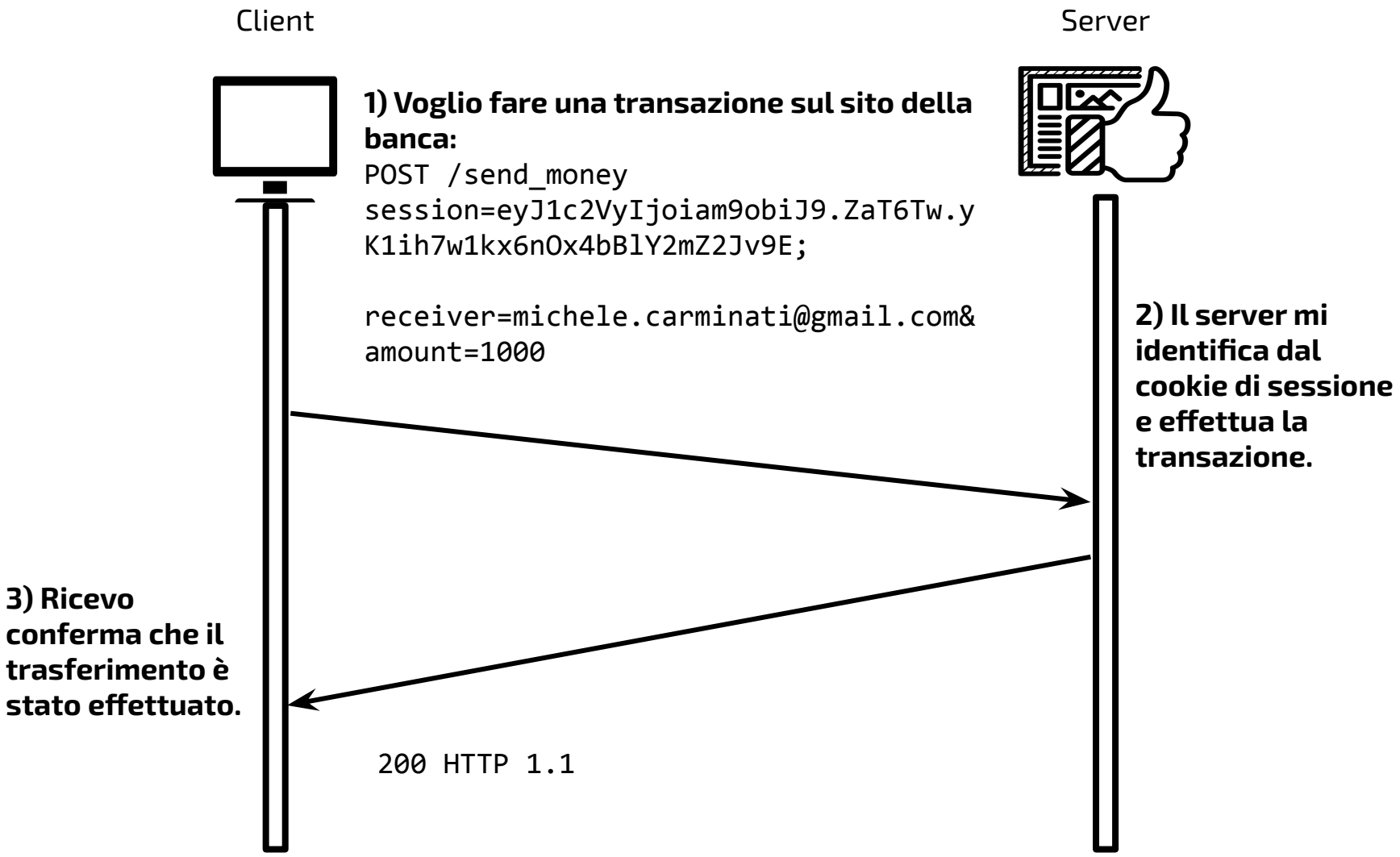
CSRF

Costringe l'utente a eseguire azioni indesiderate (azioni che modificano lo stato) su un'applicazione web in cui è attualmente autenticato con credenziali ambientali (ad esempio, tramite cookie).

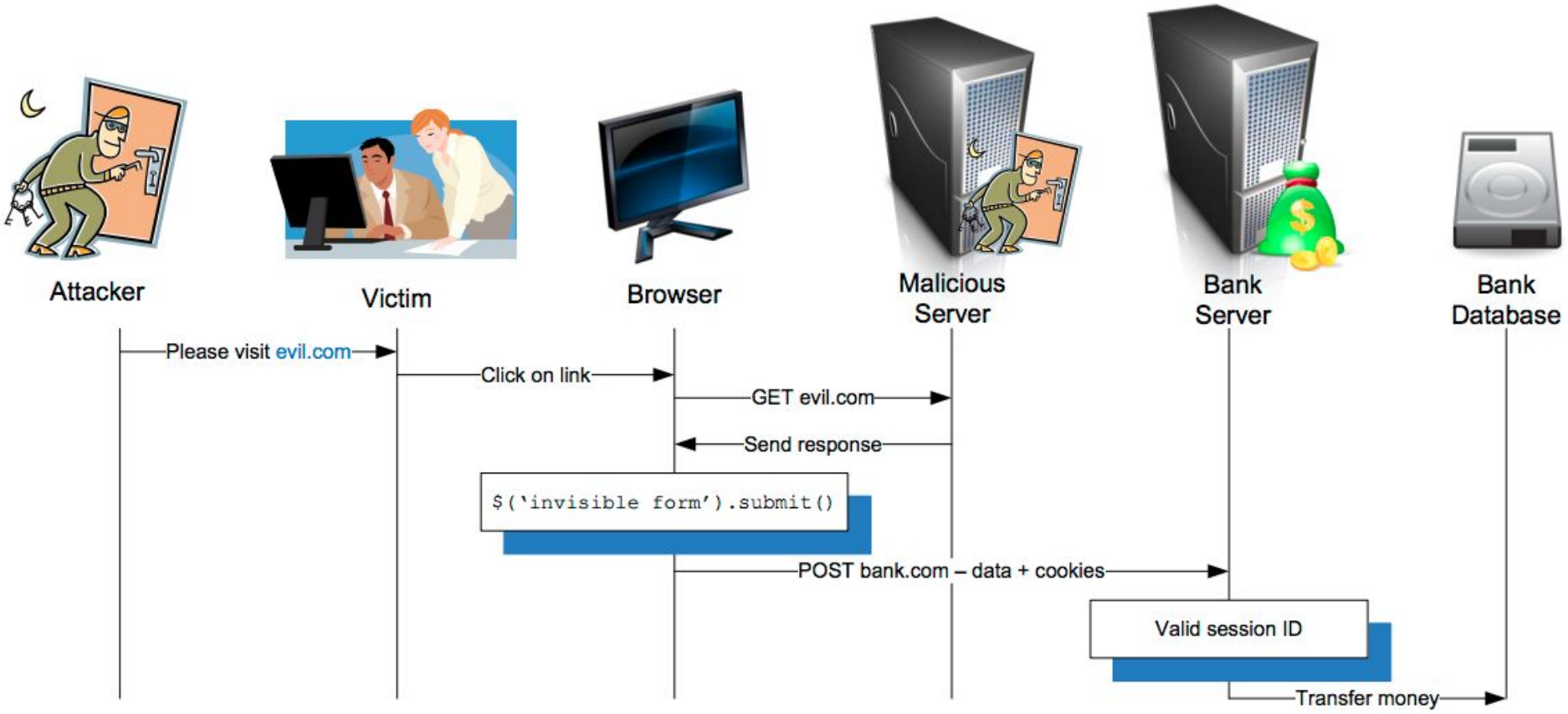
Concetto chiave:

- I cookie vengono utilizzati per la gestione della sessione.
- Tutte le richieste provenienti dal browser includono i cookie dell'utente.
- Le richieste malevole (ad esempio, link appositamente costruiti) vengono instradate verso l'applicazione web vulnerabile attraverso il browser della vittima.
- I siti web non possono distinguere se le richieste provenienti da utenti autenticati siano state generate da un'interazione esplicita dell'utente oppure no.

State Changing Actions



Attacco CSRF



CSRF Token

- Token di verifica casuale.
- Associato alla sessione dell'utente (unico).
- Rigenerato a ogni richiesta (ad esempio, incluso in ogni form che coinvolge operazioni sensibili).
- Inviato al server e poi confrontato con il token memorizzato; l'operazione lato server è consentita solo se coincide.

Non viene memorizzato nei cookie

```
<form class="form-signin" method="post">
    <h3 class="form-signin-heading">Transfer money</h3>
   [...]
    <input type="number" id="inputAmount" name="amt" class="form-control"
placeholder="Amount" required>
    <input type="hidden" value="9GiKZU6HoR" name="csrf_token">
    <button class="btn btn-lg btn-primary btn-block"
type="submit">Confirm</button>
</form>
```

Same Origin Policy (SOP)

- La Same Origin Policy è una delle regole di sicurezza fondamentali del web.
- Serve a impedire che un sito web possa leggere o manipolare dati di un altro sito senza autorizzazione.
- L'origine è definita da tre elementi:
  - PROTOCOLLO (http/https)
  - HOST (es. example.com)
  - PORTA (es. 80, 443)
- Due pagine hanno la stessa origine solo se tutti e tre questi elementi coincidono.
- **Idea**: uno script può inviare richieste ad altri siti (es, Paypal.com), ma non può leggere la risposta.
- Questo impedisce ad un attaccante di leggere il CSRF token.

SameSite Policy

Idea: non inviare affatto i cookie di sessione per le richieste che provengono da siti diversi.

I siti web specificano questo comportamento quando impostano un cookie, usando l'attributo SameSite.

SameSite=strict

- Non inviare i cookie per qualsiasi utilizzo cross-site.

SameSite=lax

- Invia i cookie solo per la navigazione tra domini (ad es. cliccare un link), ma non per form POST cross-site, immagini, frame, ecc.

Conclusioni

- I problemi principali nelle applicazioni web sono dovute da **mancata sanitizzazione dell'input**.
- Altre vulnerabilità (CSRF ed altre) sono dovute a **mancati controlli di provenienza delle richieste**.
- Per le XSS, allowlisting è spesso meglio e più sicuro di blocklisting.