

Cookie



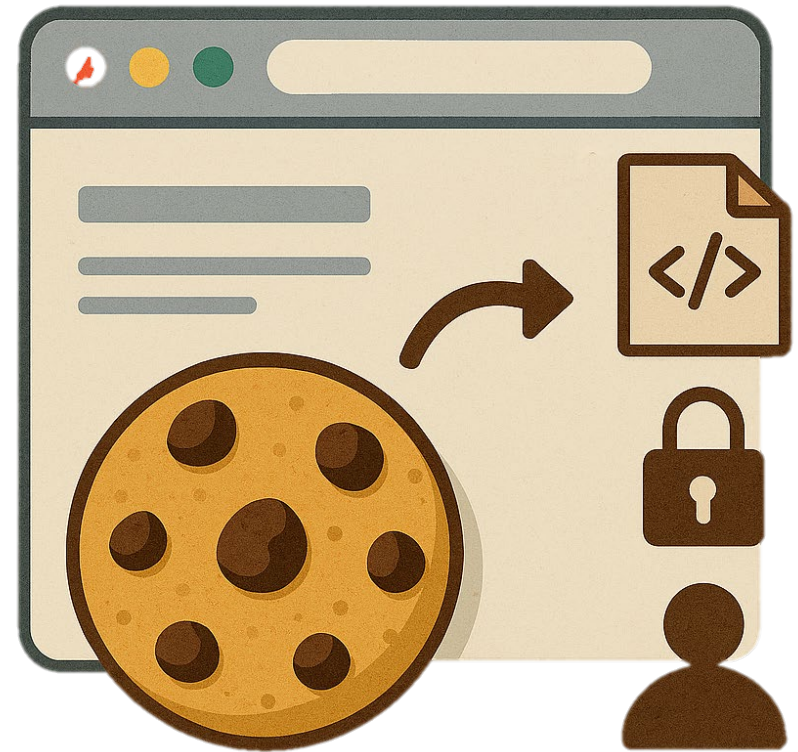
Cosa sono:

- Piccoli «file» di testo salvati nel browser
- Contengono coppie chiave=valore
- Usati per mantenere stato tra richieste HTTP
- Inviati automaticamente al server con ogni richiesta al dominio



A cosa servono:

- Gestione delle sessioni
- Autenticazione (login persistente)
- Memorizzazione preferenze utente
- Analisi e tracciamento traffico



Tipologie:

- Di sessione vs Persistenti
- Sicuri (solo su Hhttps)
- HttpOnly (non accessibili da JavaScript)
- SameSite (controllo cross-site)



Security

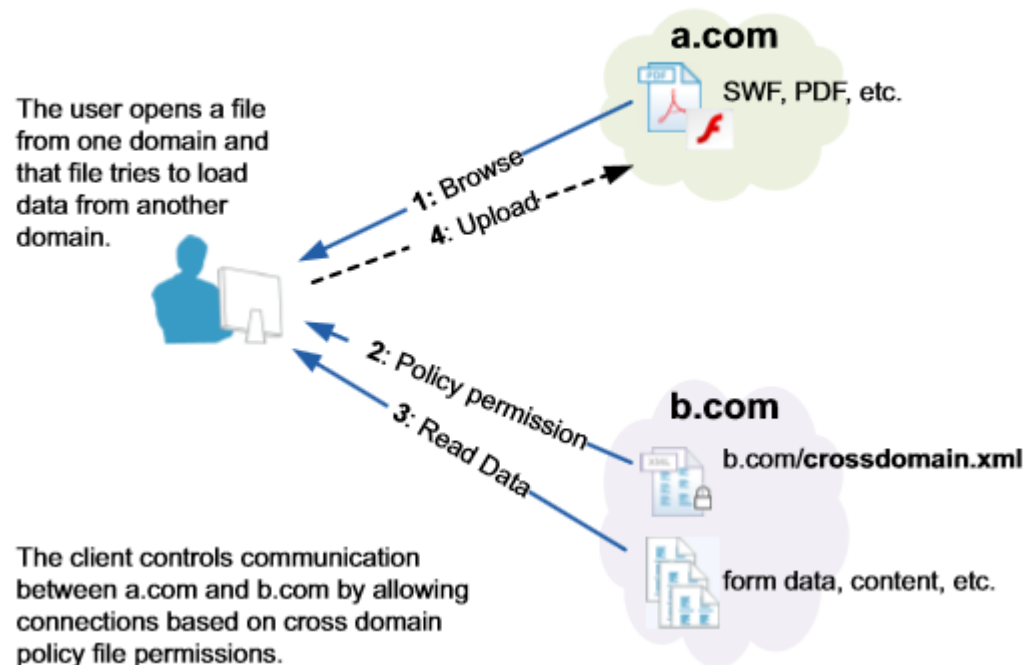
CORS



Cos'è:

CORS: Cross-Origin Resource Sharing

Figure 1 Cross domain workflow



Cos'è:

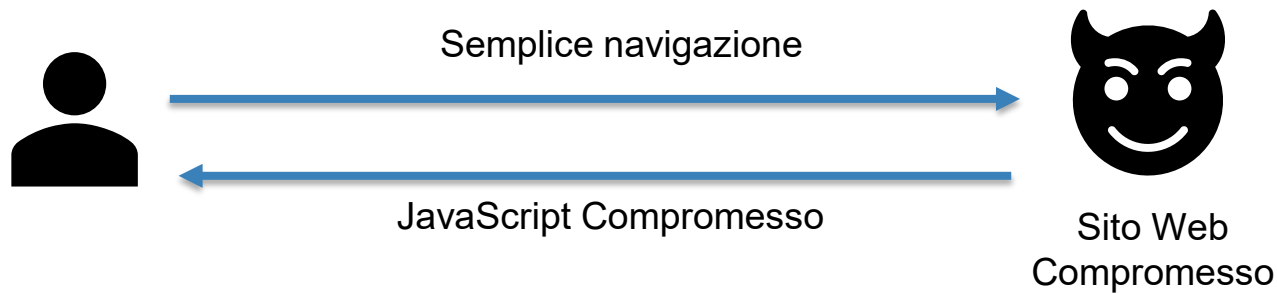
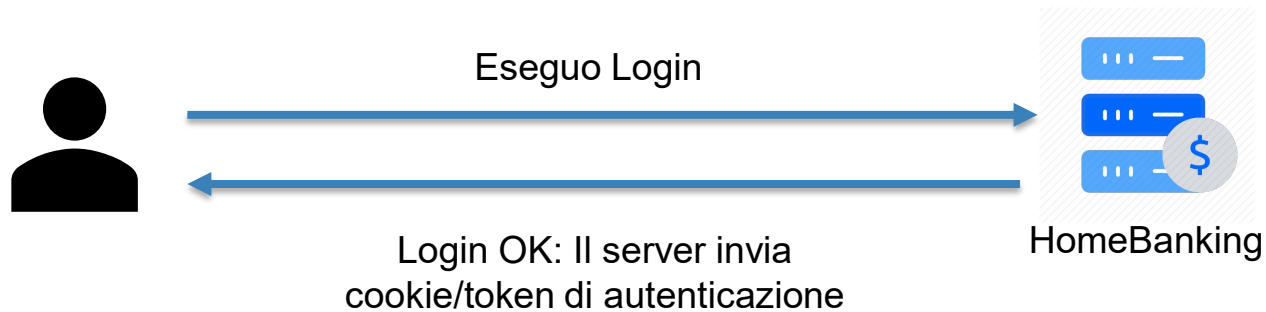
CORS: Cross-Origin Resource Sharing

Il Cross-Origin Resource Sharing (CORS) è un meccanismo che usa header HTTP aggiuntivi per indicare a un browser che un'applicazione Web in esecuzione su un'origine (dominio) dispone dell'autorizzazione per accedere alle risorse selezionate da un server di origine diversa. Un'applicazione web invia una **cross-origin HTTP request** quando richiede una risorsa che ha un'origine (protocollo, dominio e porta) differente dalla propria.

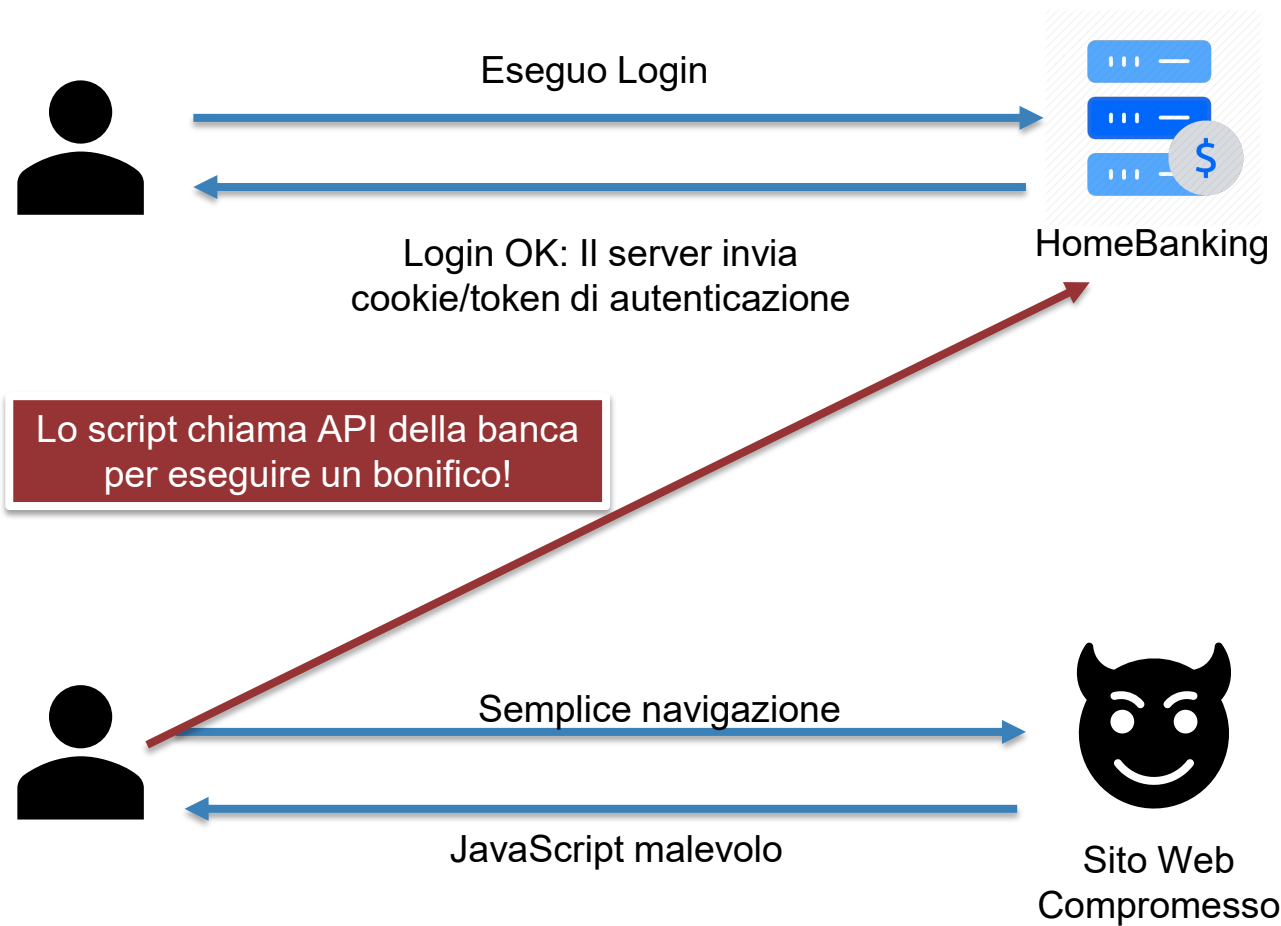
Esempio di cross-origin request: Il codice Javascript di frontend per un'applicazione web servita da `http://domain-a.com` utilizza `XMLHttpRequest` per inviare una richiesta a `http://api.domain-b.com/data.json`.

Importante: per permettere l'accesso su server di origine diversa si deve necessariamente **abilitare** e non **disabilitare** il CORS.

Perché è importante il CORS e cosa combatte? Gli attacchi CSRF (Cross site request forgery)



Perché è importante il CORS e cosa combatte? Gli attacchi CSRF (Cross site request forgery)



Rest

Tutti lo nominano ma
pochi lo conoscono



Rest

- ◎ **REPRESENTATIONAL STATE TRANSFER**
 - ◎ **È UN PARADIGMA**
 - ◎ **NON È UN PROTOCOLLO!**
-
- ◎ **NASCE GRAZIE ALLA TESI DI Roy Fielding del 2000**
 - ◎ https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf (**PAGINA 75**)

Principi del Rest

- ◎ REST is a **client-server** architecture
- ◎ REST is **stateless**
- ◎ REST is **cacheable**
- ◎ REST provides a **uniform interface** between components
- ◎ REST is a **layered system**
- ◎ REST optionally provides **code on demand**

Principi del Rest

- © Richardson Maturity Model di **Martin Fowler**
- © <https://martinfowler.com/articles/richardsonMaturityModel.html>
- © <https://blog.restcase.com/4-maturity-levels-of-rest-api-design/>

Glory of REST



Level 3: Hypermedia Controls

Level 2: HTTP Verbs

Level 1: Resources

Level 0: The Swamp of POX



Esempio di rest

Risorsa	GET read	POST create	PUT update	DELETE
<i>/books</i>	Ritorna una lista di libri	Crea un nuovo libro	Aggiorna i dati di tutti i libri	Elimina tutti i libri
<i>/books/145</i>	Ritorna uno specifico libro	metodo non consentito (405)	Aggiorna uno specifico libro	Elimina uno specifico libro

GET /books/411/authors/ Restituisce la lista degli autori del libro 411

GET /books/411/authors/1 Restituisce l'autore #1 del libro 411

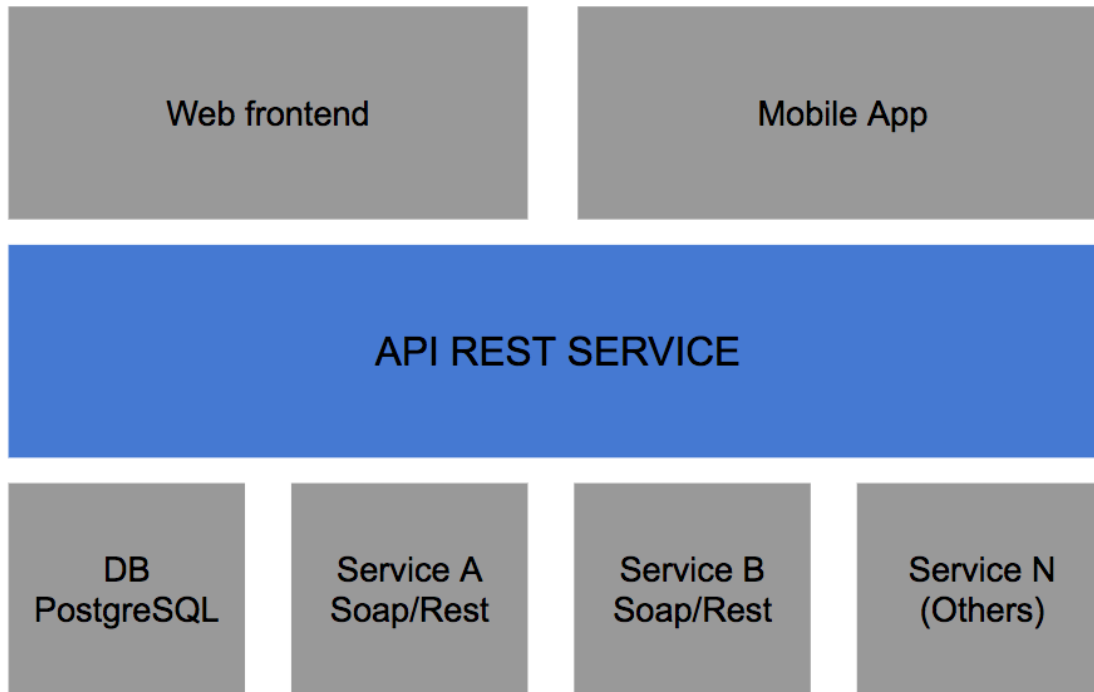
<https://developers.spreker.com/api/>

<https://developer.twitter.com/en/docs/api-reference-index>

<https://api.nasa.gov/>

Perché tutti vogliono REST?

- ◎ **Facile**
- ◎ **Comprensibile**
- ◎ **Flessibile**



NodeJs – ExpressJS – API Rest

```
var express = require('express');  
var app = express();  
  
app.get('/', function(req, res, next) {  
  next();  
})  
  
app.listen(3000);
```

HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

Callback argument to the middleware function, called "next" by convention.

HTTP response argument to the middleware function, called "res" by convention.

HTTP request argument to the middleware function, called "req" by convention.

<https://dev.to/lenmorld/quick-rest-api-with-node-and-express-in-5-minutes-336j>

Security

XSS



Cos'è:

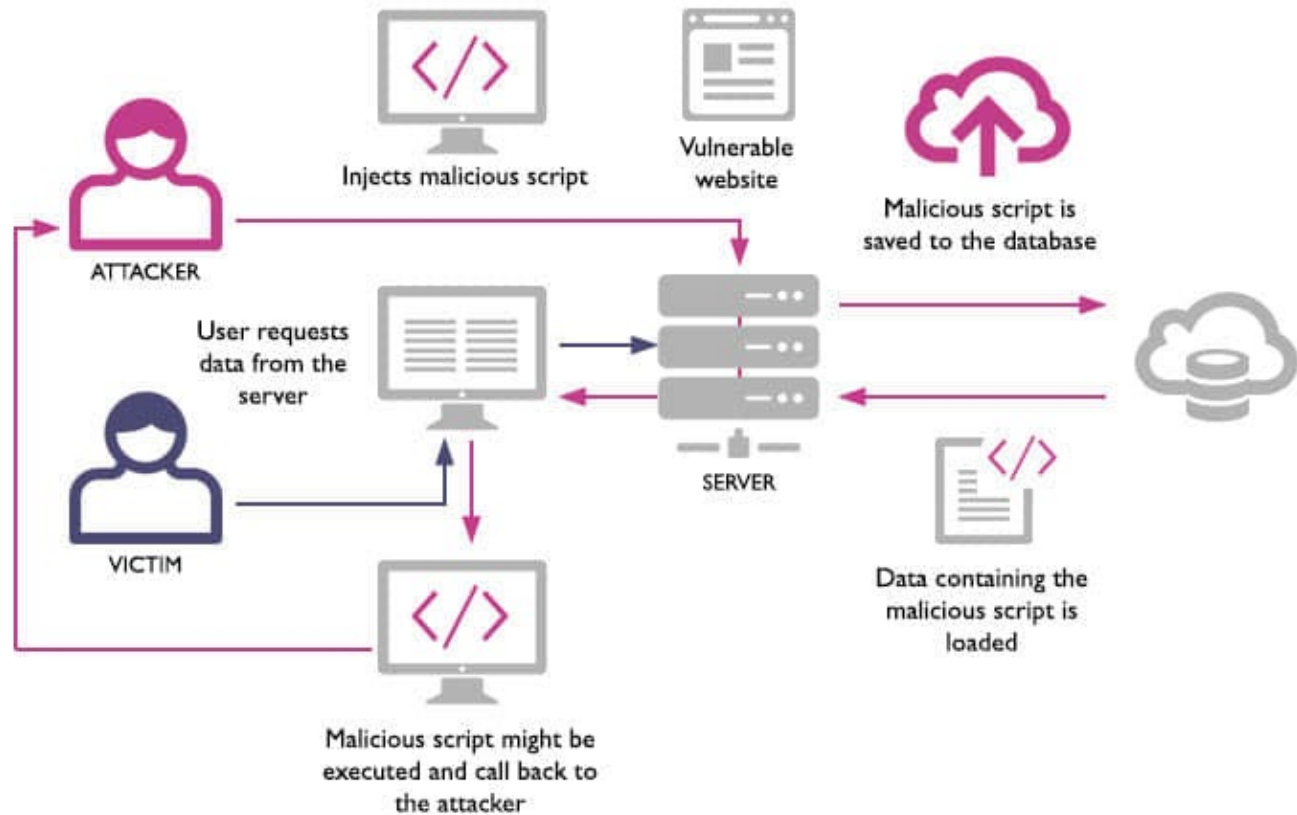
XSS: Cross-site scripting

Il **cross-site scripting (XSS)** è una [vulnerabilità](#) che affligge [siti web dinamici](#) che impiegano un insufficiente controllo dell'input nei [form](#).

https://it.wikipedia.org/wiki/Cross-site_scripting

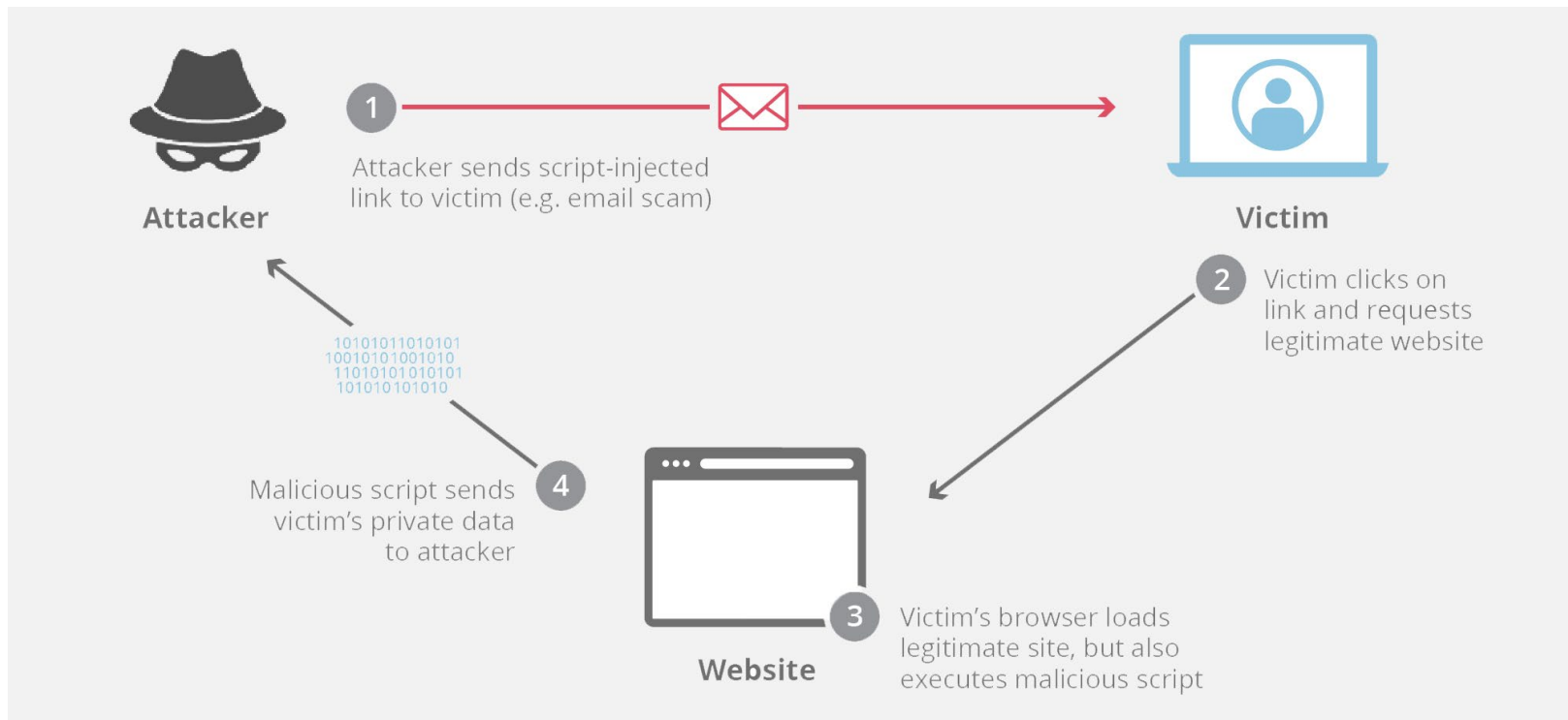
Demo:

XSS storicizzato su DB



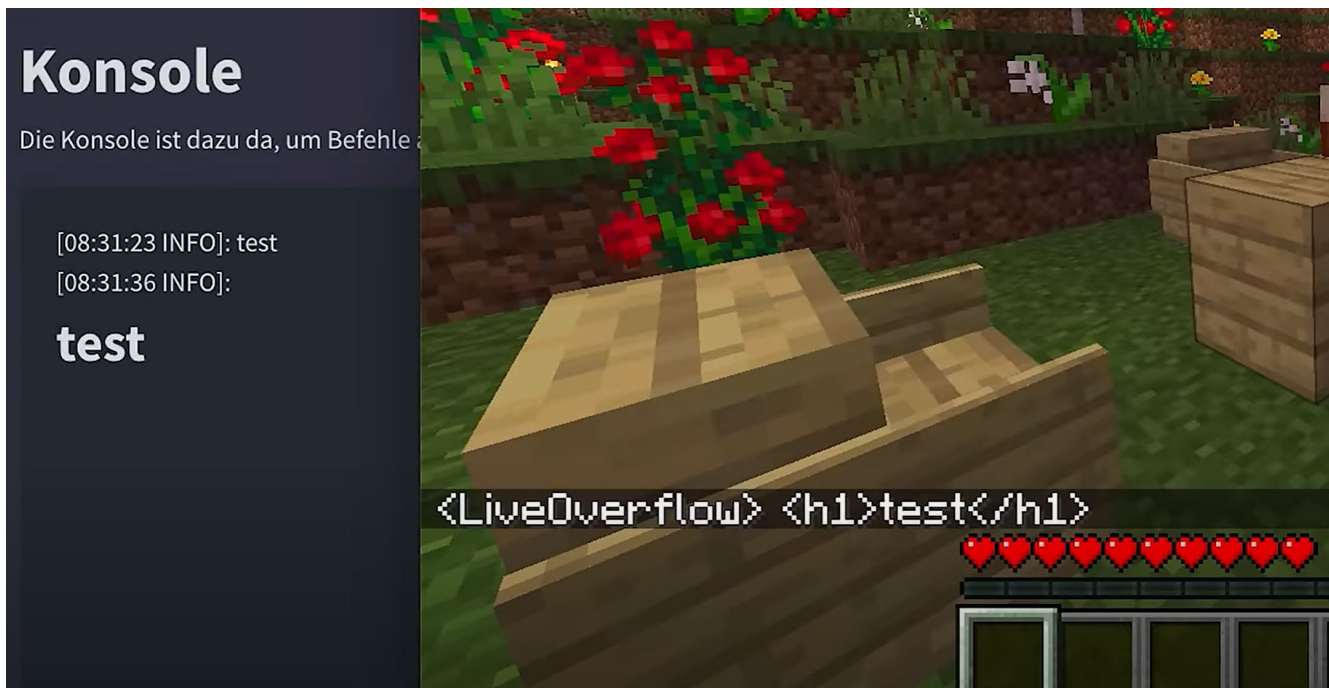
Demo:

XSS solo su client (phishing)



Demo:

XSS su minecraft



https://www.youtube.com/watch?v=3_e4t3l8fgY