Web Security 1
Il protocollo HTTP

Lorenzo LEONARDINI
Università di Pisa

https://cybersecnatlab.it

License & Disclaimer

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Obiettivi

- Conoscere il protocollo HTTP
- Comprendere il concetto di autenticazione e di sessione
- Conoscere gli strumenti per analizzare siti web

Argomenti

- Il protocollo HTTP
- Autenticazione e cookie
- TLS e HTTPS
- Altri protocolli
- Strumenti

Argomenti

- Il protocollo HTTP
- Autenticazione e cookie
- TLS e HTTPS
- Altri protocolli
- Strumenti

World Wide Web

- Il World Wide Web (WWW) è un servizio di Internet che permette lo scambio di informazioni
- Nasce nei primi anni 90, il giorno in cui Tim Berners-Lee pubblicò il primo sito web
- È composto da *siti web* che pubblicano le proprie risorse, rendendole accessibili agli utenti

World Wide Web

- Il web si basa su un'architettura client-server
- I server sono dispositivi che offrono servizi
- I client si connettono ai server per poter accedere alle funzionalità offerte

World Wide Web

- Il client web si collega al server web per accedere a una risorsa
  - Il client web si chiama browser web

URL

- Le risorse vengono identificate dagli URL (Uniform Resource Locator)
- Gli URL hanno una struttura ben definita:

protocollo://host[:porta]/risorsa

ad esempio:

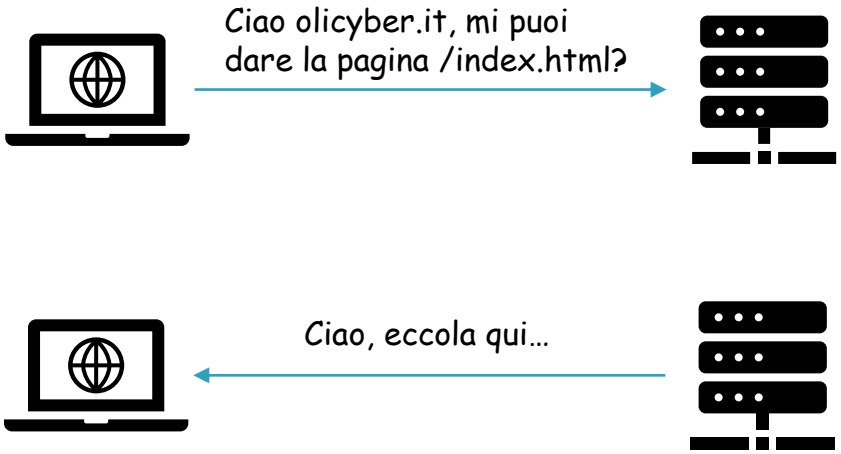
https://olicyber.it/index.html

Il protocollo HTTP

- Client e server comunicano utilizzando il protocollo HTTP
- Un protocollo definisce una serie di regole affinché due componenti possano comunicare tra loro e comprendersi

Il protocollo HTTP

- La comunicazione viene avviata dal client, che manda una richiesta al server per ottenere una risorsa
- Il server interpreta la richiesta e invia la risposta al client



Il protocollo HTTP

Richiesta

```
GET /index.html HTTP/1.1
Host: olicyber.it
Accept-Language: it-IT;q=0.9,en-us;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: Keep-Alive
```

Risposta

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: closed

<!DOCTYPE html>
...
```

La richiesta HTTP

- Request line:
  - Metodo
    - GET, POST, PUT, DELETE, HEAD, OPTIONS…
  - Path
  - Versione del protocollo:
    - HTTP/1.0, HTTP/1.1, HTTP/2.0

```
GET /index.html HTTP/1.1
Host: olicyber.it
Accept-Language: it-IT;q=0.9,en-us;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: Keep-Alive
```

La richiesta HTTP

- Headers: forniscono maggiori informazioni per gestire la richiesta correttamente

```
GET /index.html HTTP/1.1
Host: olicyber.it
Accept-Language: it-IT;q=0.9,en-us;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: Keep-Alive
```

La richiesta HTTP

- Body (opzionale): utilizzato per inviare al server dati allegati alla richiesta

```
GET /index.html HTTP/1.1
Host: olicyber.it
Accept-Language: it-IT;q=0.9,en-us;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: Keep-Alive
```

In questo caso è assente

La risposta HTTP

- Status line:
  - Versione del protocollo
  - Status code
    - 2xx: successo
    - 3xx: redirect
    - 4xx: errore del client
    - 5xx: errore del server
  - Status message

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: closed

<!DOCTYPE html>
...
```

La risposta HTTP

- Headers: forniscono maggiori informazioni riguardanti la risposta

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: closed

<!DOCTYPE html>
...
```

La risposta HTTP

- Body: il contenuto della risorsa richiesta

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: closed

<!DOCTYPE html>
...
```

I metodi HTTP più comuni

- I server web possono gestire i vari metodi come preferiscono, le scelte più comuni sono:
  - GET: per leggere una risorsa (es. ottenere una pagina web)
  - POST/PUT: per modificare una risorsa (es. inviare un form)
  - DELETE: per eliminare una risorsa
  - HEAD: per ottenere informazioni su una risorsa
  - OPTIONS: per sapere quali metodi possono essere utilizzati

Gli header più comuni

- Nelle richieste:
  - Host (obbligatorio): specifica il sito a cui è destinata una richiesta (lo stesso server potrebbe ospitare più siti)
  - User-Agent: identifica il client (es. Chrome 108 su Windows 10)
  - Authorization: autorizza la richiesta con appositi token (es. API)

Gli header più comuni

- Nelle richieste:
  - Accept[-*]: specifica il formato di dati che il client è in grado di gestire:
    - **Accept**: il *mime type* della risposta (es. text/html, application/json)
    - **Accept-Language**: la lingua preferita dal client (es. en-us, it-it)
    - **Accept-Encoding**: gli encoding supportati dal client (es. gzip)
  - Connection: specifica come gestire la connessione TCP (es. keep-alive)

Gli header più comuni

- Nelle risposte:
  - Server: fornisce informazioni sul server (es. nginx, PHP, Werkzeug)
  - Location: URL a cui il client deve effettuare il redirect
  - Content-Type: *mime type* della risorsa
  - Content-Length: dimensione in byte del body

Gli header più comuni

- Gli header sono tantissimi
- Nulla vieta di usare header custom, basta che client e server sappiano entrambi il loro significato!
- Tutti gli header standard sono documentati sulle MDN Web Docs (mozilla.org)

Le pagine web

- Il browser è un client speciale che interpreta il body della risposta e permette all'utente di interagirci, a seconda del tipo di risorsa
- Le pagine web sono realizzate utilizzando un linguaggio di markup, chiamato HTML

Le pagine web

- Le pagine possono includere altri tipi di risorse:
  - Immagini
  - Video
  - Audio
  - Fogli di stile (CSS)
  - JavaScript (JS)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Esempio</title>
    </head>
    <body>
        <h1>Un titolo</h1>
        <p style="color: red;">
            Un paragrafo rosso
        </p>
        <script>
            alert("Hello world");
        </script>
        <script src="/main.js"></script>
    </body>
</html>
```

Le pagine web

- JavaScript è un linguaggio di scripting nato per il web, adesso utilizzato anche in diverse applicazioni
- Il codice viene eseguito all'interno del browser durante la navigazione
- Permette di realizzare siti interattivi

Argomenti

- Il protocollo HTTP
- Autenticazione e cookie
- TLS e HTTPS
- Altri protocolli
- Strumenti

Autenticazione

- Spesso è necessario identificare lo specifico utente che si sta collegando a un sito web
  - Per mostrare una pagina personalizzata
  - Per fornire specifiche risorse e funzionalità solo agli utenti autorizzati
- All'utente viene richiesto di identificarsi mediante un processo di autenticazione

Autenticazione

- Dal momento in cui un utente naviga su un sito web, si dice che inizia una sessione
- La vita di una sessione è composta da tre momenti principali:
  1. Autenticazione: l'utente si identifica (facendo login ad esempio)
  2. Sessione Utente: l'utente è identificato
  3. Logout: l'utente finisce la sua sessione

Autenticazione

- Il metodo di autenticazione più diffuso fa uso della coppia username e password
- L'username indica l'identità dell'utente, la password segreta garantisce che solo l'utente legittimo sia in grado di autenticare la propria identità

Autorizzazione

- Dopo l'autenticazione vengono spesso concesse determinate autorizzazioni all'utente
- Mentre l'autenticazione identifica l'utente, l'autorizzazione definisce cosa è permesso fare all'interno del sistema
  - Es. ruoli (admin, moderatore, utente…)

Autenticazione

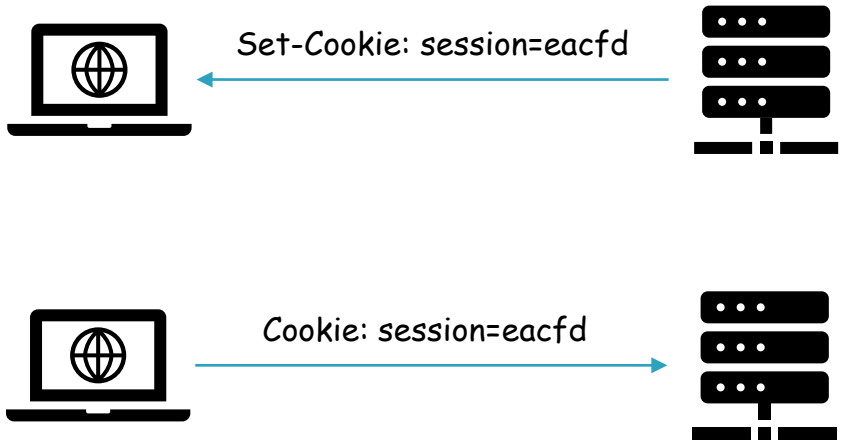
- Tuttavia, HTTP è un protocollo *stateless*: il server non ha informazioni sulle richieste precedenti a quella attuale
- Il server non può sapere che nella scorsa richiesta hai fatto un login!
- Come identificare i client?

I cookie

- Idea: il server chiede al client di mandare un identificativo allegato ad ogni richiesta
- Ogni id rappresenta un client univoco, quando il server riceve l'id sa associarlo a uno specifico utente
- Header Set-Cookie e Cookie
- Il cookie permette al server di identificare la sessione del client e prende il nome di cookie di sessione

I cookie

- I cookie sono coppie chiave-valore
- Vengono gestiti in automatico dal browser web
- Ad ogni richiesta HTTP vengono allegati tutti i cookie associati al dominio in questione



I cookie

- Quando i cookie vengono utilizzati per identificare un utente è importante che nessun attaccante ne conosca il valore
- Molti server utilizzano un id randomico ad alta entropia, in modo che non possa essere indovinato da un attaccante

I cookie

- PHP ad esempio, utilizza un token a 128 bit, generato casualmente e univoco per ogni utente che visita il sito
- Il cookie diventa di fatto una password temporanea

I cookie

- Poiché i cookie sono gestiti dal browser, è importante che il server non si fidi ciecamente del loro valore
- Un'alternativa è usare cookie firmati crittograficamente dal server
- Il client non può modificare il cookie a piacimento perché invaliderebbe la firma

I cookie

- Quando si utilizzano cookie firmati si può anche decidere di memorizzarci informazioni sull'utente
- Un esempio sono i token JWT (JSON Web Token), costituiti da un header, un body in formato JSON contenente valori arbitrari e una firma
- Le tre parti sono codificate in Base64 e separate da punti

I cookie

- Il browser mette in atto alcuni meccanismi di sicurezza per proteggere i cookie:
  - I cookie settati su un determinato dominio sono inviati solo a quel dominio
  - Un dominio non può settare cookie per altri domini

I cookie

- Per proteggere i cookie, l'header Set-Cookie permette di definire diversi parametri:
  - Durata: esprimibile sia con una data di scadenza, sia come numero di secondi per cui il cookie è valido. Il browser elimina i cookie scaduti
  - Dominio: il dominio a cui è associato il cookie
  - Path: per limitare il cookie a solo alcune path del sito

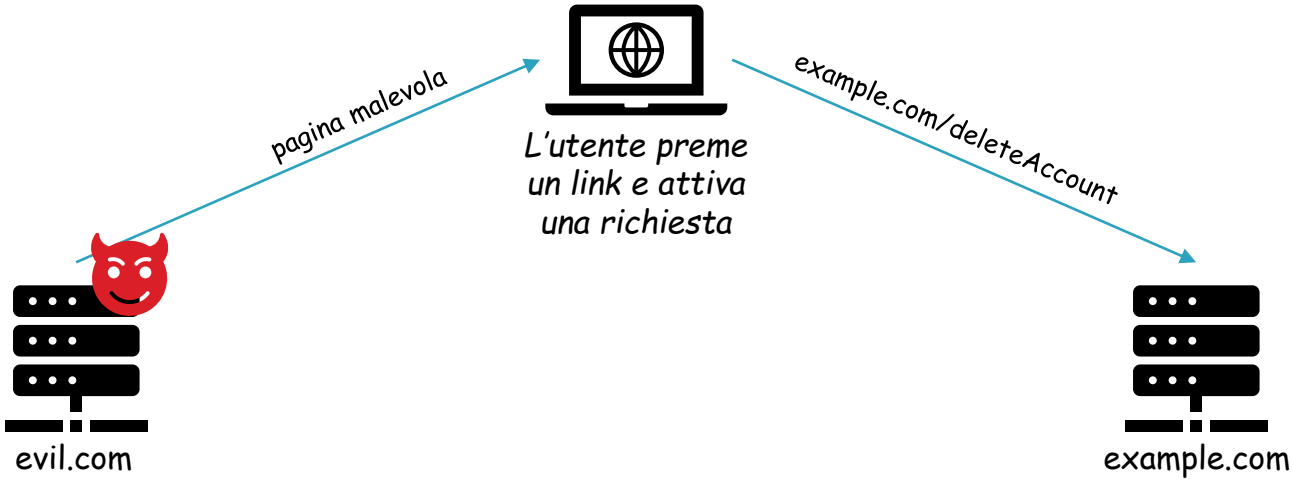
I cookie

- Per proteggere i cookie, l'header Set-Cookie permette di definire diversi parametri:
  - Secure: richiede che il cookie venga inviato solo tramite protocollo HTTPS
  - HttpOnly: fa si che il cookie non sia accessibile da JavaScript
  - SameSite: per controllare quando il cookie viene trasmesso nel caso di richieste cross-site

Cross-Site Request Forgery

- Cosa succede se il sito evil.com fa una richiesta a example.com/deleteAccount (ad esempio tramite JavaScript o con un form malevolo)?
- Il browser manda il mio cookie di sessione a example.com e il mio account viene eliminato
- Attacco chiamato Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery



Cross-Site Request Forgery

- Negli anni sono state introdotte diverse mitigazioni:
  - Token CSRF
  - Flag SameSite del cookie (richiede HTTPS, il browser deve rispettarlo!)

Cross-Site Request Forgery

- I token CSRF sono stringhe randomiche ad alta entropia generate dal server e necessarie per validare le richieste
- Il token è trasparente all'utente legittimo
- Siti esterni non sono in grado di ottenere il token e non possono includerlo in richieste malevole

Cross-Site Request Forgery

- Primo approccio:
  - Il server inserisce il token in un campo nascosto del form
  - Quando l'utente invia il form, il token viene trasmesso assieme ai campi definiti dall'utente
  - Il server può controllare la validità del token
  - Poiché il token viene inserito nella pagina ed è sempre diverso, siti malevoli non possono conoscerne il valore

Cross-Site Request Forgery

- Secondo approccio:
  - Il server invia il token al client in un cookie extra
  - Quando l'utente invia il form, il cookie viene trasmesso nella richiesta
  - Il server può controllare la validità del token
  - Poiché il cookie ha uso singolo, deve essere impostato visitando la pagina legittima e siti malevoli non possono conoscerne il valore

Cross-Site Request Forgery

- In tutti gli approcci i token CSRF sono legati alla specifica sessione utente del client web
- Client diversi hanno token diversi
- I token dovrebbero essere utilizzati una volta sola
- Il server deve rifiutare tutte le richieste che non contengono token CSRF

Argomenti

- Il protocollo HTTP
- Autenticazione e cookie
- TLS e HTTPS
- Altri protocolli
- Strumenti

TLS e HTTPS

- Il protocollo HTTP trasmette i dati in chiaro, senza alcun tipo di cifratura
- Un attore malevolo in grado di intercettare il traffico sarebbe in grado di leggere password, cookie e dati sensibili
- HTTP non è in grado di garantire l'identità dei server a cui il client si connette

TLS e HTTPS

- TLS è un protocollo crittografico progettato per offrire comunicazioni sicure
- Cifra i messaggi tra client e server
- Offre meccanismi di autenticazione mediante l'utilizzo di certificati

TLS e HTTPS

- HTTPS (Hypertext Transfer Protocol Secure) è un'estensione di HTTP che utilizza TLS per proteggere la comunicazione
- Client e server inizializzano una sessione TLS
- All'interno della sessione TLS si utilizza HTTP in maniera standard

TLS e HTTPS

- I server si presentano ai browser fornendo un certificato crittografico
- Il certificato è firmato da una Certificate Authority (CA) di cui il browser si fida, e che garantisce l'autenticità del server
- I certificati delle Certificate Authority sono gestiti dal sistema operativo o installati dal browser

TLS e HTTPS

- In realtà vi è una catena di certificati, in modo da facilitarne la gestione e la revoca

Argomenti

- Il protocollo HTTP
- Autenticazione e cookie
- TLS e HTTPS
- Altri protocolli
- Strumenti

Protocolli web

- Internet non è costituito dal solo protocollo HTTP:
  - FTP (File Transfer Protocol) per il trasferimento di file tra client e server
  - WS (WebSocket) per offrire comunicazione bidirezionale tramite una connessione TCP
  - …

WebSocket

- WebSocket è progettato per essere compatibile con HTTP
- Utilizza HTTP per effettuare un handshake e utilizza l'header Upgrade per passare al protocollo WS
- Facilita lo scambio di dati in tempo reale tramite una connessione che viene mantenuta sempre aperta

Argomenti

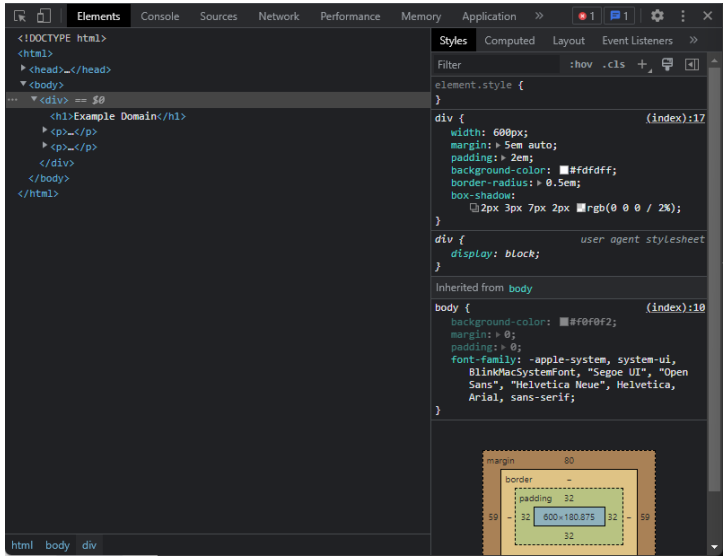
- Il protocollo HTTP
- Autenticazione e cookie
- TLS e HTTPS
- Altri protocolli
- Strumenti

Strumenti

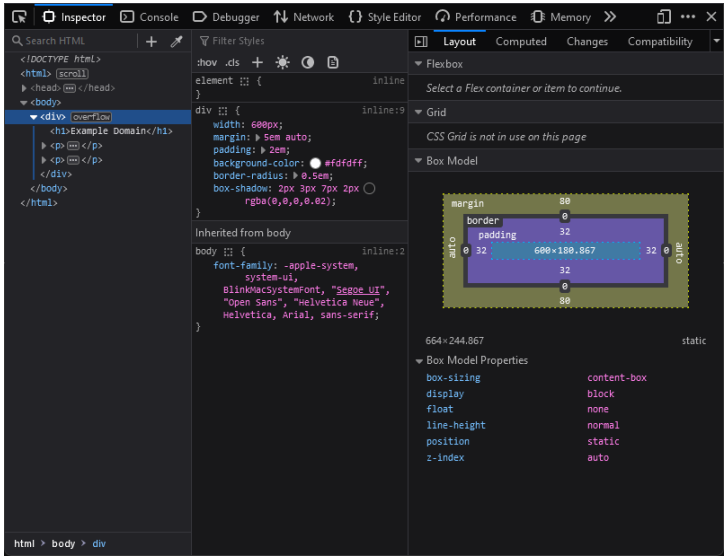
- Uno strumento base molto utile per analizzare e interagire con siti web è lo strumento per sviluppatori offerto dai browser
- Su Chrome e Firefox si può aprire con la combinazione di tasti CTRL+SHIFT+I o con il tasto F12

Interagire con la pagina

- Chrome

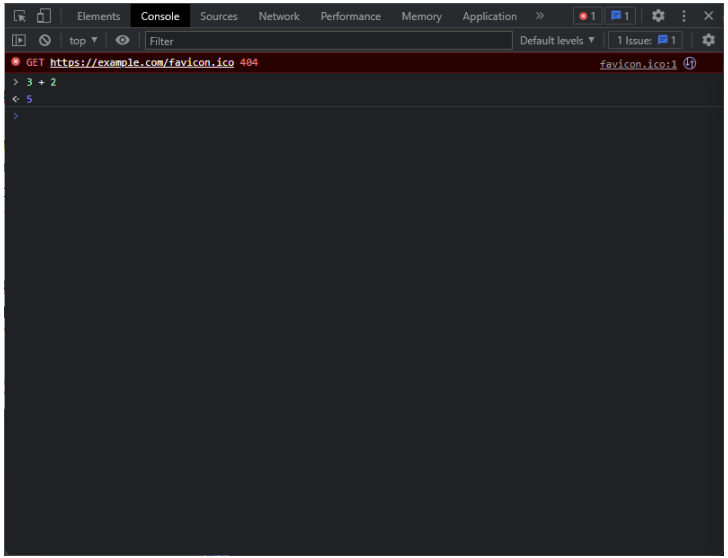


- Firefox

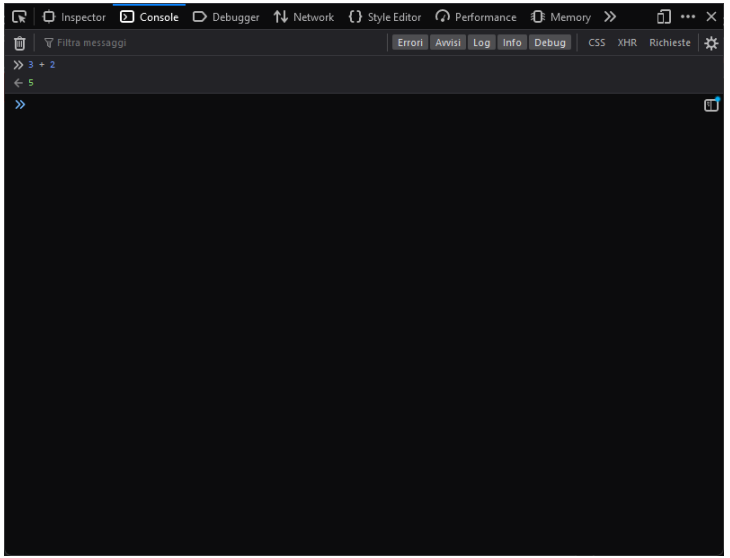


Console JavaScript

- Chrome

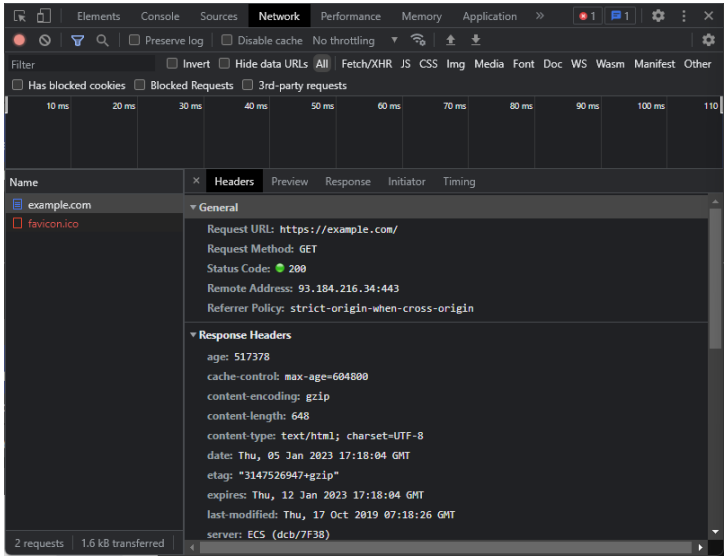


- Firefox

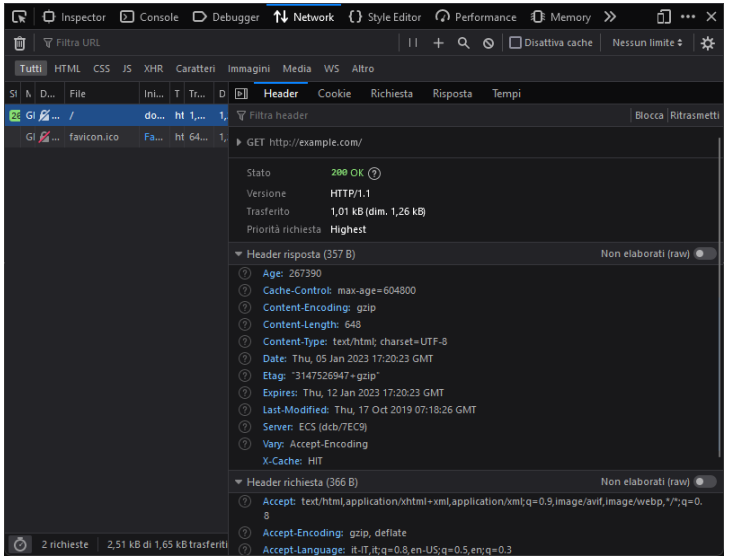


Analizzare il traffico

- Chrome

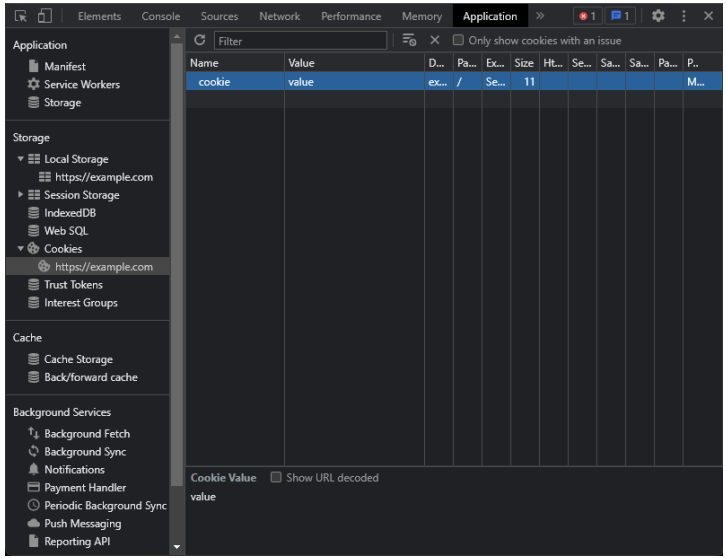


- Firefox

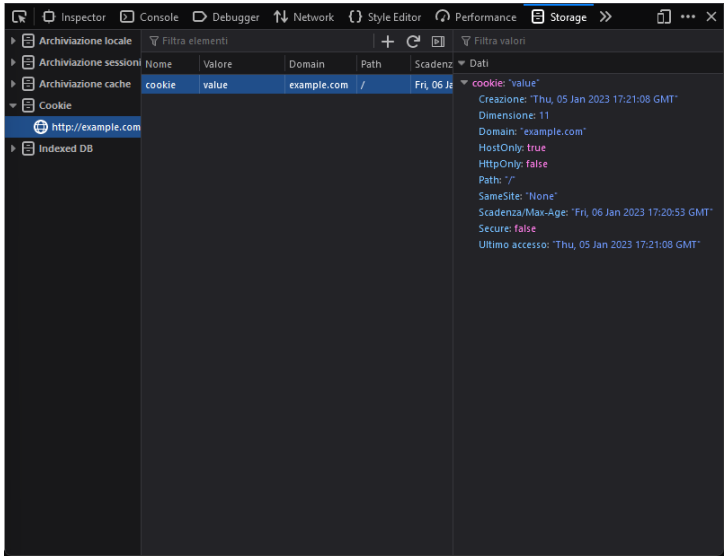


Visualizzare e modificare i cookie

- Chrome



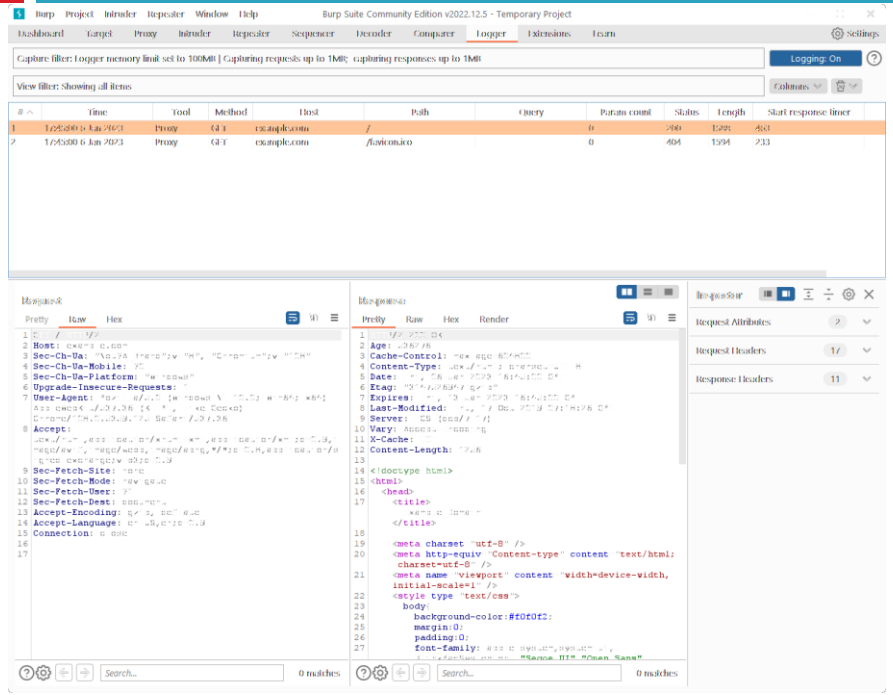
- Firefox

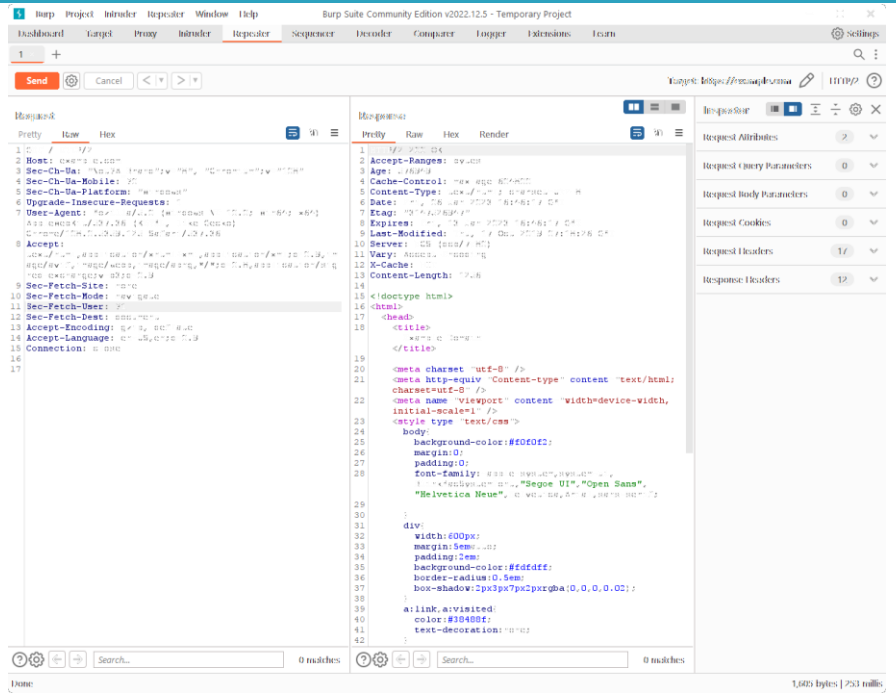


Burp

- Uno strumento più avanzato per l'analisi dei siti web è Burp Suite
- Disponibile gratuitamente la versione Community con alcune limitazioni
- Strumenti di analisi passiva e attiva
  - Proxy
  - Repeater
  - Intruder

Burp





Web Security 1
Il protocollo HTTP

Lorenzo LEONARDINI
Università di Pisa

https://cybersecnatlab.it