Filtri e sicurezza

Corso di Laboratorio di Informatica

Prof. Dei Rossi, Leonardo Essam

Fidarsi è bene, ma non fidarsi è meglio! (1)

Abbiamo visto come ottenere dei dati in input dall'utente.

Solitamente, la maggior parte degli utenti userà la nostra applicazione come si deve e non cercherà di trovare delle falle di sicurezza nella nostra applicazione... :)

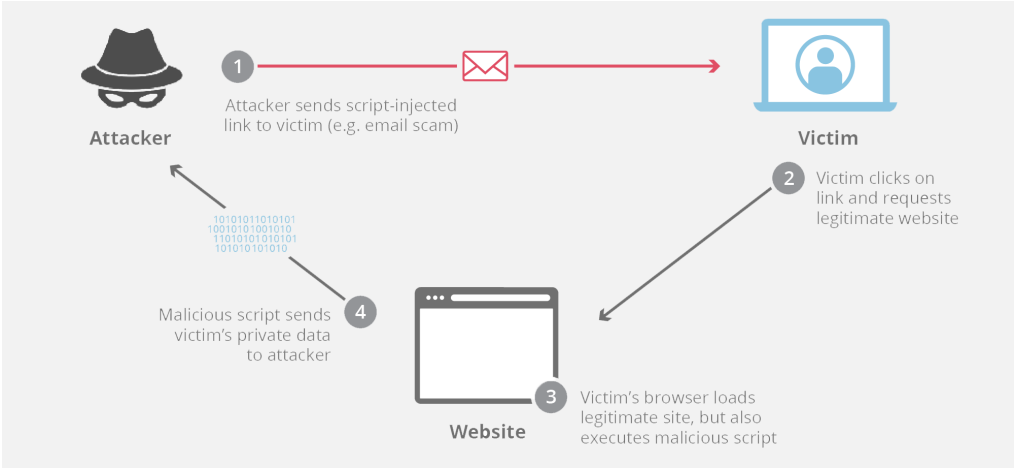
... le ultime parole famose ...

Fidarsi è bene, ma non fidarsi è meglio! (2)

La validazione degli input è una operazione fondamentale per garantire la sicurezza della propria Web App!

Uno degli attacchi più famosi è sicuramente il Cross-site scripting (XSS), questo tipo di attacco non riguarda nello specifico il server Web su cui viene eseguita l'applicazione ma riguarda l'iniettare del codice JavaScript malevolo all'interno di pagine Web visualizzate da altri utenti.

Cross-site scripting (XSS)



Validazione della form (1)

In PHP quando si lavora con i dati inviati dagli utenti è cosa *buona e giusta* verificare la "bontà" dei dati ricevuti.

In PHP esistono diversi metodi per svolgere vari tipi di controlli.

Validazione della form (2)

Un primo controllo, per evitare errori nella logica del codice, è controllare che effettivamente vengano compilati i campi della form.

Sappiamo che in HTML è possibile mettere il tag `required`, ma esso può essere aggiornato con la tecnica più vecchia del mondo "tasto destro > ispeziona".

In PHP, si può verificare l'esistenza di un campo della form con:

```
$esiste_variabile = key_exists("<chiave>", $array);
```

La funzione `key_exists([...])` prende come primo parametro una chiave e come secondo l'array da controllare. Questo è utile con `$_POST` perché è un array chiave-valore!

Sicurezza dei dati (1)

Una volta verificata l'effettiva presenza dei dati richiesti dallo script, è necessario poi assicurarsi che quei dati non siano "malevoli".

La funzione `htmlspecialchars()` in PHP serve a **convertire caratteri speciali in entità HTML**, così che un testo venga mostrato nel browser esattamente com'è, senza essere interpretato come HTML o JavaScript.

È una funzione fondamentale per prevenire **XSS (Cross-Site Scripting)** e per stampare in sicurezza dati provenienti dall'utente.

Sicurezza dei dati (2)

Sintassi della funzione:

```
htmlspecialchars(
    string $string,
    int $flags = ENT_COMPAT,
    ?string $encoding = null,
    bool $double_encode = true
): string
```

Sicurezza dei dati (3)

Alcune traduzioni:

- & → &
- " → "
- ’ → ' (se abiliti `ENT_QUOTES`)
- < → <
- > → >

Sicurezza dei dati (4)

Esempio:

```
$input = '<script>alert("ciao")</script>';
echo(htmlspecialchars($input));
```

Risultato:

```
&lt;script&gt;alert(&quot;ciao&quot;)&lt;/script&gt;
```

In questo modo la stringa non verrà interpretata da JavaScript!

Sicurezza dei dati (5)

Per completezza, la funzione inversa di `htmlspecialchars` è:

```
$input = '&lt;script&gt;alert(&quot;ciao&quot;) [...]';
echo(htmlspecialchars_decode($input));
```

Risultato:

```
<script>alert("ciao")</script>
```

Per approfondire

- **PHP Form Validation**
  W3Schools - PHP Tutorial (link)
- **PHP Form Required**
  W3Schools - PHP Tutorial (link)
- **Cos'è il cross-site scripting?**
  CloudFlare - Learning > Security (link)

Buonarroti