Cunning Injections

WORKSHOP ON SECURITY FRAMEWORKS 2019

Sergio Esposito

Agenda

Nel corso dell'incontro sfrutteremo vulnerabilità esistenti su tool largamente diffusi.

- **CVE-2019-12922**, CSRF su PhpMyAdmin 4.9.0.1
- **CVE-2019-9737**, DOM-based XSS su Editor-md 1.5.0
- **CVE-2019-16728**, mXSS su DOMPurify 2.0.0

Tali vulnerabilità verranno utilizzate in locale a scopo dimostrativo.

Don't try this ~~at home~~ in the wild!

CSRF (Cross Site Request Forgery)

Consiste nel fare effettuare all'utente, attraverso un sito malevolo X (o un sito benigno al cui interno viene iniettato il payload malevolo), azioni su un altro sito Y nel quale l'utente è **già autenticato**. Ciò in genere avviene senza che l'utente ne abbia contezza.

L'attacco che vedremo oggi colpisce **PhpMyAdmin**, versioni **4.9.0.1** e inferiori, e permette di **cancellare uno (o più) server** gestiti attraverso la piattaforma.

PhpMyAdmin

Attraverso tale piattaforma è possibile effettuare varie **operazioni di amministrazione** su ogni server che viene aggiunto attraverso l'apposito pannello. Tali operazioni comprendono:

- Creazione, gestione e manutenzione di **database**
- Possibilità di effettuare **query** sugli stessi database
- Creazione e gestione degli **utenti** che possono accedere a tali database
- Creazione e gestione dei relativi **backup**

XSS (Cross Site Scripting)

Consiste nell'**iniezione di codice Javascript** indesiderato all'interno di una pagina web. Tale attacco potrebbe richiedere azioni da parte dell'utente (come ad esempio cliccare un link), ma ne esistono varianti (come lo *Stored XSS*) che non ne richiedono alcuna, e che potenzialmente passano anche inosservate agli occhi della vittima.

L'attacco che vedremo oggi colpisce **Editor-md**, versioni **1.5.0** e inferiori, e permette di **effettuare un DOM-Based XSS** attraverso la piattaforma.

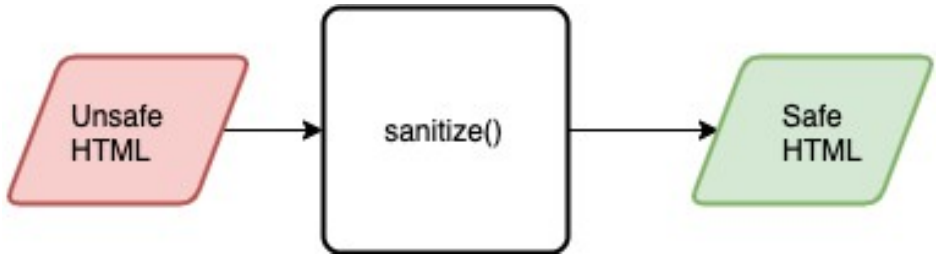
Editor-md

Si tratta di un **editor markdown** molto utilizzato, specialmente da aziende orientali. Come avviene sugli altri editor, è possibile scrivere del codice markdown, insieme a del codice HTML se lo si desidera, e vedere il risultato **renderizzato in tempo reale** su un apposito riquadro.

Gli editor sono **spesso soggetti ad attacchi**. Per evitare la scrittura di script malevoli da inviare ad altri utilizzatori dell'editor o, peggio, per la scrittura post malevoli che verranno pubblicati su un forum, è possibile mettere in **blacklist** alcuni tag (come ad esempio il tag *<script>*) perché non vengano renderizzati.

Com'è possibile immaginare, questo non è sufficiente per proteggersi da attacchi.

mXSS (Mutation-based XSS)



Per far fronte ai vari injection attack, oltre a **sanitizzare gli input** e seguire delle **linee guida di sviluppo sicuro**, può rivelarsi estremamente efficace utilizzare apposite librerie di protezione o tool che sanitizzano automaticamente gli input (AntiSamy, jSoup…).

Nel 2013, *Mario Heiderich et al* aggirano le protezioni offerte da tali strumenti tramite l'utilizzo di **payload innocui** poiché malformati. Tali payload infatti, nel loro stato originario, non generano alcun attacco: venendo tuttavia 'sanitizzati' dalle librerie e interpretati poi dal browser, tali payload si **trasformano in malevoli** ed innescano l'attacco, bypassando effettivamente le misure di protezione imposte. Tale attacco viene chiamato dai ricercatori **mXSS**.

DOMPurify

Qualche tempo dopo aver inventato mXSS, Mario Heiderich et al sviluppano **DOMPurify**, una libreria di sanitizzazione contro **DOM-Based XSS**. Chiamando la funzione *DOMPurify.sanitize(input)* è infatti possibile sanitizzare il contenuto della variabile *input*, affinché sia sicuro renderizzare eventuale codice HTML al suo interno.

L'ultimo attacco mostra come sia possibile bypassare **DOMPurify**, versioni **2.0.0** e precedenti, tramite l'utilizzo di specifici payload malformati.

Shall we dance?