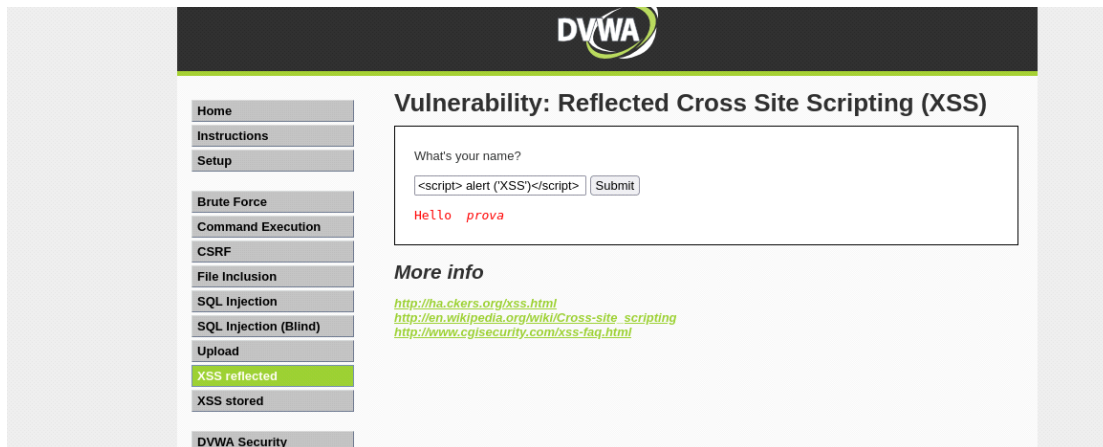
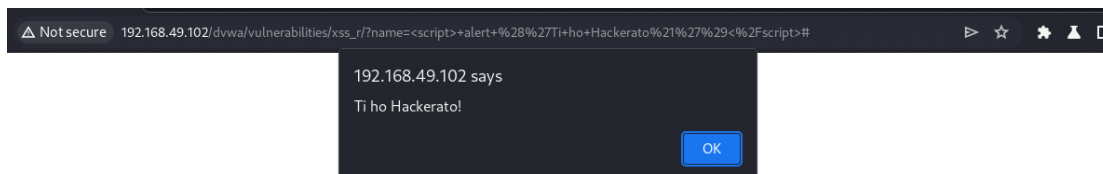


Report XSS Reflected



Per effettuare un attacco di tipo XSS reflected, dobbiamo prima di tutto assicurarci che il sito "vittima" sia vulnerabile a questo tipo di attacco, andando a inserire nella barra un semplice comando html per comprendere se lo stampa a video o se lo esegue, quindi utilizziamo `<i> prova`, se c'è la vulnerabilità, quello che vedremo nel punto di riflesso sarà *prova* scritto in corsivo. Una volta che ci siamo accertati di ciò, possiamo procedere all'attacco e, nella barra andiamo a inserire un piccolo programma in html che ci farà comparire a video un pop up, come nella foto sottostante:

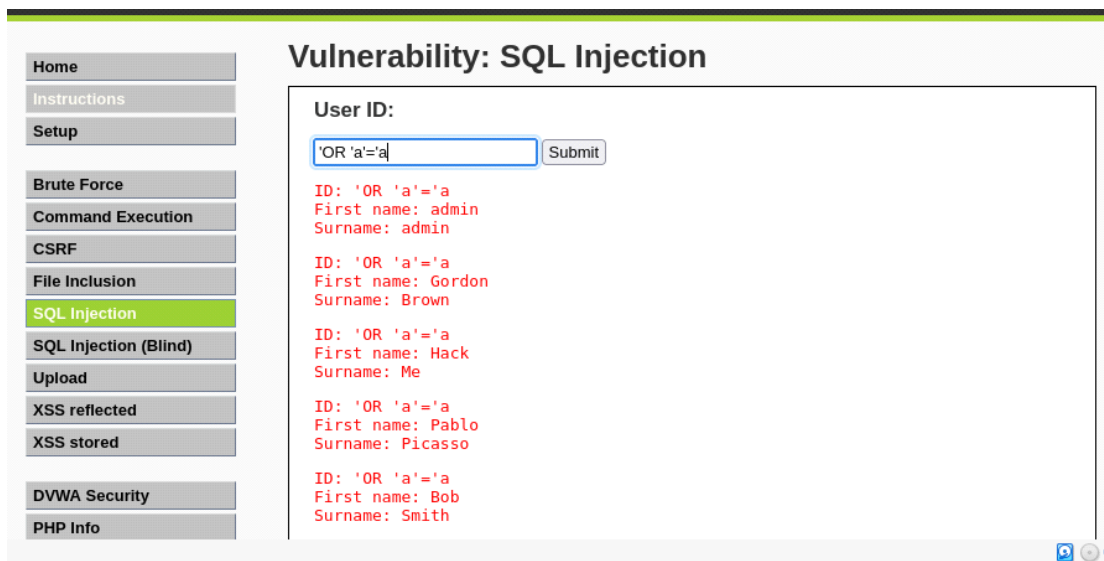


A questo punto a un attaccante non resta che spammare il link malevolo ad alcune persone (un po' come funziona il phishing) e aspettare che qualcuno cada nella trappola premendo il link e avviando così il malware.

Report Attacco SQL Injection

Questa tipologia di attacco permette di accedere ai dati contenuti all'interno di un database di una webapp, sfruttando una vulnerabilità molto presente ancora oggi. Ma come funziona?

Il funzionamento di questa vulnerabilità è difficile da spiegare ma semplice da applicare perché basta andare a scrivere un comando all'interno della barra di ricerca delle webapp. La spiegazione è che i programmi scritti in php per i database, permettono di andare a visualizzare i dati al loro interno tramite un ID che va a inserire proprio l'utente per vedere i suoi dati all'interno del database; Il problema è che questo meccanismo di difesa è stato aggirato in una maniera molto logica e molto efficace, semplicemente andando a utilizzare i commenti, o scrivendo **'OR 'a'='a'**. In questo modo andremo a aggirare il sistema di difesa utilizzando l'operatore booleano OR che confrontando due valori o entrambi veri o uno falso e uno vero ritorna sempre vero e sappiamo che a è sempre uguale ad a quindi facendo in questo modo il risultato del confronto sarà sempre TRUE e quindi il database andrà a stampare a video tutti i dati all'interno della tabella predefinita, come possiamo vedere di seguito:



Vulnerability: SQL Injection

User ID:

ID: 'OR 'a'='a'
First name: admin
Surname: admin

ID: 'OR 'a'='a'
First name: Gordon
Surname: Brown

ID: 'OR 'a'='a'
First name: Hack
Surname: Me

ID: 'OR 'a'='a'
First name: Pablo
Surname: Picasso

ID: 'OR 'a'='a'
First name: Bob
Surname: Smith

Facendo in questo modo siamo entrati in possesso di tutti i dati che erano presenti all'interno del database, aggirando il metodo di difesa.