Premises

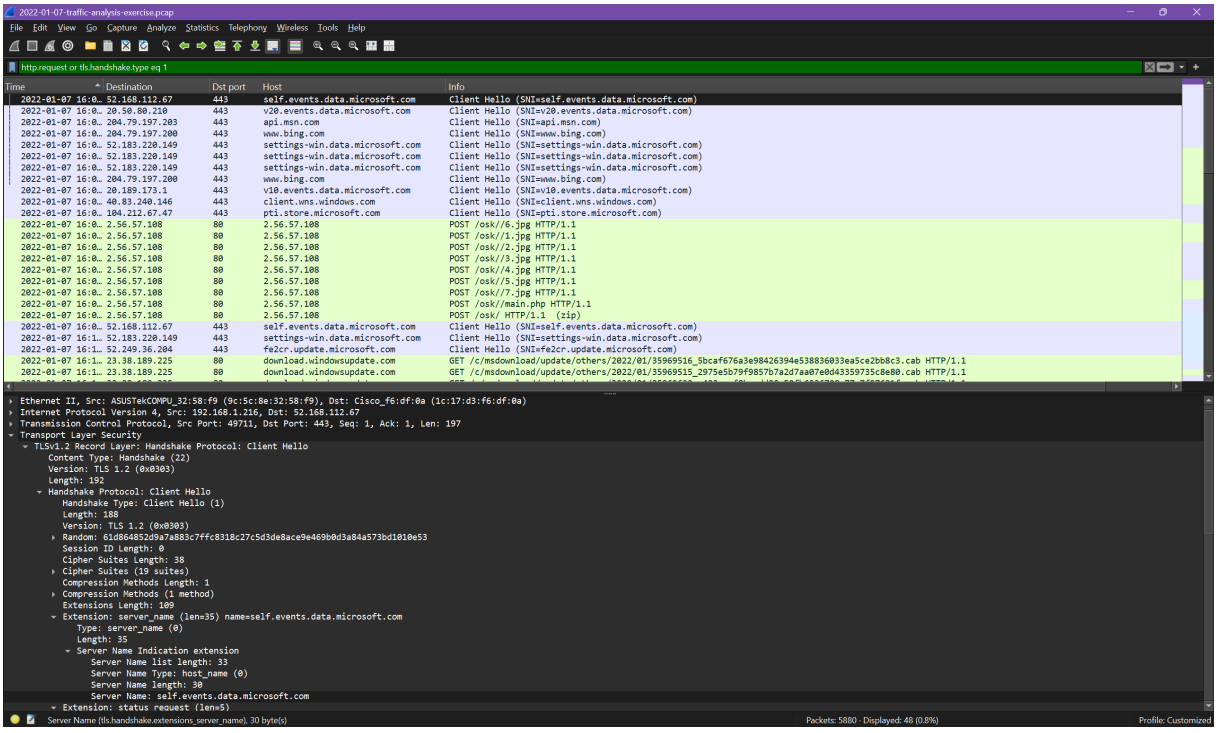
Lab Activity – Liviu-Ioan ZECHERU

All assignments will be structured as snapshot of the assignment then snapshot of my result.

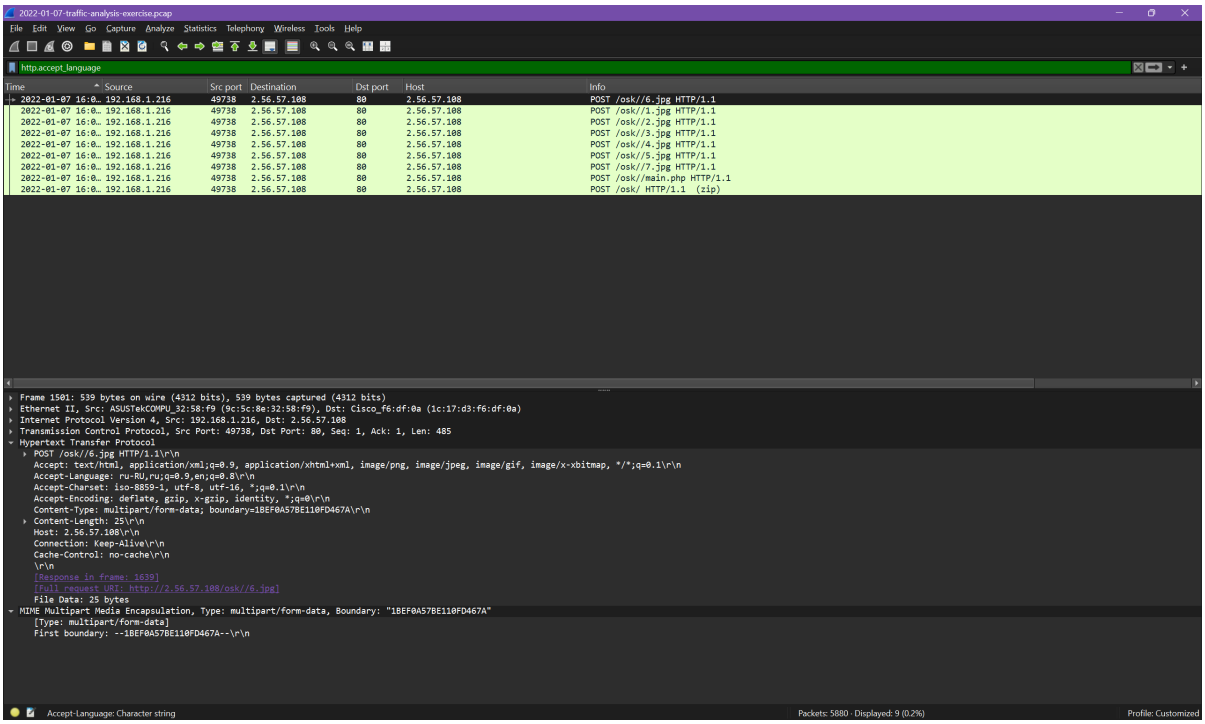
Day 1

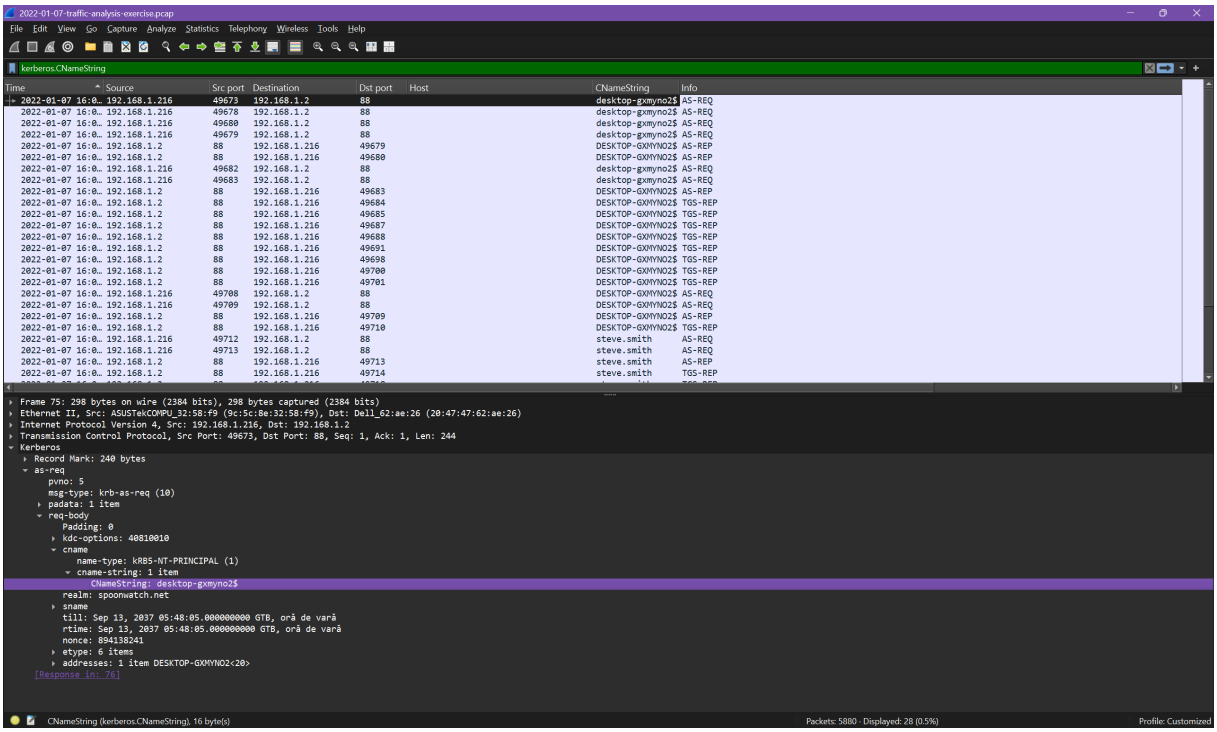
Traffic Analysis

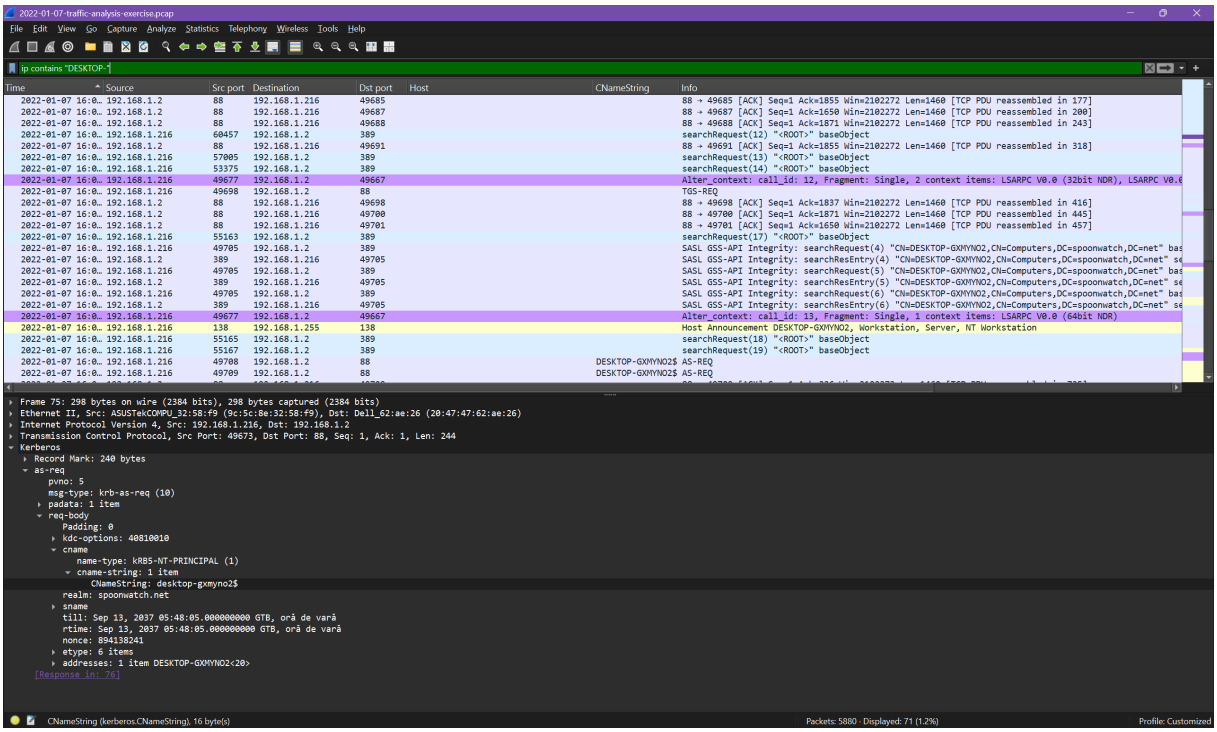
1. Download and unzip the pcap. Password is "infected".
2. Walk through Changing your column display

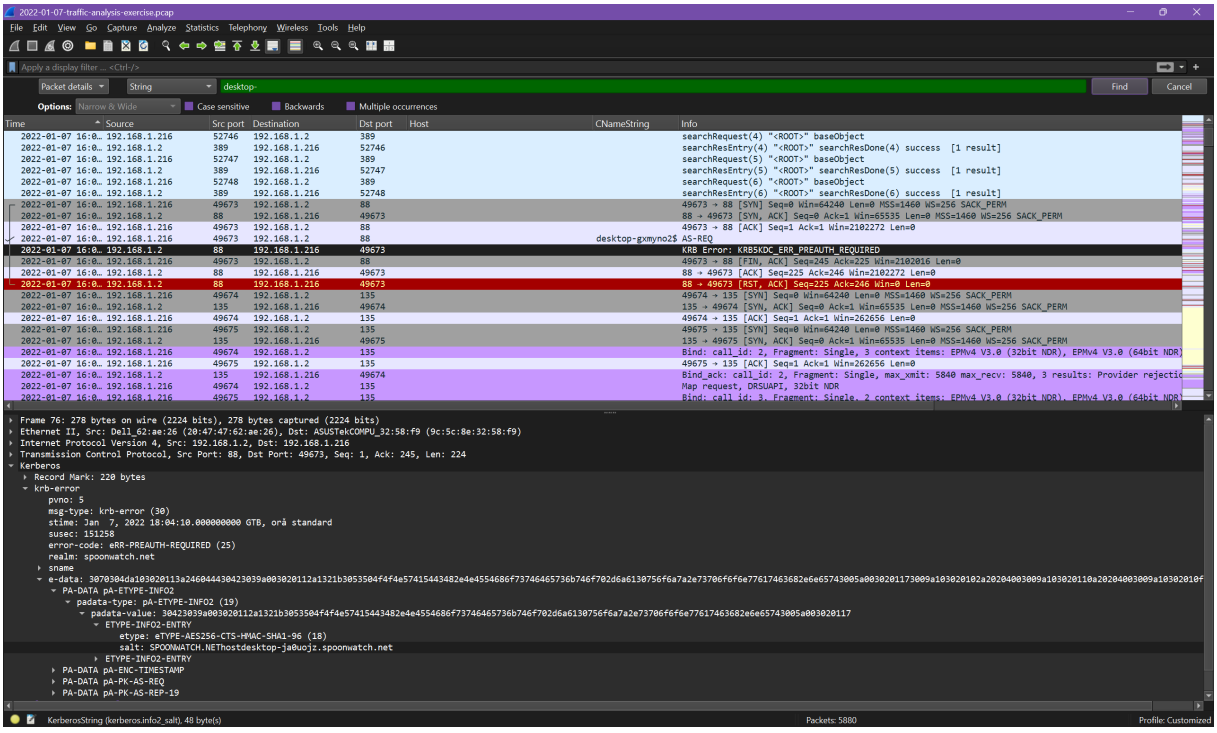


3. Walk through Identify hosts and users

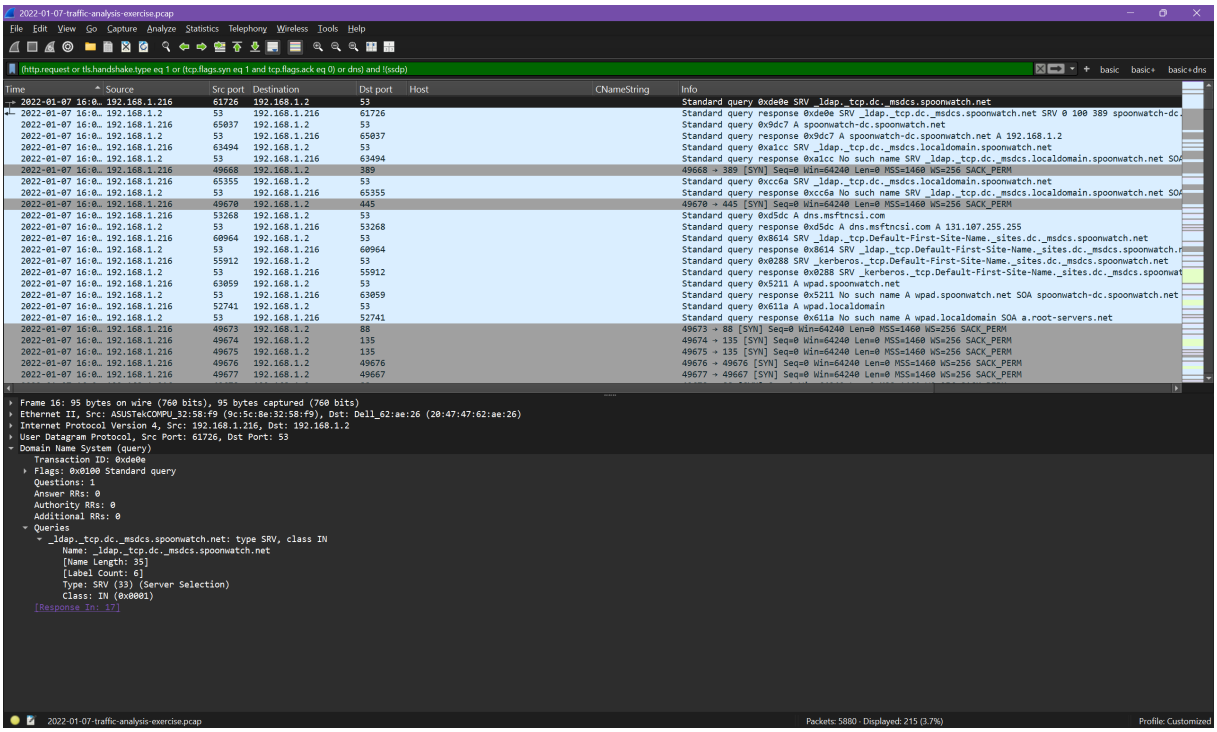

2022-01-07-traffic-analysis-exercise.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http.accept_language
Time Source Src port Destination Dst port Host Info
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk//6.jpg HTTP/1.1
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk//1.jpg HTTP/1.1
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk//2.jpg HTTP/1.1
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk//3.jpg HTTP/1.1
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk//4.jpg HTTP/1.1
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk//5.jpg HTTP/1.1
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk//7.jpg HTTP/1.1
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk//main.php HTTP/1.1
2022-01-07 16:0… 192.168.1.216 49738 2.56.57.108 80 2.56.57.108 POST /osk/ HTTP/1.1 (zip)
Frame 1501: 539 bytes on wire (4312 bits), 539 bytes captured (4312 bits)
Ethernet II, Src: ASUSTekCOMPU_32:58:f9 (9c:5c:8e:32:58:f9), Dst: Cisco_f6:df:0a (1c:17:d3:f6:df:0a)
Internet Protocol Version 4, Src: 192.168.1.216, Dst: 2.56.57.108
Transmission Control Protocol, Src Port: 49738, Dst Port: 80, Seq: 1, Ack: 1, Len: 485
Hypertext Transfer Protocol
POST /osk//6.jpg HTTP/1.1\r\n
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1\r\n
Accept-Language: ru-RU,ru;q=0.9,en;q=0.8\r\n
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1\r\n
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0\r\n
Content-Type: multipart/form-data; boundary=1BEF0A57BE110FD467A\r\n
Content-Length: 25\r\n
Host: 2.56.57.108\r\n
Connection: Keep-Alive\r\n
Cache-Control: no-cache\r\n
\r\n
[Response in frame: 1639]
[Full request URI: http://2.56.57.108/osk//6.jpg]
File Data: 25 bytes
MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "1BEF0A57BE110FD467A"
[Type: multipart/form-data]
First boundary: --1BEF0A57BE110FD467A--\r\n
Accept-Language: Character string
Packets: 5880 · Displayed: 9 (0.2%)
Profile: Customized


2022-01-07-traffic-analysis-exercise.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
kerberos.CNameString
Time Source Src port Destination Dst port Host CNameString Info
2022-01-07 16:0… 192.168.1.216 49673 192.168.1.2 88 desktop-gxmyno2$ AS-REQ
2022-01-07 16:0… 192.168.1.216 49678 192.168.1.2 88 desktop-gxmyno2$ AS-REQ
2022-01-07 16:0… 192.168.1.216 49680 192.168.1.2 88 desktop-gxmyno2$ AS-REQ
2022-01-07 16:0… 192.168.1.216 49679 192.168.1.2 88 desktop-gxmyno2$ AS-REQ
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49679 DESKTOP-GXMYNO2$ AS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49680 DESKTOP-GXMYNO2$ AS-REP
2022-01-07 16:0… 192.168.1.216 49682 192.168.1.2 88 desktop-gxmyno2$ AS-REQ
2022-01-07 16:0… 192.168.1.216 49683 192.168.1.2 88 desktop-gxmyno2$ AS-REQ
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49683 DESKTOP-GXMYNO2$ AS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49684 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49685 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49687 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49688 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49691 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49698 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49700 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49701 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.216 49708 192.168.1.2 88 DESKTOP-GXMYNO2$ AS-REQ
2022-01-07 16:0… 192.168.1.216 49709 192.168.1.2 88 DESKTOP-GXMYNO2$ AS-REQ
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49709 DESKTOP-GXMYNO2$ AS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49710 DESKTOP-GXMYNO2$ TGS-REP
2022-01-07 16:0… 192.168.1.216 49712 192.168.1.2 88 steve.smith AS-REQ
2022-01-07 16:0… 192.168.1.216 49713 192.168.1.2 88 steve.smith AS-REQ
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49713 steve.smith AS-REP
2022-01-07 16:0… 192.168.1.2 88 192.168.1.216 49714 steve.smith TGS-REP
Frame 75: 298 bytes on wire (2384 bits), 298 bytes captured (2384 bits)
Ethernet II, Src: ASUSTekCOMPU_32:58:f9 (9c:5c:8e:32:58:f9), Dst: Dell_62:ae:26 (20:47:47:62:ae:26)
Internet Protocol Version 4, Src: 192.168.1.216, Dst: 192.168.1.2
Transmission Control Protocol, Src Port: 49673, Dst Port: 88, Seq: 1, Ack: 1, Len: 244
Kerberos
Record Mark: 240 bytes
as-req
pvno: 5
msg-type: krb-as-req (10)
padata: 1 item
req-body
Padding: 0
kdc-options: 40810010
cname
name-type: kRB5-NT-PRINCIPAL (1)
cname-string: 1 item
CNameString: desktop-gxmyno2$
realm: spoonwatch.net
sname
till: Sep 13, 2037 05:48:05.000000000 GTB, orå de varå
rtime: Sep 13, 2037 05:48:05.000000000 GTB, orå de varå
nonce: 894118241
etype: 6 items
addresses: 1 item DESKTOP-GXMYNO2<20>
[Response in: 76]
CNameString (kerberos.CNameString), 16 byte(s)
Packets: 5880 · Displayed: 28 (0.5%)
Profile: Customized

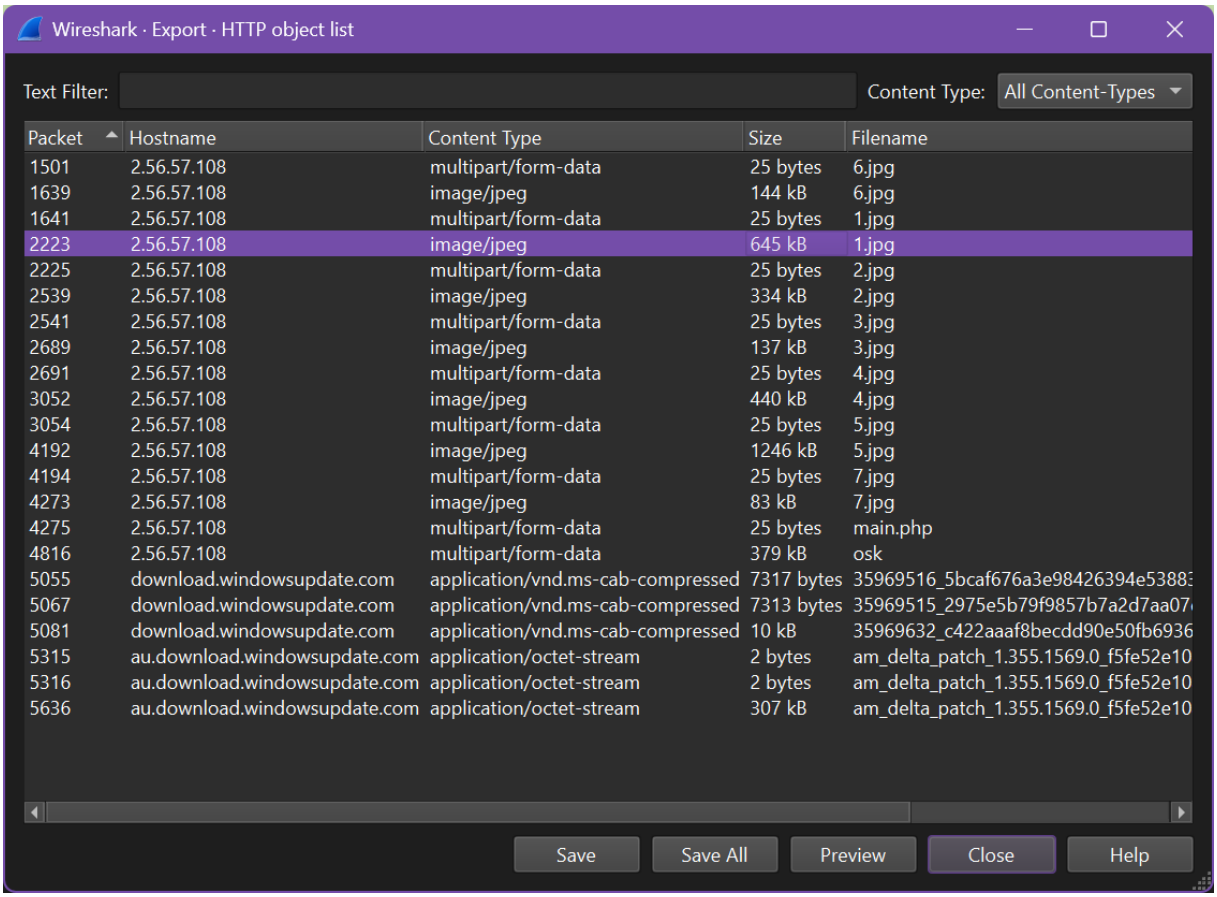


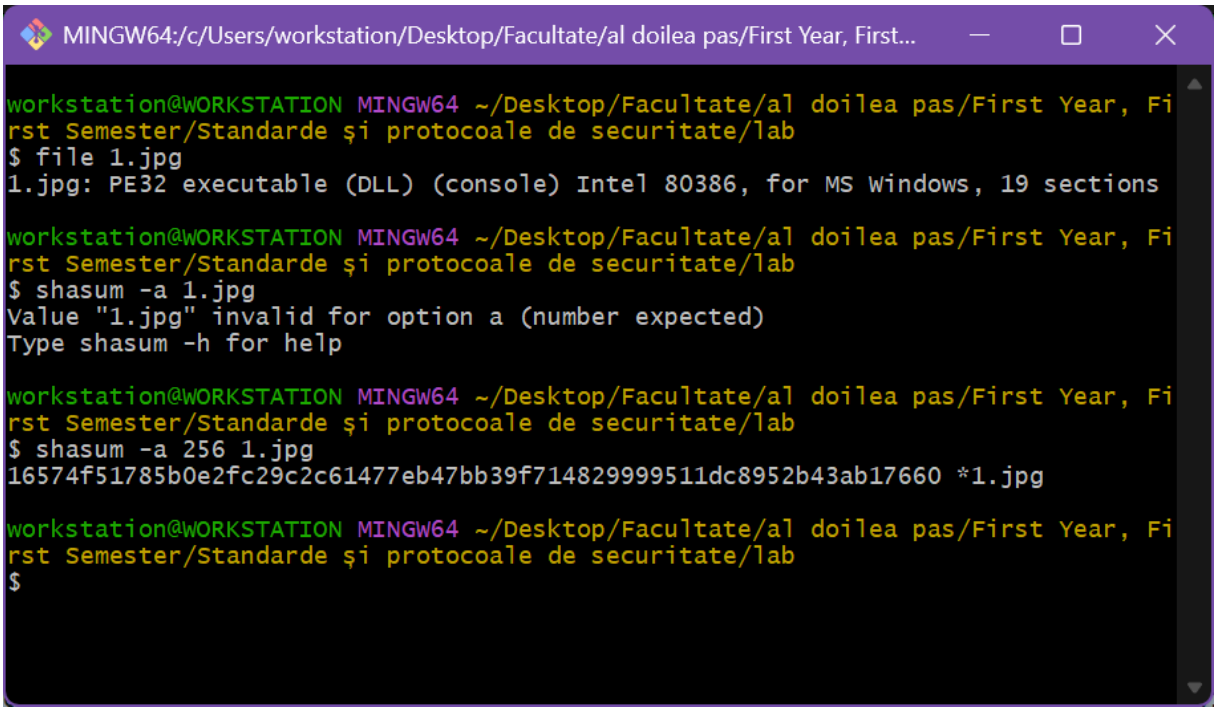


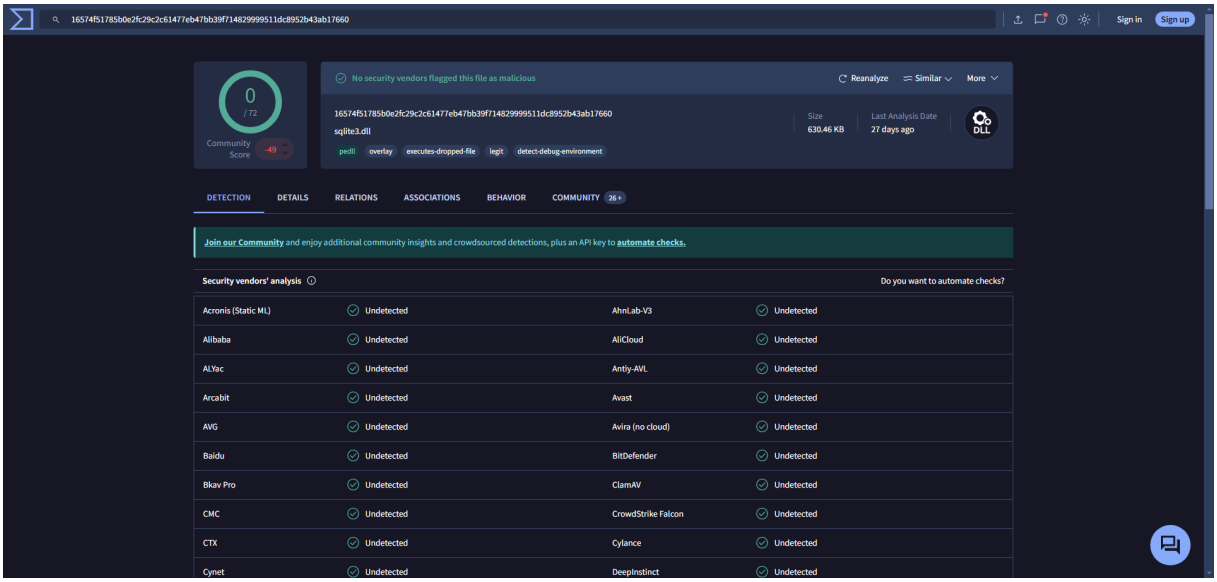
4 Walk through Display Filter Expression



5 Walk through Exporting objects from a pcap





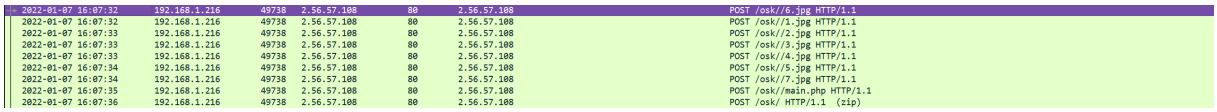


Produce a document with your findings. It should contain:

1. **Executive summary**. What happened (who, when what)
2. **Victim details** - hostname, IP address, MAC address, Windows user account name.
3. **IOCs** - IP adressess, domains, URLs, files, etc.
4. **Screenshots** with your findings.

1. Executive Summary

On 7th of January (my birthday 😁), 2022 at approximately half past 16:07, a Windows host owned by Steve Smith was compromised with the OskiStealer malware.



2022-01-07 16:07:32	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk//6.jpg HTTP/1.1
2022-01-07 16:07:32	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk//1.jpg HTTP/1.1
2022-01-07 16:07:33	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk//2.jpg HTTP/1.1
2022-01-07 16:07:33	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk//3.jpg HTTP/1.1
2022-01-07 16:07:33	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk//4.jpg HTTP/1.1
2022-01-07 16:07:34	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk//5.jpg HTTP/1.1
2022-01-07 16:07:34	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk//7.jpg HTTP/1.1
2022-01-07 16:07:35	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk//main.php HTTP/1.1
2022-01-07 16:07:36	192.168.1.216	49738	2.56.57.108	80	2.56.57.108	POST /osk/ HTTP/1.1 (zip)

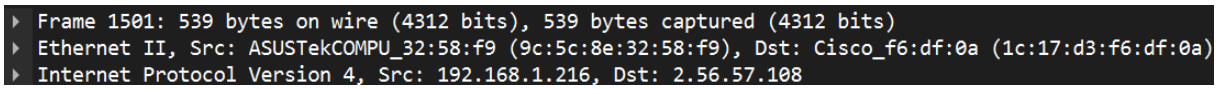
2. Victim details

MAC address: 9c:5c:8e:32:58:f9

IP address: 192.168.1.216

Host name: DESKTOP-GXMYNO2

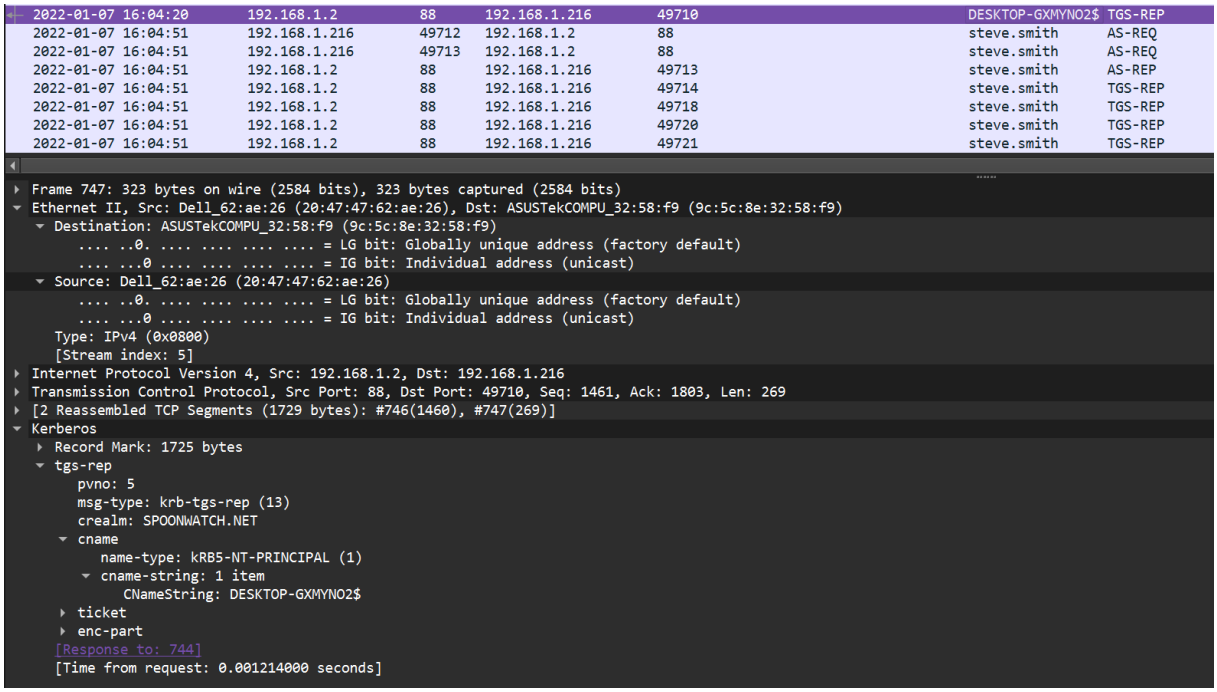
Windows user account: steve.smith



```
▸ Frame 1501: 539 bytes on wire (4312 bits), 539 bytes captured (4312 bits)
▸ Ethernet II, Src: ASUSTekCOMPU_32:58:f9 (9c:5c:8e:32:58:f9), Dst: Cisco_f6:df:0a (1c:17:d3:f6:df:0a)
▸ Internet Protocol Version 4, Src: 192.168.1.216, Dst: 2.56.57.108
```

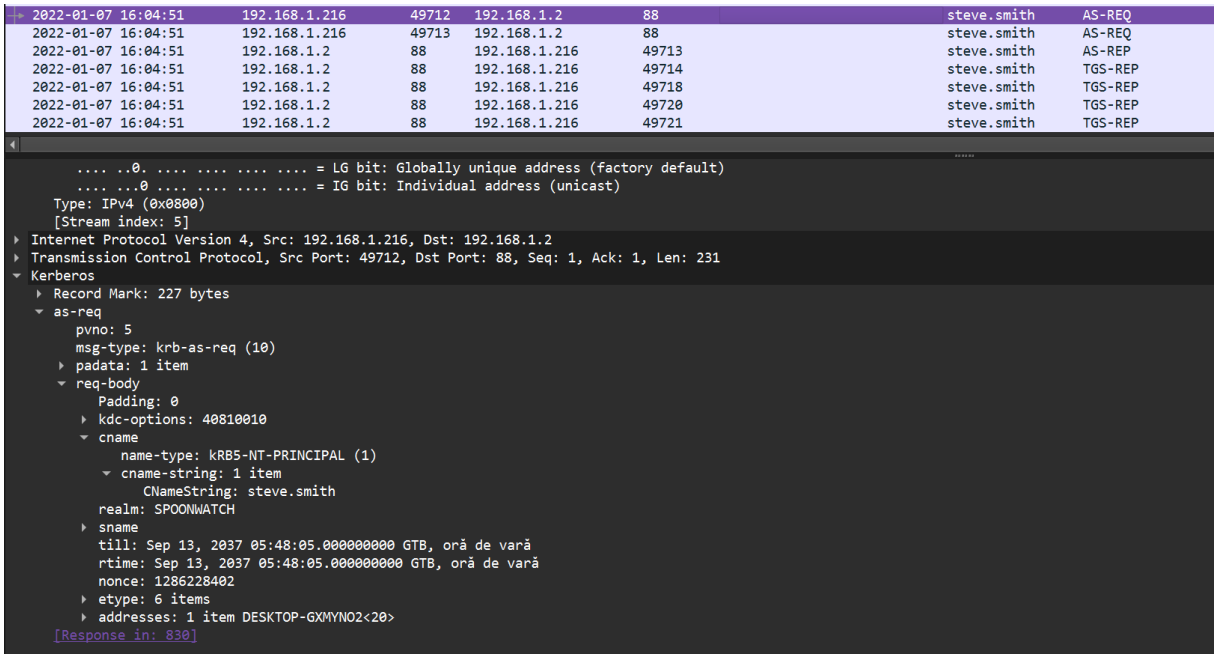
Here we can see both MAC and IP addresses.

Investigating Kerberos we can link his identity.



2022-01-07 16:04:20	192.168.1.2	88	192.168.1.216	49710	DESKTOP-GXMYNO2$	TGS-REP
2022-01-07 16:04:51	192.168.1.216	49712	192.168.1.2	88	steve.smith	AS-REQ
2022-01-07 16:04:51	192.168.1.216	49713	192.168.1.2	88	steve.smith	AS-REQ
2022-01-07 16:04:51	192.168.1.2	88	192.168.1.216	49713	steve.smith	AS-REP
2022-01-07 16:04:51	192.168.1.2	88	192.168.1.216	49714	steve.smith	TGS-REP
2022-01-07 16:04:51	192.168.1.2	88	192.168.1.216	49718	steve.smith	TGS-REP
2022-01-07 16:04:51	192.168.1.2	88	192.168.1.216	49720	steve.smith	TGS-REP
2022-01-07 16:04:51	192.168.1.2	88	192.168.1.216	49721	steve.smith	TGS-REP

```
▸ Frame 747: 323 bytes on wire (2584 bits), 323 bytes captured (2584 bits)
▾ Ethernet II, Src: Dell_62:ae:26 (20:47:47:62:ae:26), Dst: ASUSTekCOMPU_32:58:f9 (9c:5c:8e:32:58:f9)
  ▾ Destination: ASUSTekCOMPU_32:58:f9 (9c:5c:8e:32:58:f9)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: Dell_62:ae:26 (20:47:47:62:ae:26)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
    [Stream index: 5]
▸ Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.216
▸ Transmission Control Protocol, Src Port: 88, Dst Port: 49710, Seq: 1461, Ack: 1803, Len: 269
▸ [2 Reassembled TCP Segments (1729 bytes): #746(1460), #747(269)]
▾ Kerberos
  ▸ Record Mark: 1725 bytes
  ▾ tgs-rep
      pvno: 5
      msg-type: krb-tgs-rep (13)
      crealm: SPOONWATCH.NET
    ▾ cname
        name-type: kRB5-NT-PRINCIPAL (1)
      ▾ cname-string: 1 item
          CNameString: DESKTOP-GXMYNO2$
    ▸ ticket
    ▸ enc-part
    [Response to: 744]
    [Time from request: 0.001214000 seconds]
```



3. IOCs

Suspect malicious traffic:

Host	Port	Action
2.56.57.108	80	POST /osk//6.jpg HTTP/1.1
2.56.57.108	80	POST /osk//1.jpg HTTP/1.1
2.56.57.108	80	POST /osk//2.jpg HTTP/1.1
2.56.57.108	80	POST /osk//3.jpg HTTP/1.1
2.56.57.108	80	POST /osk//4.jpg HTTP/1.1
2.56.57.108	80	POST /osk//5.jpg HTTP/1.1
2.56.57.108	80	POST /osk//7.jpg HTTP/1.1
2.56.57.108	80	POST /osk//main.php HTTP/1.1
2.56.57.108	80	POST /osk/ HTTP/1.1 (zip)

The files associated with the malware are just copies of legitimate files just for didactic purposes. The real files were actually malware. For example, **1.jpg** is just a copy of **sqlite3.dll** a DLL library for accessing a Sqlite database.

Files info:

Hash of the file	Size of file (bytes)	File location	File type	File commonly used name
16574f51785b0e2fc29c2c61477eb47bb39f714829999511dc8952b43ab17660	645592	http://2.56.57.108/osk//1.jpg	PE32 executable (DLL) (console) Intel 80386, for MS Windows	sqlite3.dll
a770ecba3b08bbabd0a567fc978e50615f8b346709f8eb3cfacf3faab24090ba	334288	http://2.56.57.108/osk//2.jpg	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows	freebl3.dll
3fe6b1c54b8cf28f571e0c5d6636b4069a8ab00b4f11dd842cfec00691d0c9cd	137168	http://2.56.57.108/osk//3.jpg	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows	mozglue.dll
334e69ac9367f708ce601a6f490ff227d6c20636da5222f148b25831d22e13d4	440120	http://2.56.57.108/osk//4.jpg	PE32 executable (DLL) (console) Intel 80386, for MS Windows	msvcp140.dll
e2935b5b28550d47dc971f456d6961f20d1633b4892998750140e0eaa9ae9d78	1246160	http://2.56.57.108/osk//5.jpg	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows	nss3.dll
43536adef2ddcc811c28d35fa6ce3031029a2424ad393989db36169ff2995083	144848	http://2.56.57.108/osk//6.jpg	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows	softokn3.dll
c40bb03199a2054dabfc7a8e01d6098e91de7193619effbd0f142a7bf031c14d	83784	http://2.56.57.108/osk//7.jpg	PE32 executable (DLL) (console) Intel 80386, for MS Windows	vcruntime140.dll

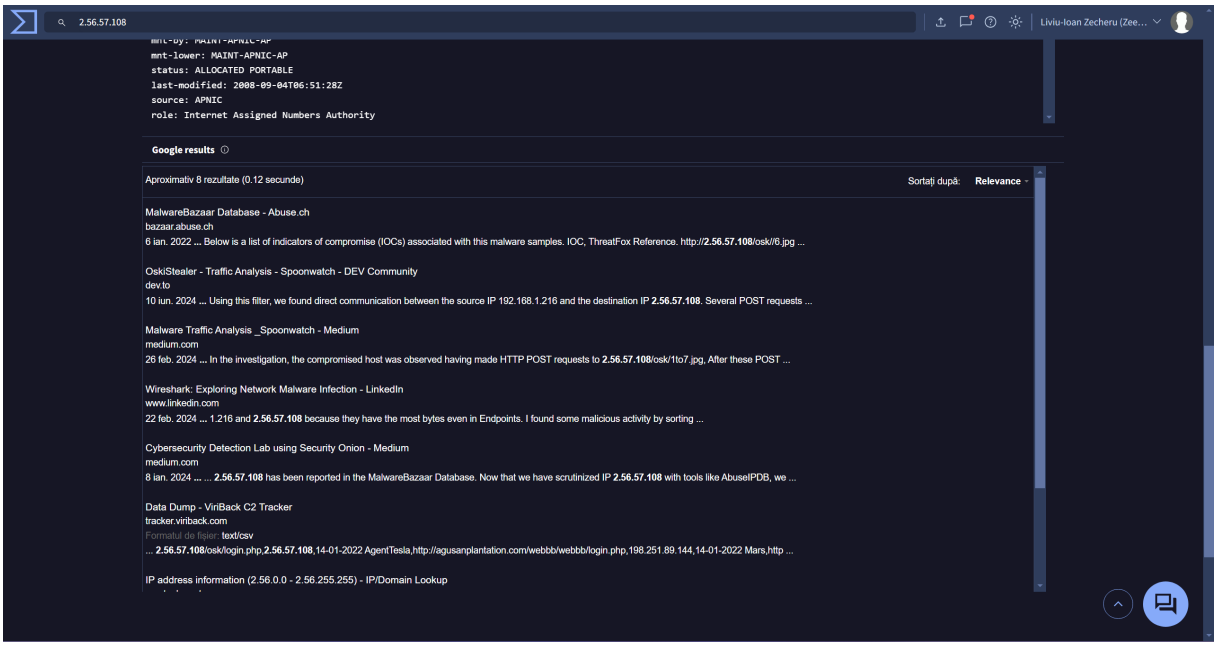
Script used for populating the above table:

```
#!/usr/bin/env bash

#

# Usage: ./my_script.sh file1 file2 ...

#

# Output format (one line per file):

#   sha256_hash size_in_bytes file_location file_type fileDescription

#

# Where:

#   - file_type       => from the API response "magic" field

#   - fileDescription => from the API response "meaningful_name" field


# -- BEGIN CONFIGURATION --

API_KEY="MY_API_KEY"

BASE_URL="http://2.56.57.108/osk/"

VT_API_ENDPOINT="https://www.virustotal.com/api/v3/files"

# -- END CONFIGURATION --
```

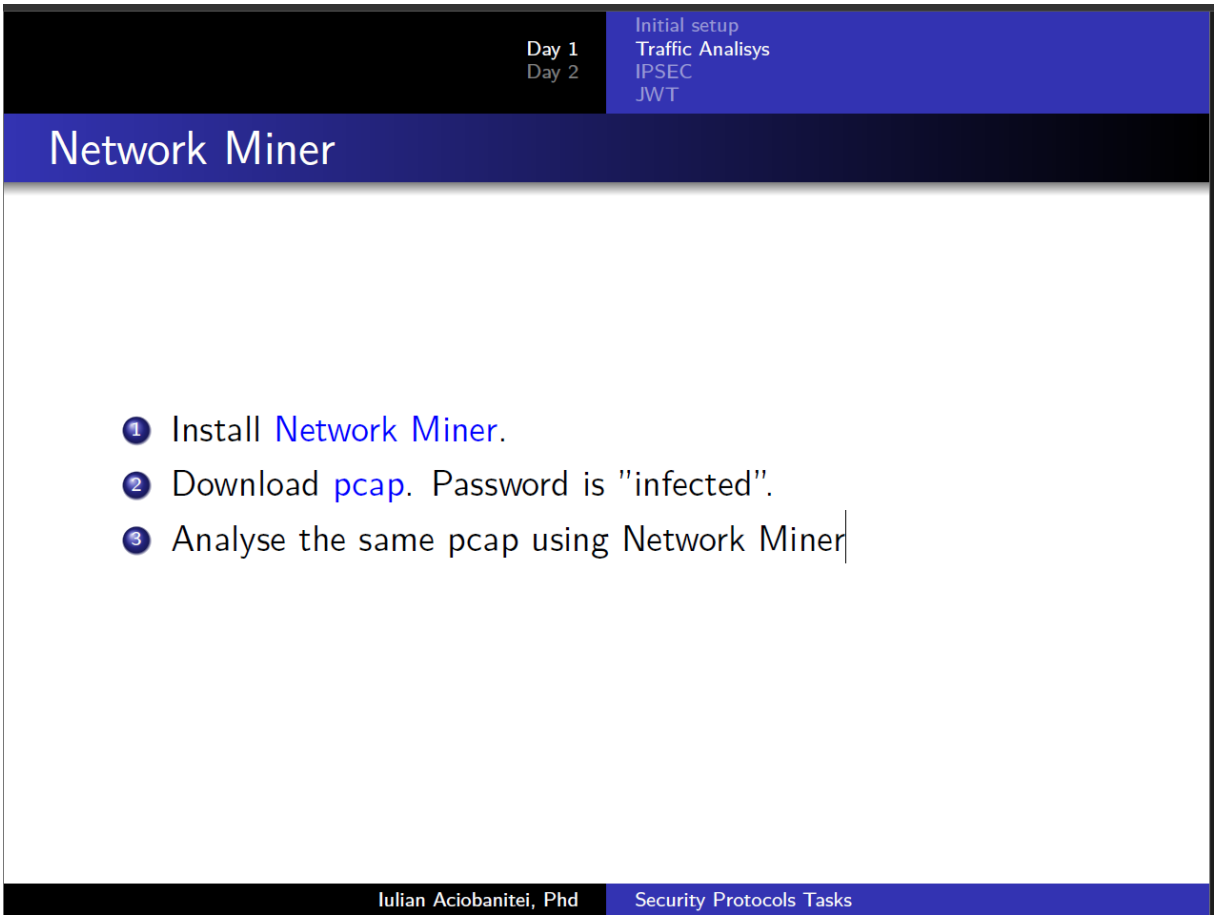
```
for file in "$@"; do
  # Ensure it's a regular file
  if [[ -f "$file" ]]; then

    # 1. Compute SHA-256
    #    On Git Bash (Windows) this should work, else use 'shasum -a 256'.
    sha256_hash="$(sha256sum "$file" | awk '{print $1}')"

    # 2. File size in bytes
    #    On Linux: stat -c%s
    #    On some Git Bash environments: stat -f%z
    size_in_bytes="$(stat -c%s "$file" 2>/dev/null || stat -f%z "$file" 2>/dev/null)"

    # 3. Construct file location
    file_name="$(basename "$file")"
    file_location="${BASE_URL}/${file_name}"

    # 4. Query VirusTotal for this SHA-256
    response_json="$(curl --silent --request GET \
      --url "${VT_API_ENDPOINT}/${sha256_hash}" \
      --header "x-apikey: ${API_KEY}")"

    # Convert the response JSON to a single line so our regex in awk is simpler
    single_line_json="$(echo "$response_json" | tr -d '\r\n')"

    # 5a. Extract "magic" => file_type
    #     We'll look for something like: "magic": "PE32 executable..."
    file_type="$(
      echo "$single_line_json" \
      | awk '{
          match($0, /"magic"[[:space:]]*:[[:space:]]*"([^"]*)"/, arr)
          if (RSTART > 0) {
            print arr[1]
          } else {
            print "N/A"
          }
```

```
        }'
    )"

    # 5b. Extract "meaningful_name" => fileDescription
    #     Looks for: "meaningful_name": "someFileName.dll"
    file_description="$(
      echo "$single_line_json" \
      | awk '{
          match($0, /"meaningful_name"[[:space:]]*:[[:space:]]*"([^"]*)"/, arr)
          if (RSTART > 0) {
            print arr[1]
          } else {
            print "N/A"
          }
        }'
    )"

    # 6. Print
    # sha256_hash size_in_bytes file_location file_type fileDescription
    # Wrapping the final 2 fields in quotes, to protect spaces
    echo -e "$sha256_hash\t$size_in_bytes\t$file_location\t$file_type\t$file_description"

  else
    echo "Skipping '$file': not a regular file."
  fi
done
```

The IP is associated with OskiStealer as per VirusTotal external reports.



Network Miner

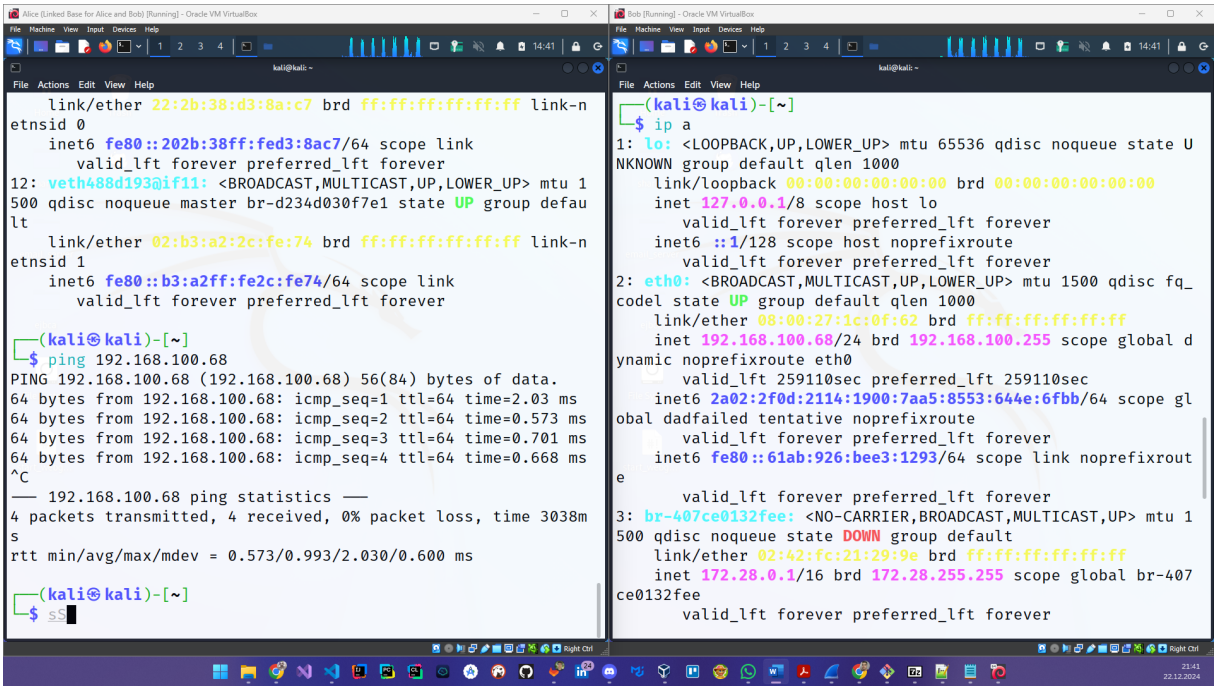


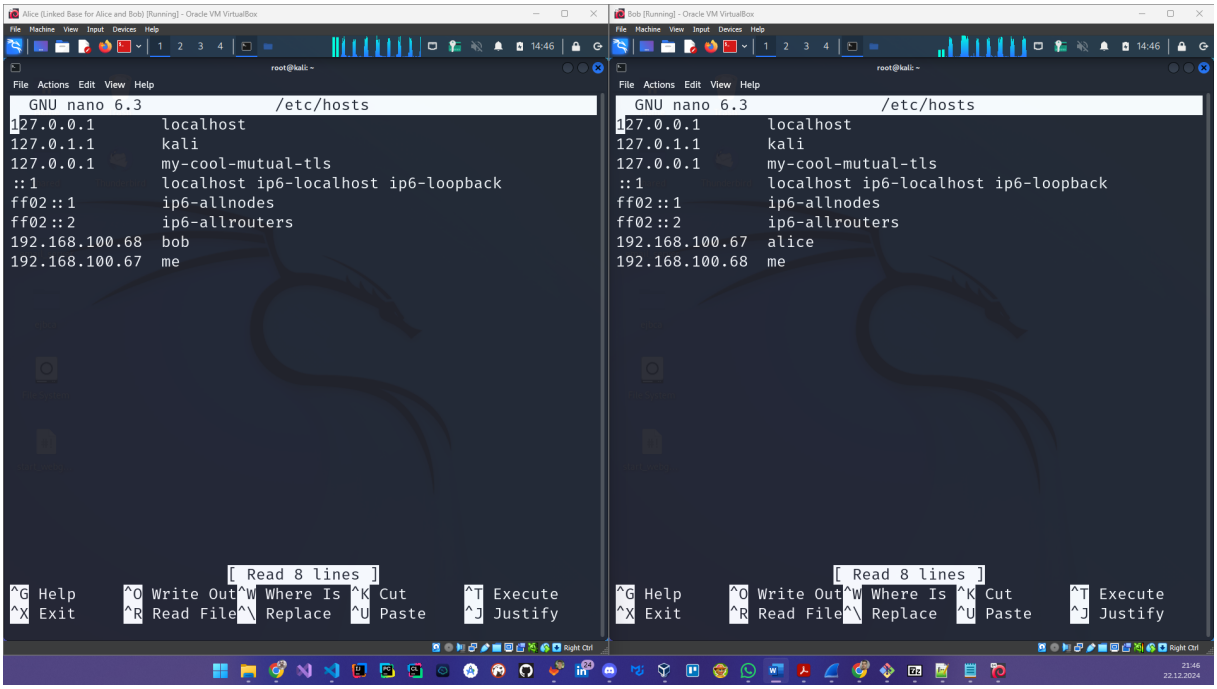
We did it live in the dedicated lab class.

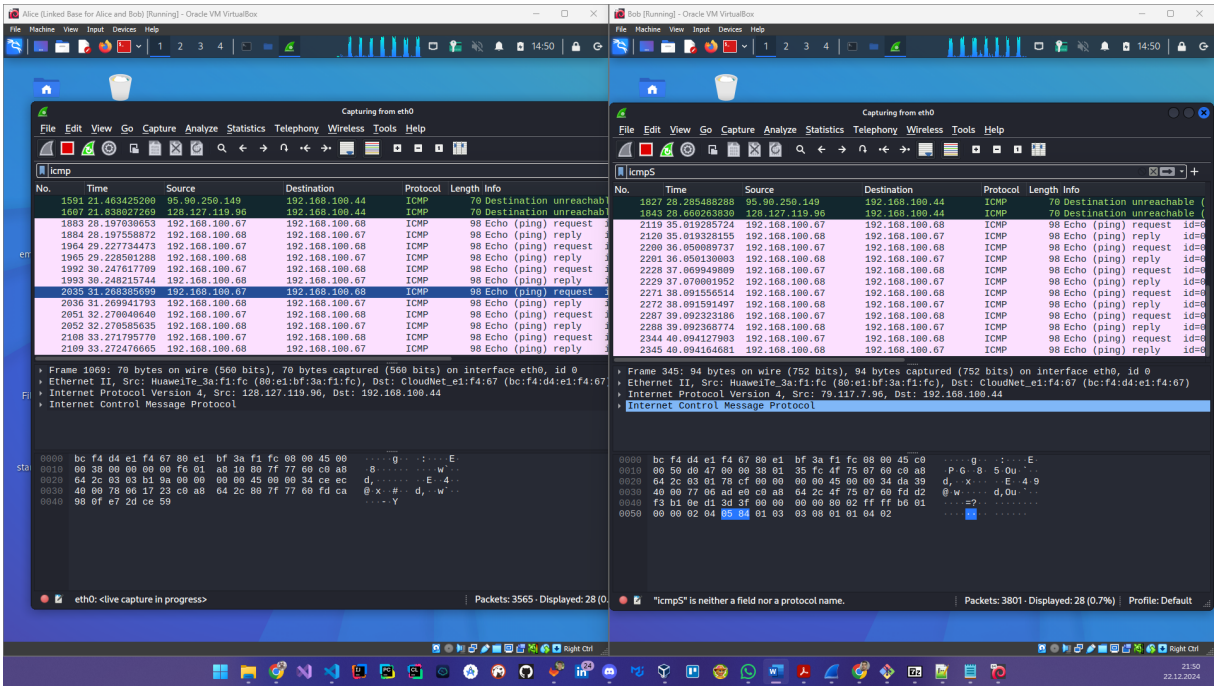
IPSEC

IPSEC - initial setup

1. Use Kali VM already created
2. Setup the vm in bridge mode
3. Choose a partner
4. Ping to your partner's IP address
5. Change hosts file and name partner's IP
6. Start Wireshark
7. Monitor (ICMP) traffic from host to host generated by ping cmd



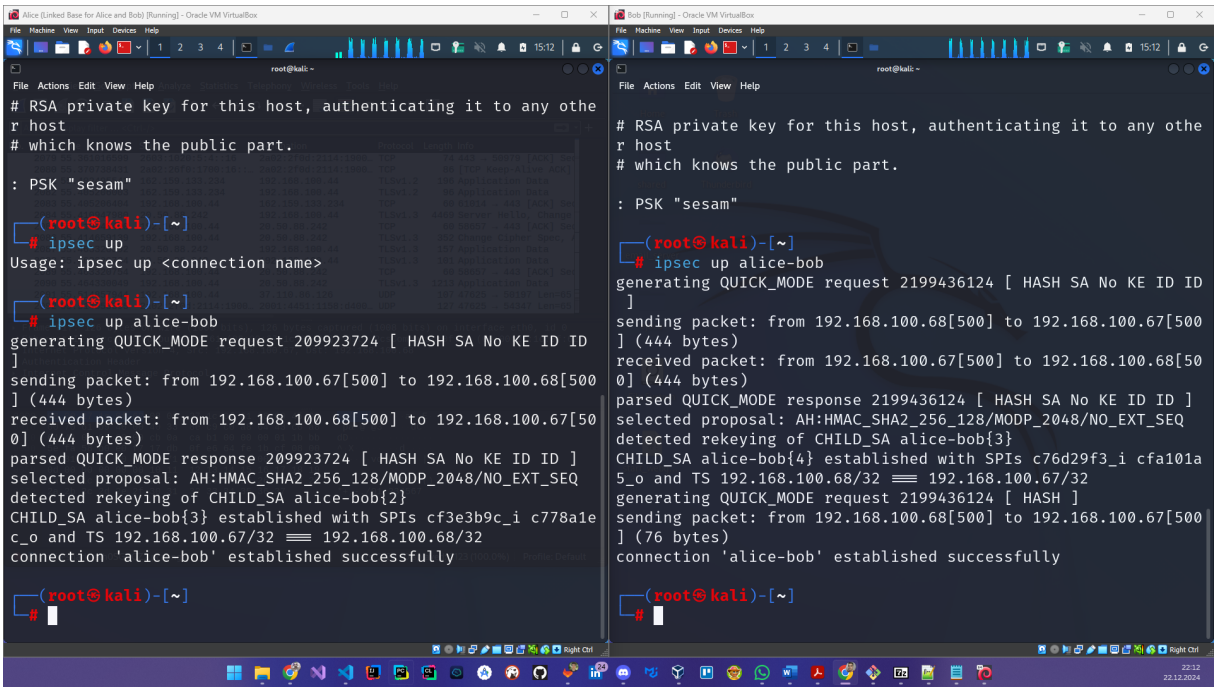

Alice (Linked Base for Alice and Bob) [Running] - Oracle VM VirtualBox
Bob [Running] - Oracle VM VirtualBox
root@kali: ~
File Actions Edit View Help
GNU nano 6.3 /etc/hosts
127.0.0.1 localhost
127.0.1.1 kali
127.0.0.1 my-cool-mutual-tls
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.100.68 bob
192.168.100.67 me
192.168.100.67 alice
192.168.100.68 me
[Read 8 lines]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp
icmpS
No. Time Source Destination Protocol Length Info
Frame 1069: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface eth0, id 0
Ethernet II, Src: HuaweiTe_3a:f1:fc (80:e1:bf:3a:f1:fc), Dst: CloudNet_e1:f4:67 (bc:f4:d4:e1:f4:67)
Internet Protocol Version 4, Src: 128.127.119.96, Dst: 192.168.100.44
Internet Control Message Protocol
Frame 345: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface eth0, id 0
Ethernet II, Src: HuaweiTe_3a:f1:fc (80:e1:bf:3a:f1:fc), Dst: CloudNet_e1:f4:67 (bc:f4:d4:e1:f4:67)
Internet Protocol Version 4, Src: 79.117.7.96, Dst: 192.168.100.44
Internet Control Message Protocol
eth0: <live capture in progress>
Packets: 3565 · Displayed: 28 (0.
"icmpS" is neither a field nor a protocol name.
Packets: 3801 · Displayed: 28 (0.7%) Profile: Default

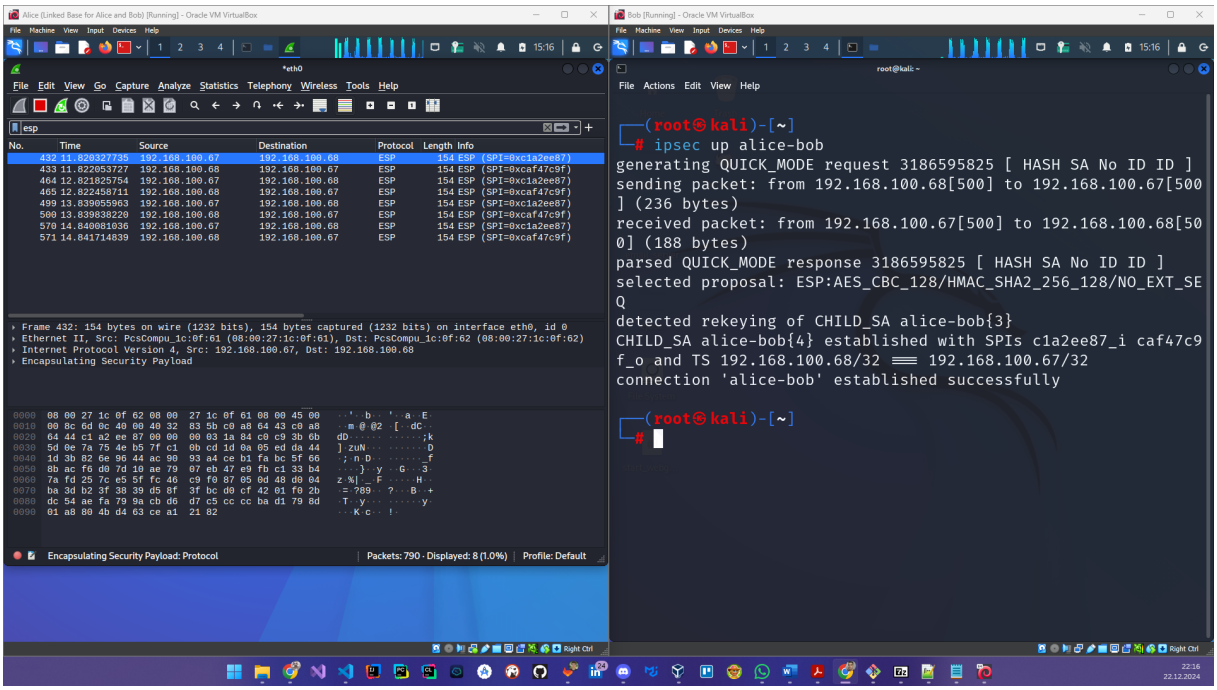
IPSEC documentation

- Check ipsec.conf file documentation
- Check host-to-host configuration example
- Install strongswan (already installed)
- Create AH tunnel based on PSK (PreShared Key)
- Create ESP tunnel based on PSK
- Create ESP tunnel based on certificates
- Check how to generate a certificate guide
- Files location:
  - /etc/ipsec.conf
  - /etc/ipsec.secrets
- Useful cmds:
  - systemctl status ipsec
  - systemctl restart ipsec

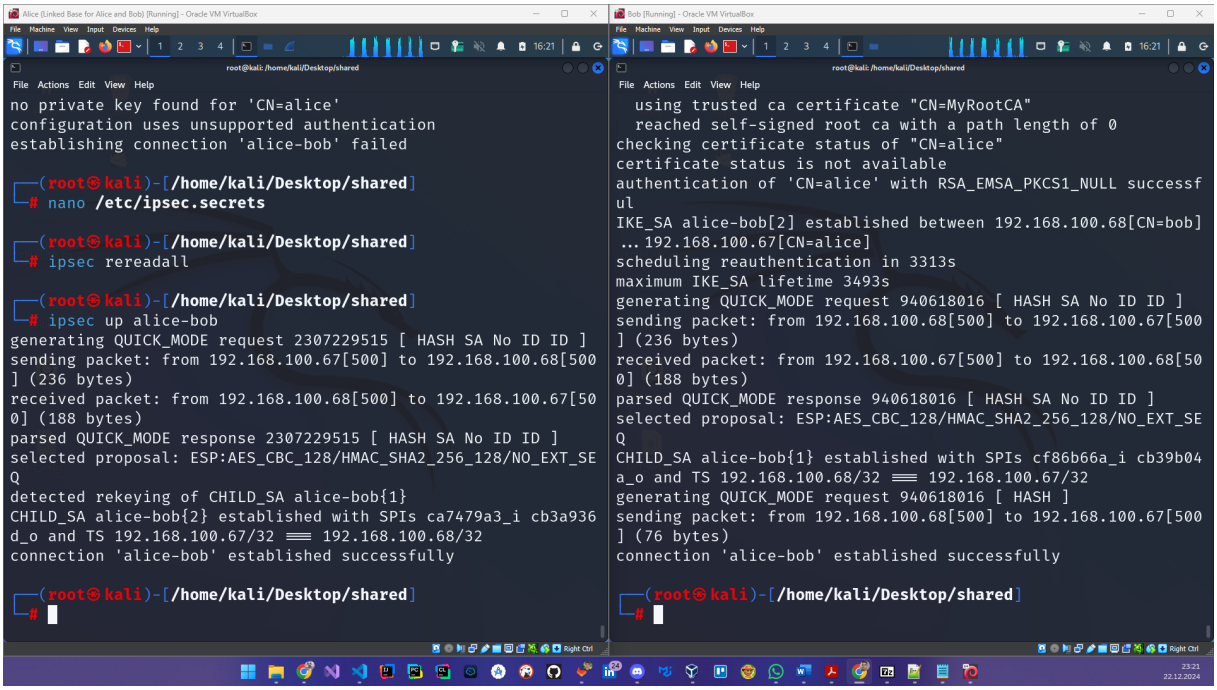
Here we set up an AH tunnel based on PSK (**sesam**).



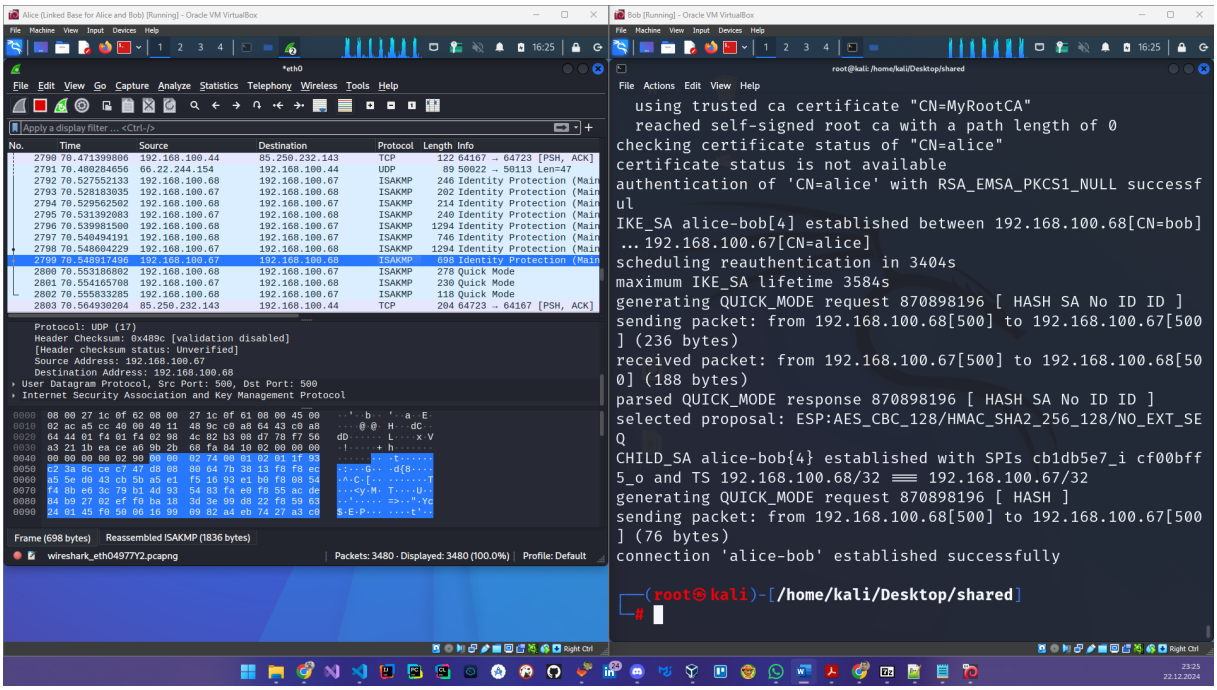
Here we set up an ESP tunnel based also on PSK (**sesam**).



For the ESP+certificates part, I had to generate a CA and put it in **/etc/ipsec.d/cacerts** (on both machines) and also sign both Alice and Bob's certificates with it in order for parties to trust each other. Also I had to modify **/etc/ipsec.secrets** to add : **RSA "alice/bob_privkey.pem"** for both parties to testify the ownership of their certificates.



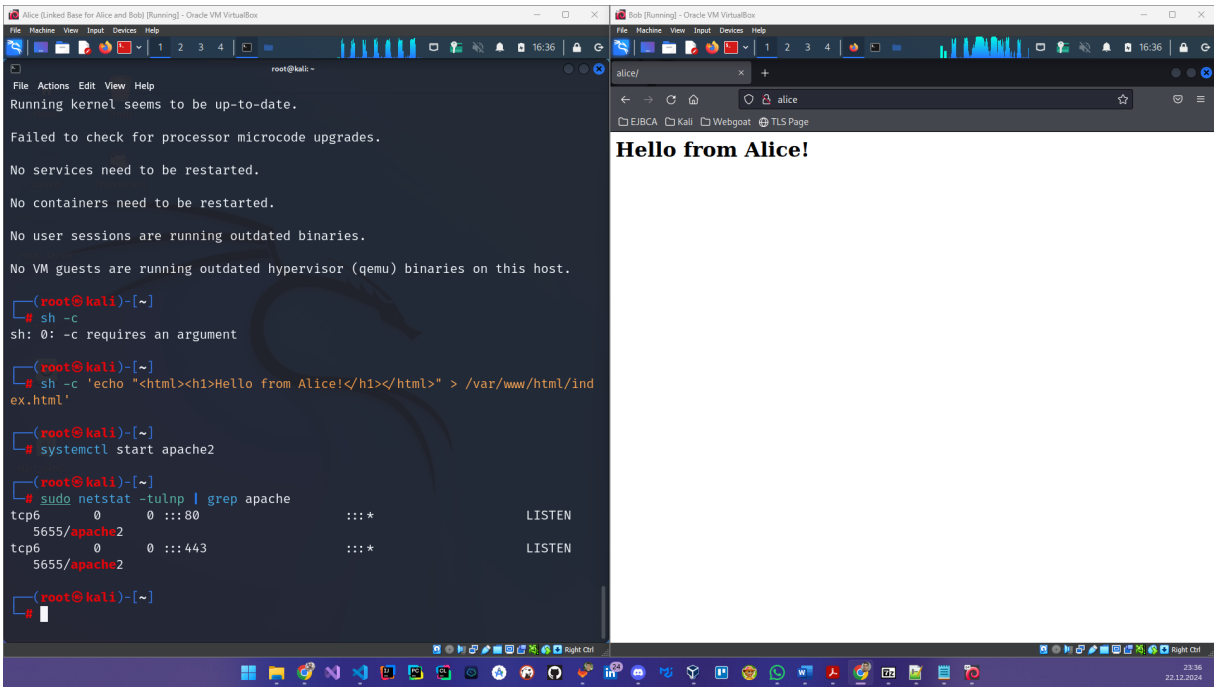
Here is the netcap when using IPSec with the ESP+certificates approach.



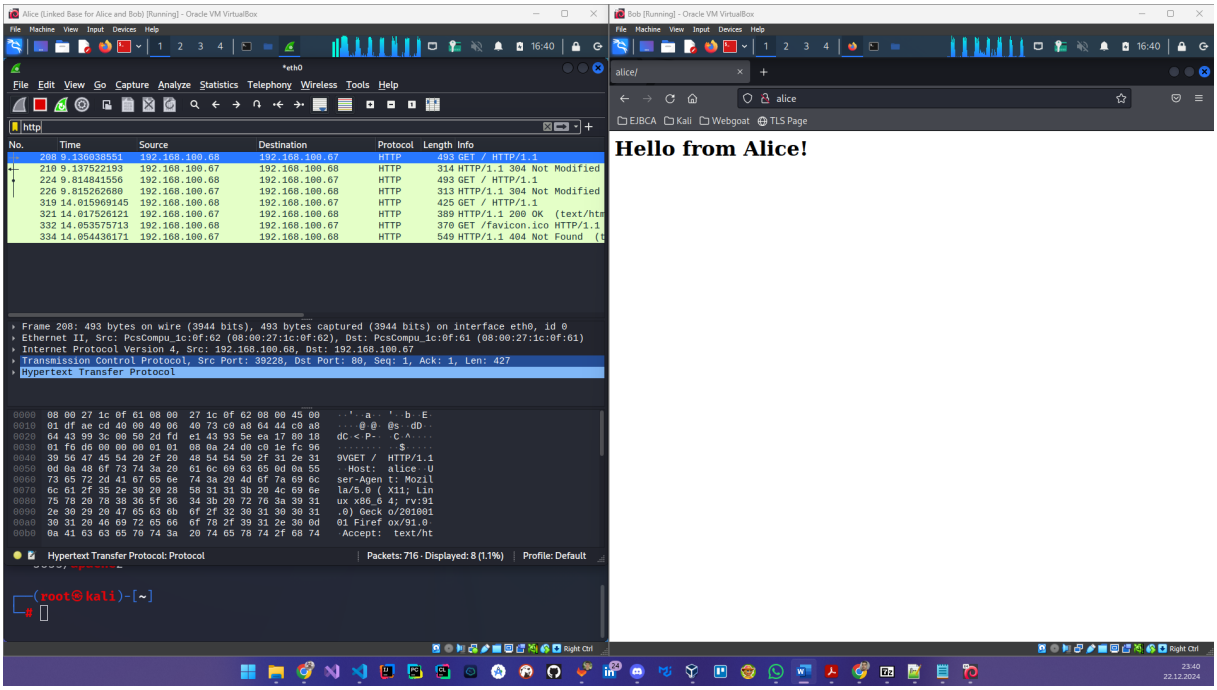
Secure HTTP traffic

Secure HTTP traffic

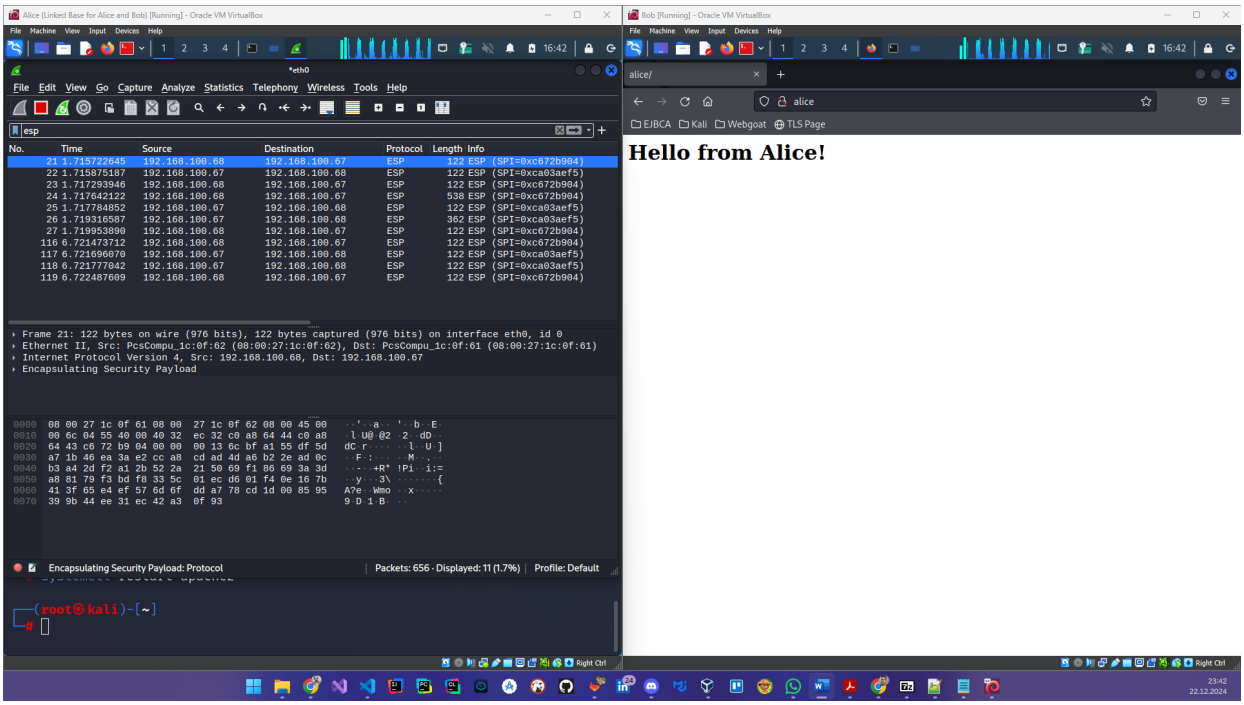
- Stop ipsec service
- Install Apache server
- Create a new html web page
- Start Apache server
- Observe clear HTTP traffic and webpage content
- Secure communication using IPSEC



Here we can see simple, plain, unencrypted HTTP requests.



After securing with **IPSec (ESP+certificates)**, there are no more **HTTP** requests. When a **GET** request is received it is rerouted through **ESP** protocol.

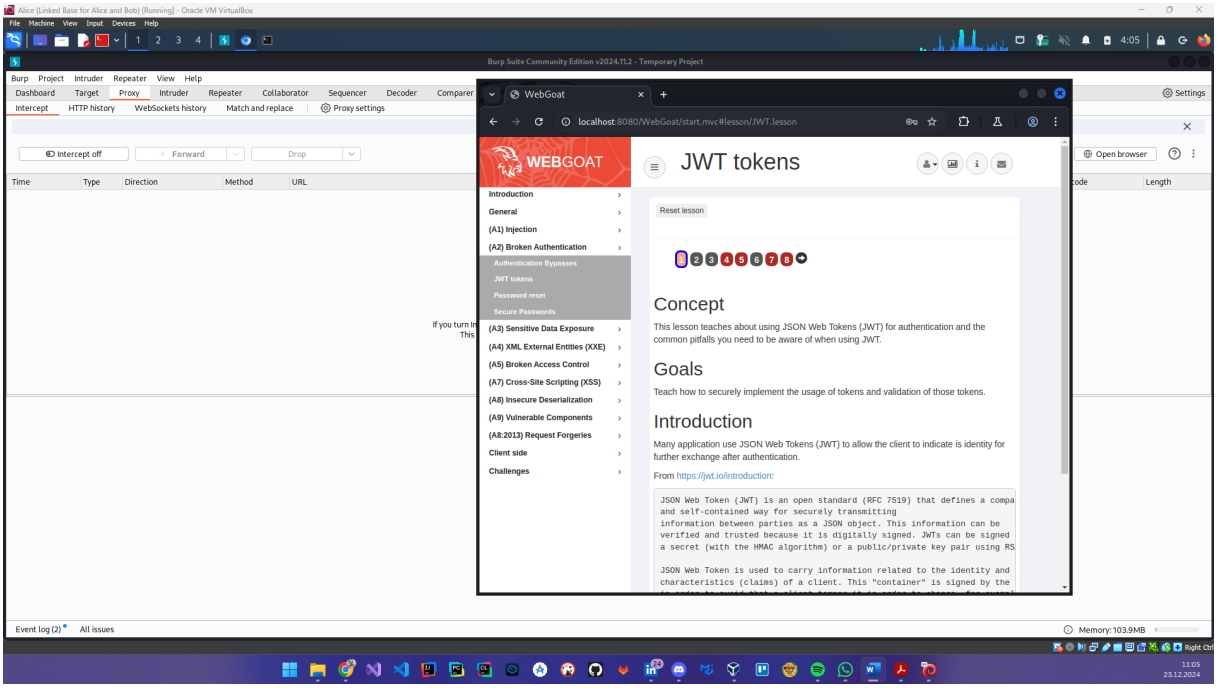


JWT

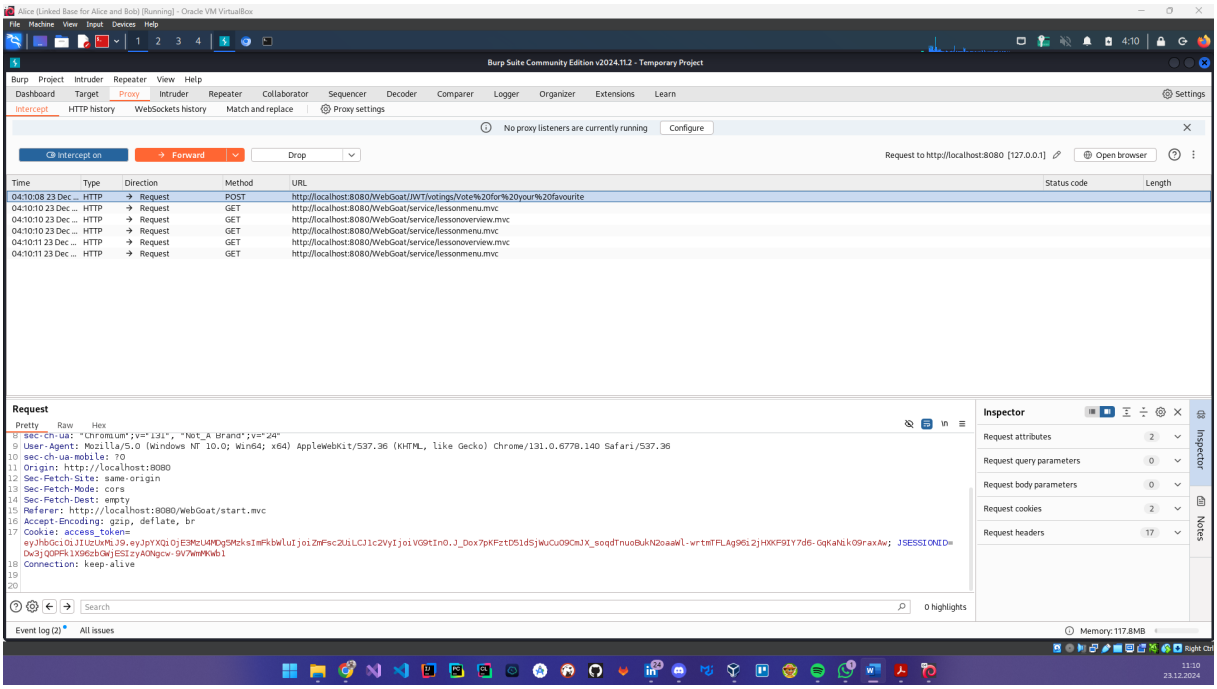
JWT - To do

- Install&Configure BurpSuite
- Start docker & webgoat
  - in Desktop folder, run 'sudo start_webgoat.sh'
- Create account on: http://localhost:8080/WebGoat
- Go to lesson (A2) Broken Authentication -> JWT Tokens
- At minimum, complete challenges 4 & 5
- At best, complete also 7 & 8

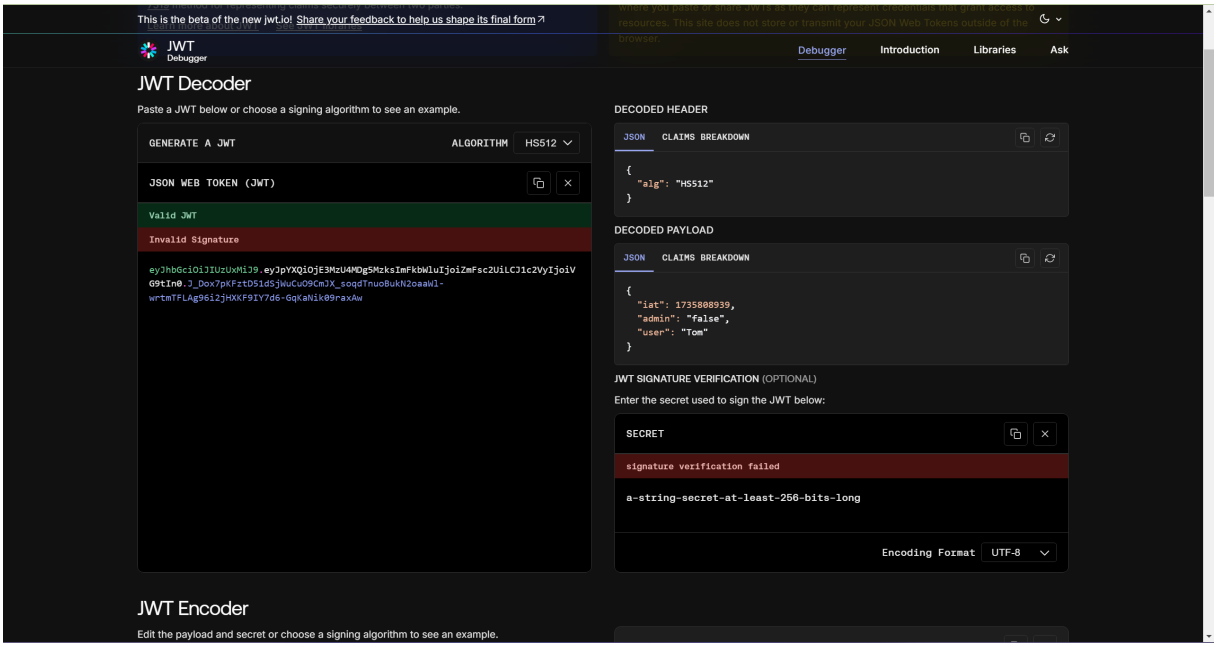
Firstly, I create an account and login on WebGoat.



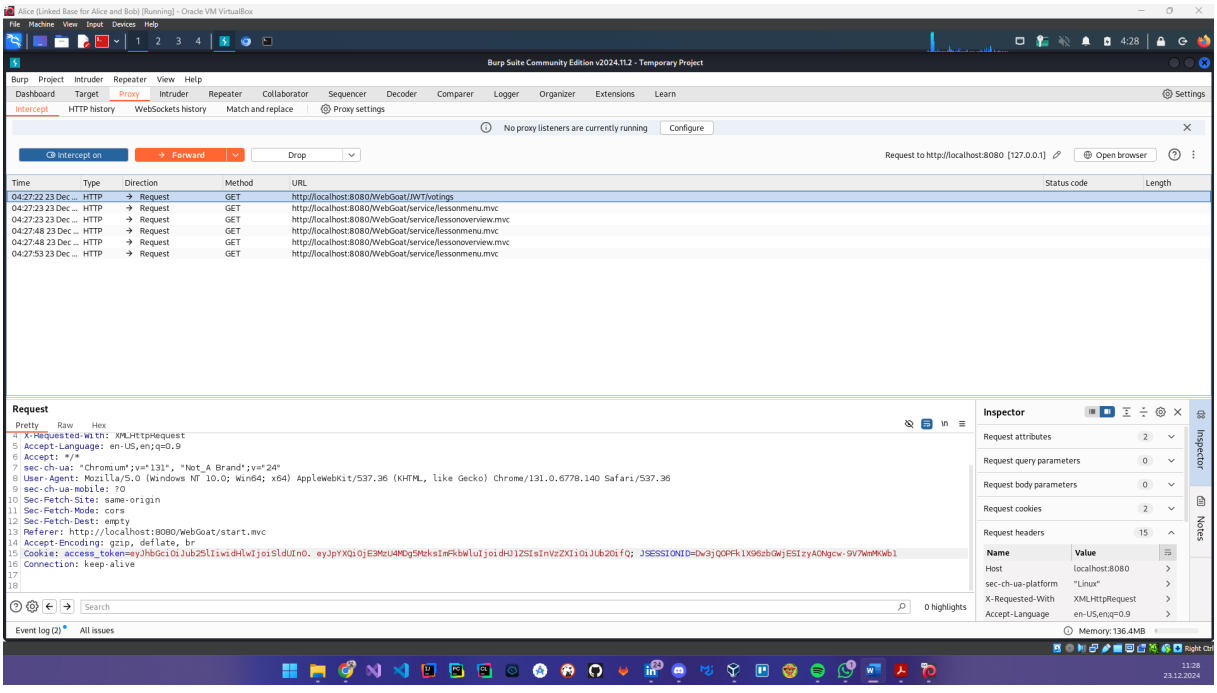
The account is ready and running. The first challenge implies to tamper the JWT and become admin. We switch on Burp to intercept the request made by voting. We can see the JWT here:



We now decode this JWT using https://jwt.io.

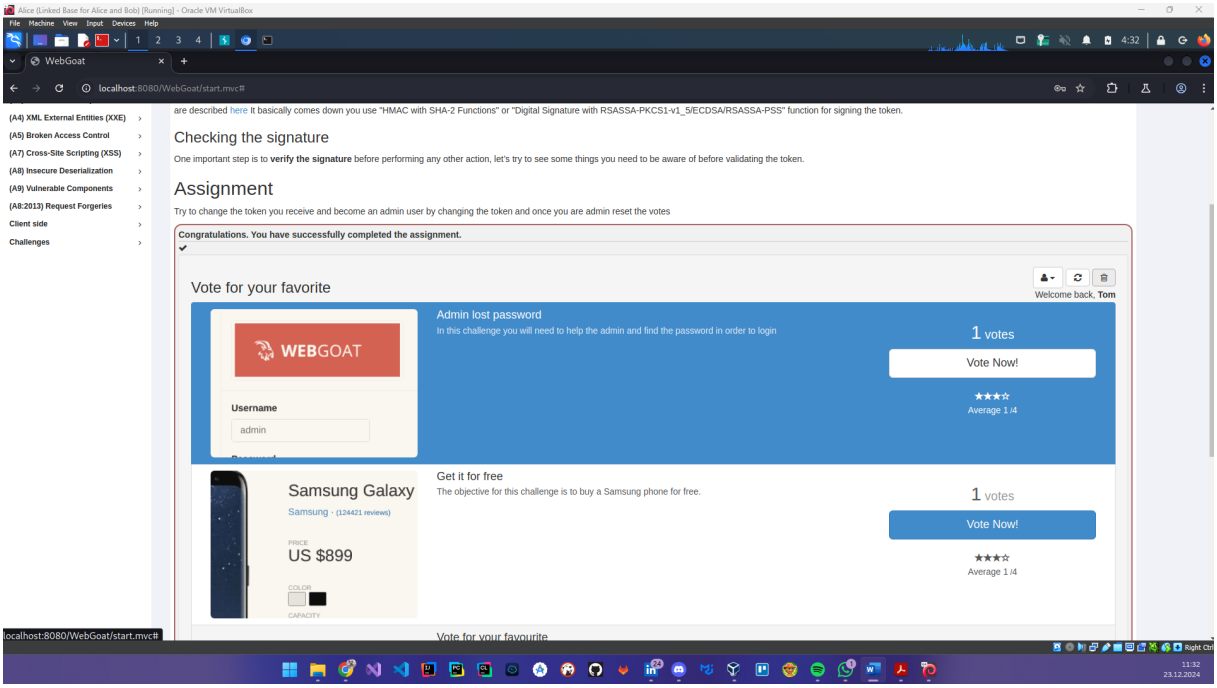


We see that the role is encoded in the JWT so we can easily change that. But in order to bypass the signature we must specify **alg: “none”** so we search for the encoded header. We found that is **eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0**. So our final JWT is encoded as **eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJpYXQiOjE3MzU4MDg5MzksImFkbWluIjoidHJ1ZSIsInVzZXIiOiJUb20ifQ.** We need to pass this to a reset votes request and that’s it. Let’s try it. We need to initiate the reset request and intercept it with Burp.

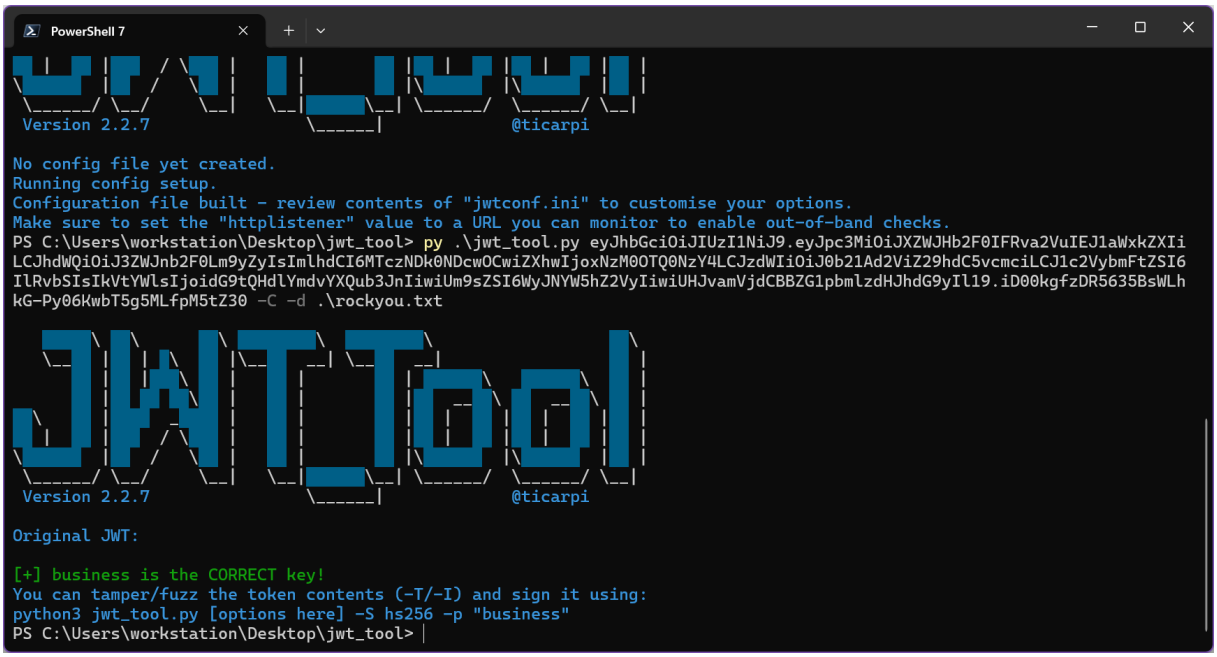


We changed the token then forwarded the request. We need to also take care of the format of the JWT because now it won’t work as long as we

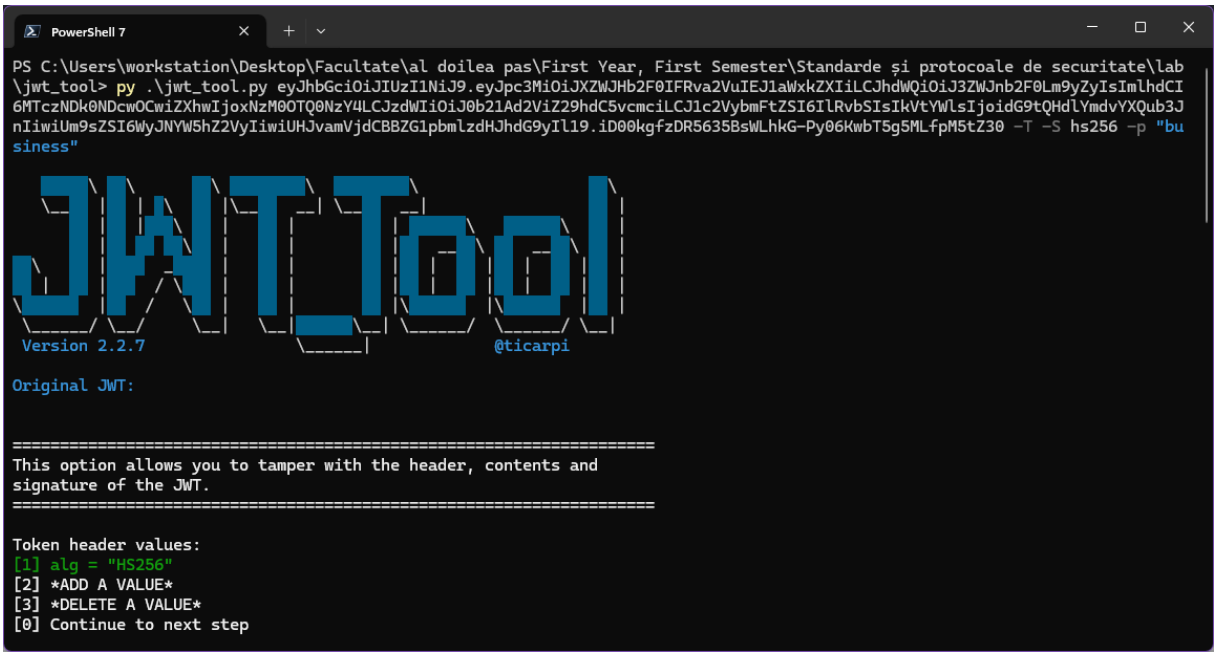
don’t respect the format. We have an extra space and a missing period at the end. After we made the modifications, here we are:

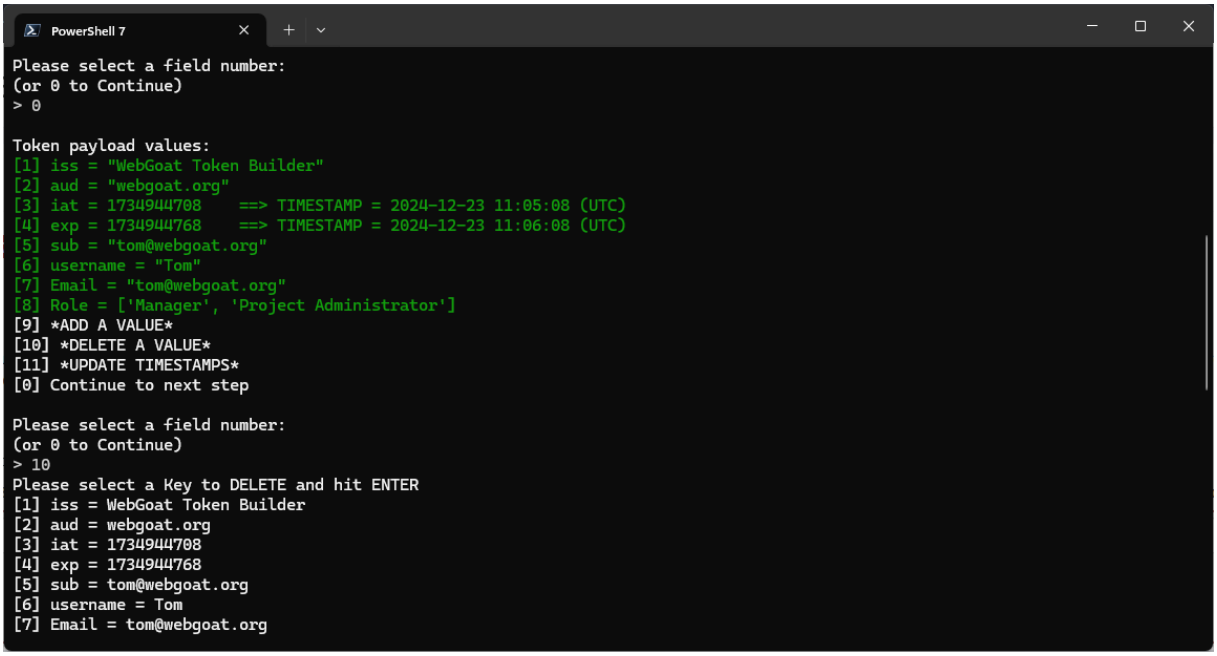


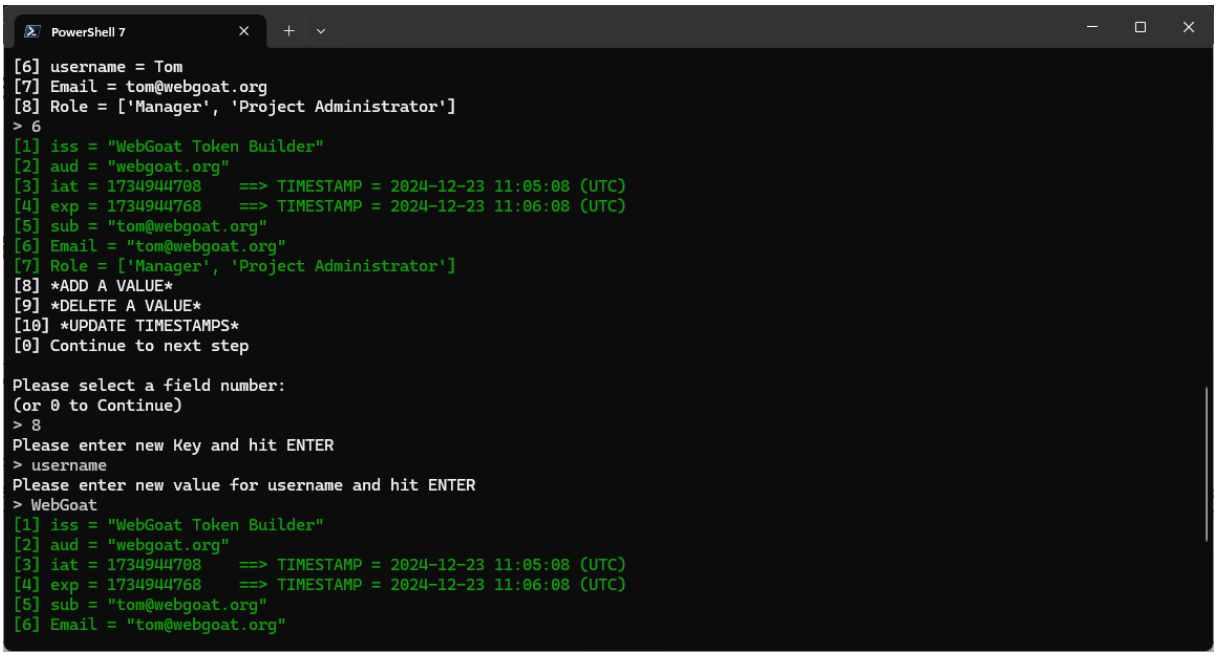
Now let’s proceed into the second challenge. Here we have a compromised JWT with a weak key onto we can try a brute force/dictionary attack. We choose to use jwt_tool with **rockyou** database for doing a dictionary attack. The result is straightforward:

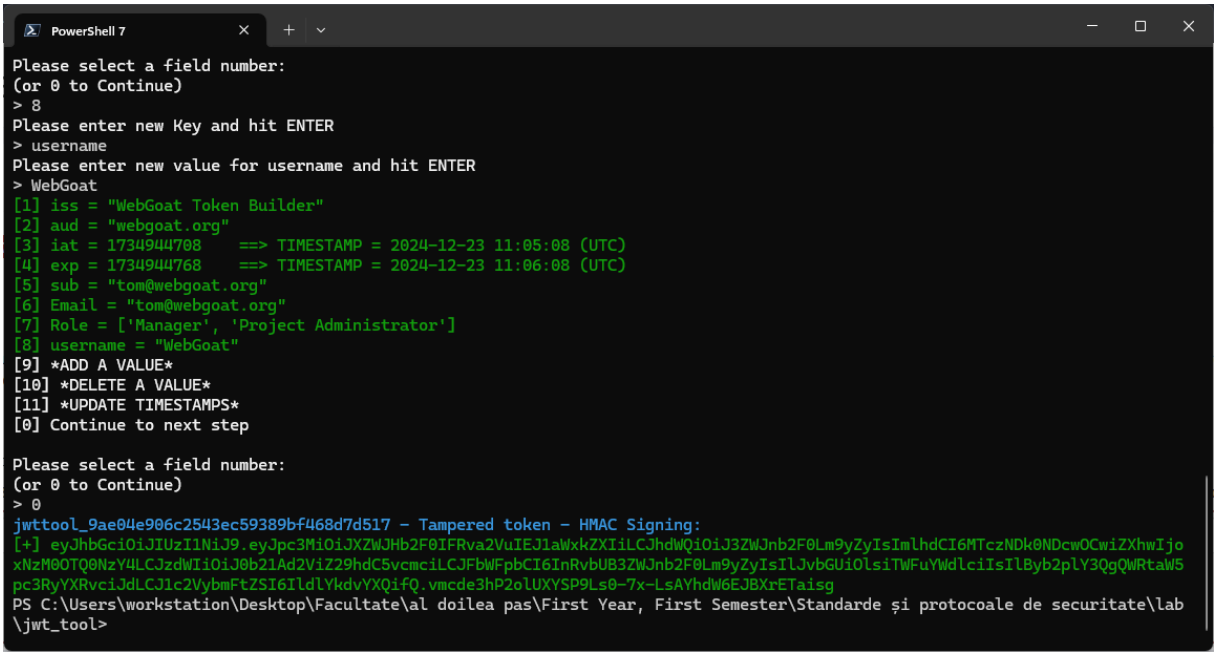


We can use the same tool to tamper the data inside that JWT. We follow the program advice and use the command **py .\jwt_tool.py <JWT> -T -S hs256 -p “business”**. We also must modify the timestamp because if it is expired it won’t be accepted.

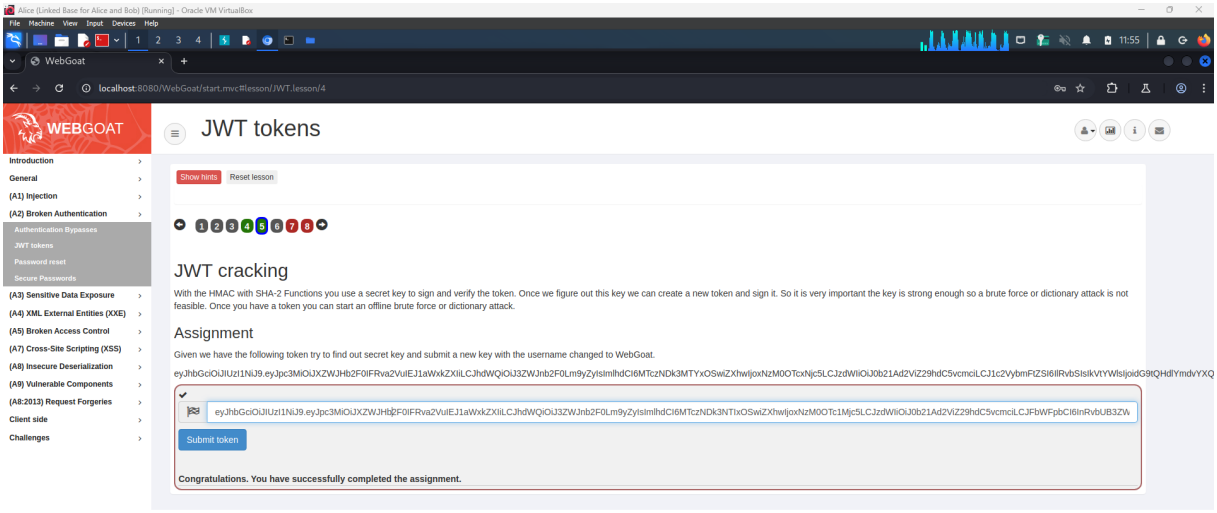

PowerShell 7
PS C:\Users\workstation\Desktop\Facultate\al doilea pas\First Year, First Semester\Standarde și protocoale de securitate\lab\jwt_tool> py .\jwt_tool.py eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJhdWQiOiJ3ZWJnb2F0Lm9yZyIsImlhdCI6MTczNDk0NDcwOCwiZXhwIjoxNzM0OTQ0NzY4LCJzdWIiOiJ0b21Ad2ViZ29hdC5vcmciLCJ1c2VybmFtZSI6IlRvbSIsIkVtYWlsIjoidG9tQHdlYmdvYXQub3JnIiwiUm9sZSI6WyJNYW5hZ2VyIiwiUHJvamVjdCBBZG1pbmlzdHJhdG9yIl19.iD00kgfzDR5635BsWLhkG-Py06KwbT5g5MLfpM5tZ30 -T -S hs256 -p "business"
JWTTool
Version 2.2.7
@ticarpi
Original JWT:
This option allows you to tamper with the header, contents and signature of the JWT.
Token header values:
[1] alg = "HS256"
[2] *ADD A VALUE*
[3] *DELETE A VALUE*
[0] Continue to next step


PowerShell 7
Please select a field number:
(or 0 to Continue)
> 0
Token payload values:
[1] iss = "WebGoat Token Builder"
[2] aud = "webgoat.org"
[3] iat = 1734944708 ==> TIMESTAMP = 2024-12-23 11:05:08 (UTC)
[4] exp = 1734944768 ==> TIMESTAMP = 2024-12-23 11:06:08 (UTC)
[5] sub = "tom@webgoat.org"
[6] username = "Tom"
[7] Email = "tom@webgoat.org"
[8] Role = ['Manager', 'Project Administrator']
[9] *ADD A VALUE*
[10] *DELETE A VALUE*
[11] *UPDATE TIMESTAMPS*
[0] Continue to next step
Please select a field number:
(or 0 to Continue)
> 10
Please select a Key to DELETE and hit ENTER
[1] iss = WebGoat Token Builder
[2] aud = webgoat.org
[3] iat = 1734944708
[4] exp = 1734944768
[5] sub = tom@webgoat.org
[6] username = Tom
[7] Email = tom@webgoat.org

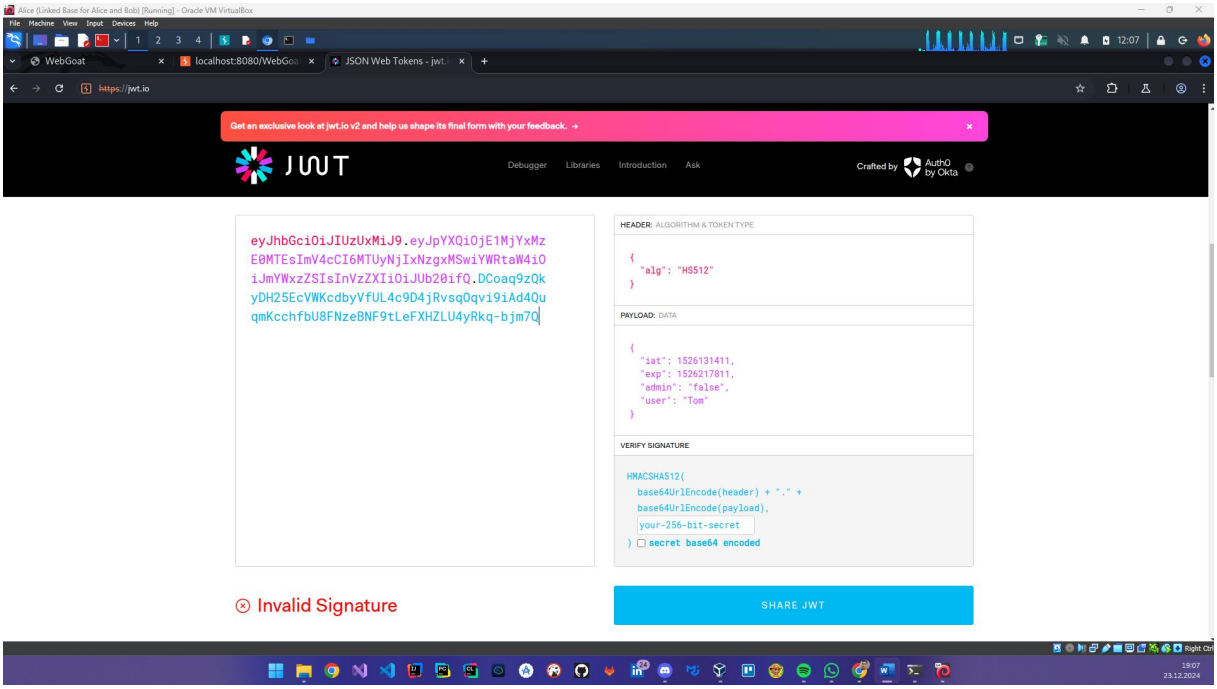




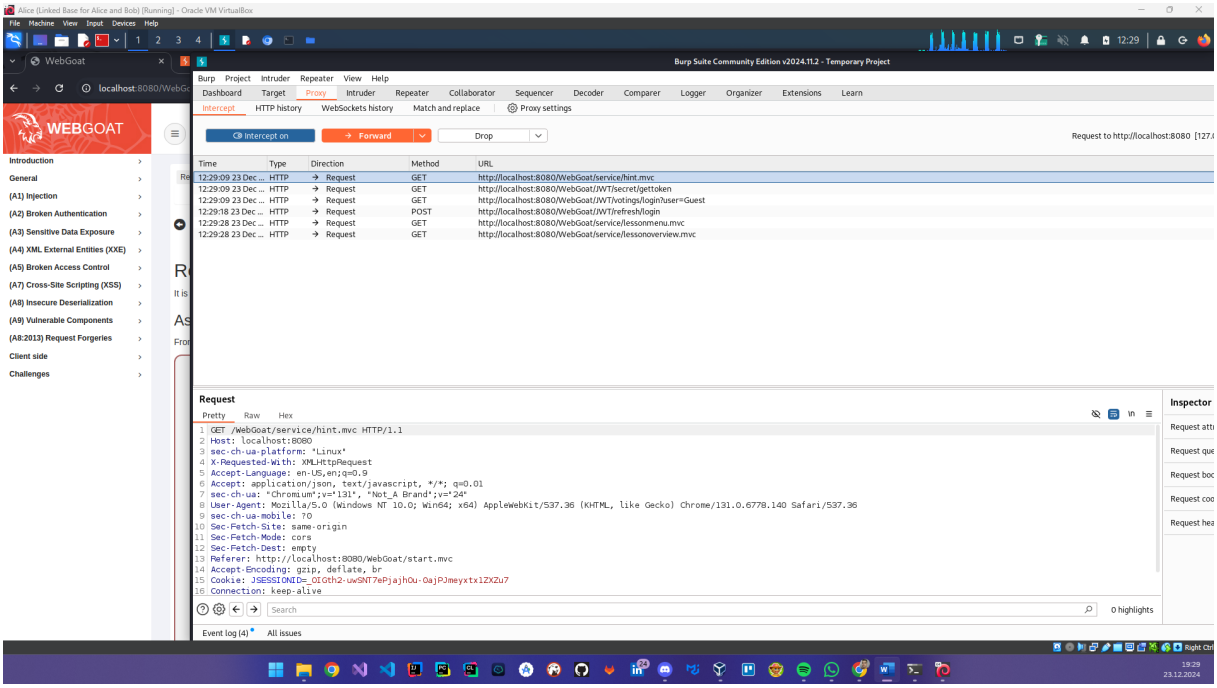
We paste the new JWT into WebGoat. Just realized that the JWT changed (because I had to turn off WebGoat and help mom make **cozonac** 😁) and now the correct password is **washington**. We repeat the same process, and we get another JWT: **eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJhdWQiOiJ3ZWJnb2F0Lm9yZyIsImlhdCI6MTczNDk3NTIxOSwiZXhwIjoxNzM0OTc1Mjc5LCJzdWIiOiJ0b21Ad2ViZ29hdC5vcmciLCJFbWFpbCI6InRvbUB3ZWJnb2F0Lm9yZyIsIlJvbGUiOlsiTWFuYWdlciIsIlByb2plY3QgQWRtaW5pc3RyYXRvciJdLCJ1c2VybmFtZSI6IldlYkdvYXQifQ.fbZ0MNFMz1_Ix9Ln6ggN_h1CEDX25-dgdGeGSLkwd40.** We get the following outcome:



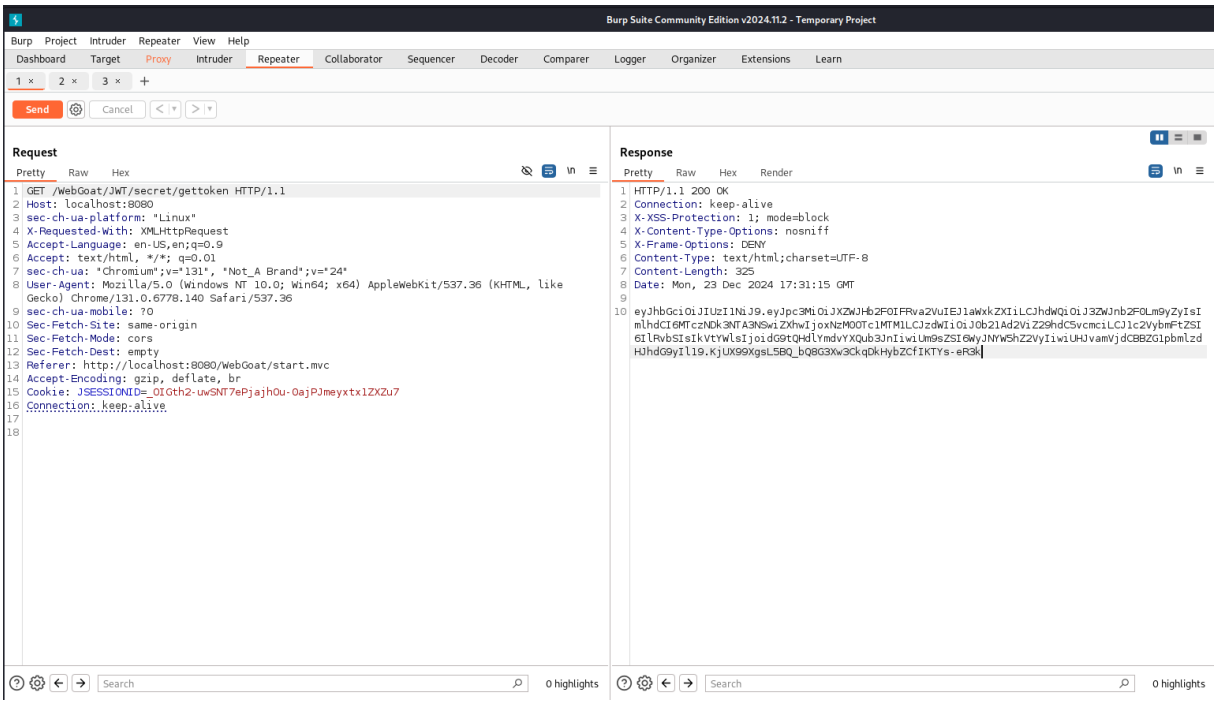
The third challenge is related to refresh tokens. As we can see from the logfile, there is a JWT associated with it. Let’s inspect it.



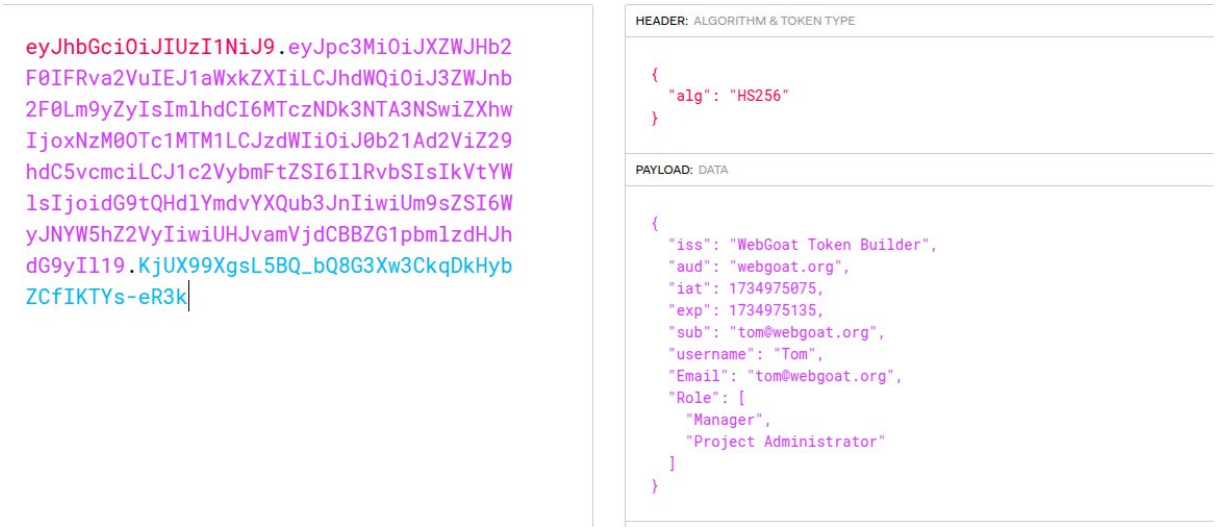
We see that exp field is set to 13 May 2018 09:23:31 GMT-04, so the token is expired. If we can obtain a new token, then we will be able to send it. We can see from the hints that the route for obtaining a new token is **/JWT/refresh/newToken**. Let’s see what requests are made when we refresh the page. There are 3 interesting requests:



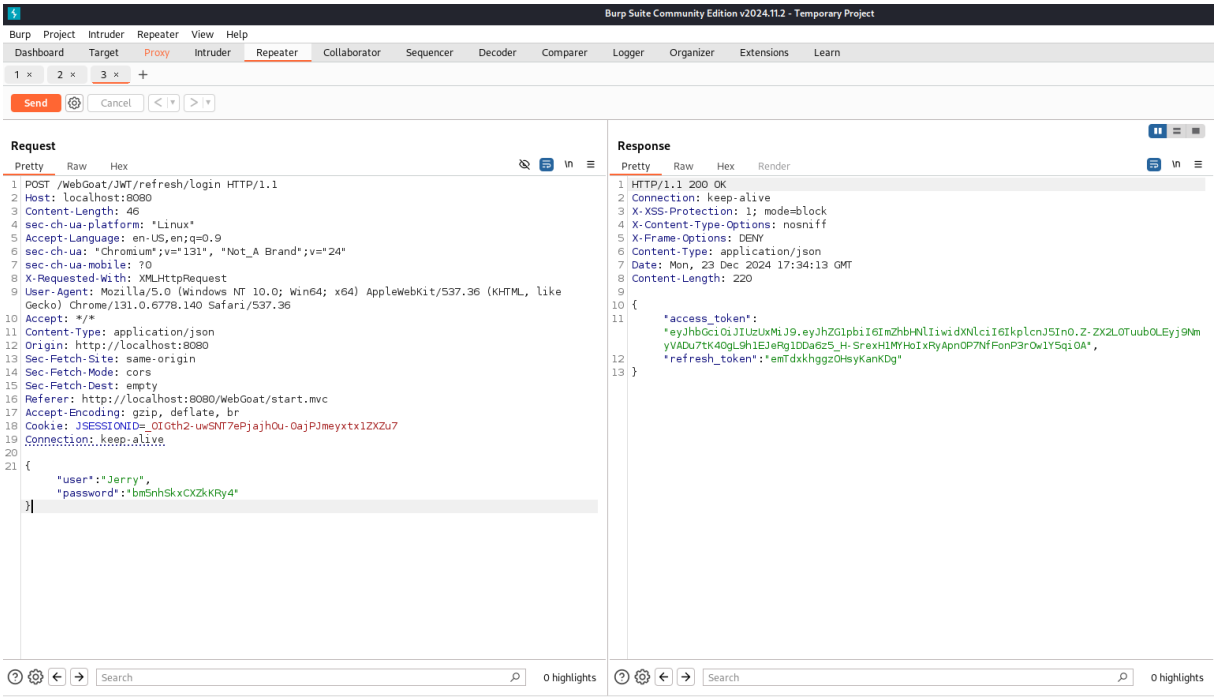
We add them to Repeater. For the first one we get:



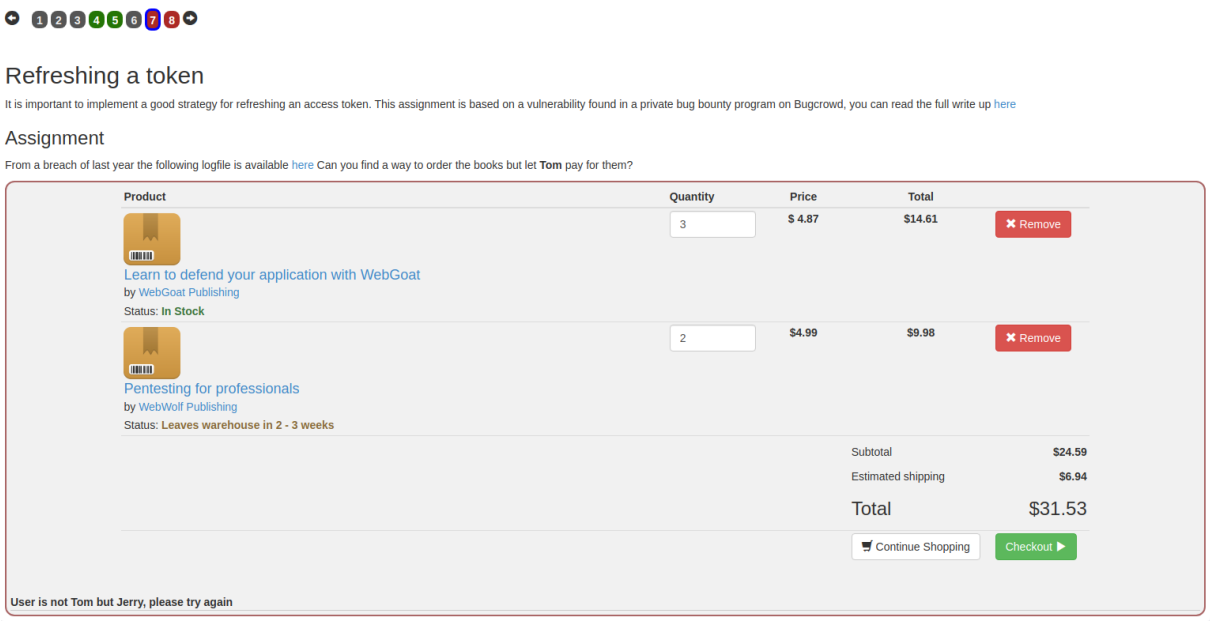
This token is decoded as:



The second one is a login from votings where we can get a JWT. The third one is good but it does not provide a good token:

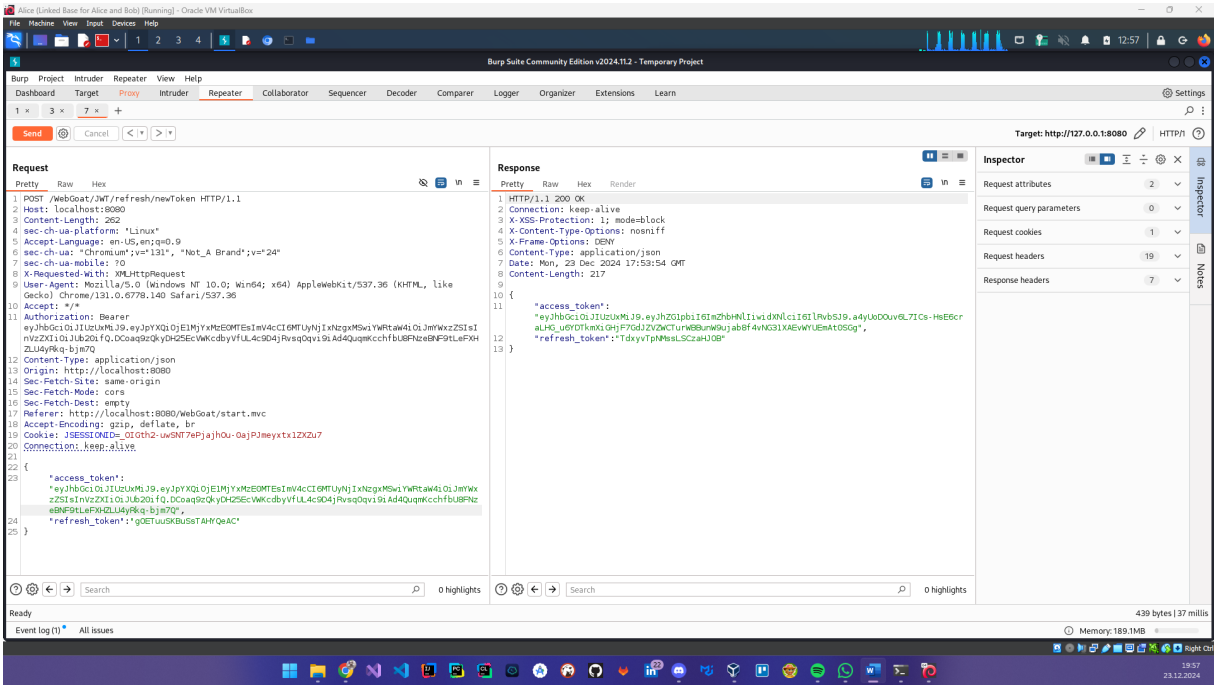


We get that the token is not associated with Jerry instead of Tom.

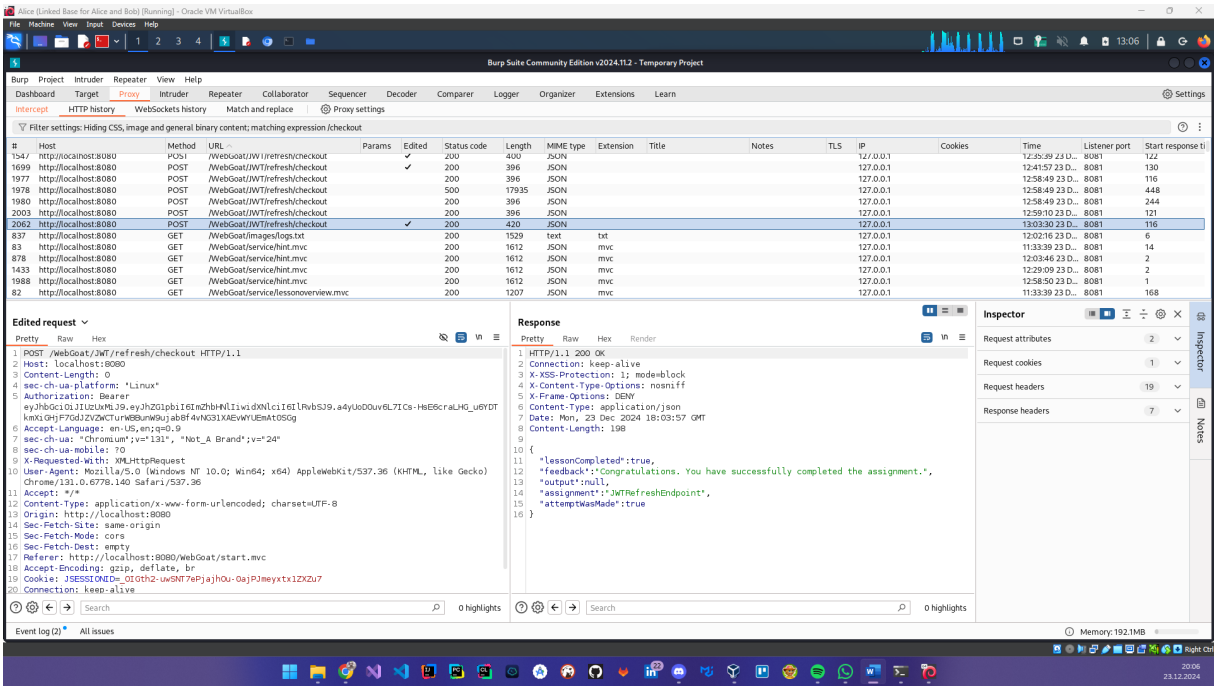


So, we need a token on behalf of Tom. We see that /JWT/secret/gettoken returns an invalid token, but it is from Tom. Let's get Tom a new token based on that.

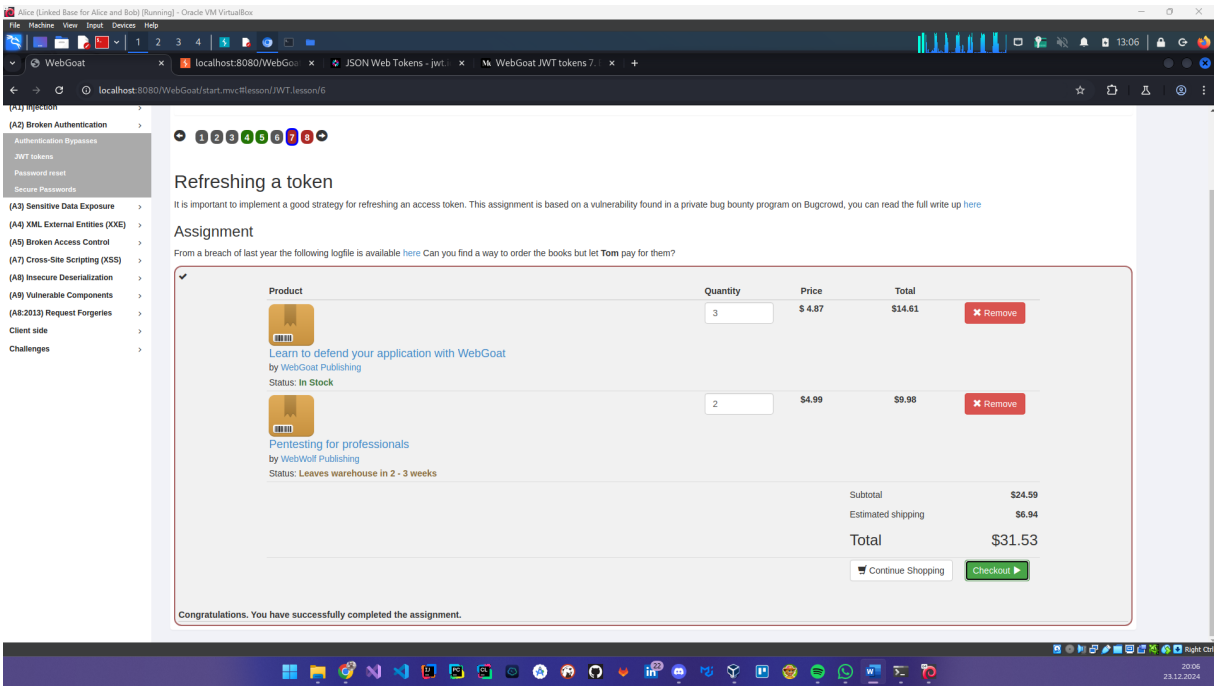
We first try to POST /JWT/refresh/newToken. We get a 401 Unauthorized response, so the Authorization field must be specified. Then we try the request with an empty body – we get a malformed request. We try to populate the body with the same format of **/JWT/refresh/login** as that seems the way to go. We use access_token as Tom's token from the hints and refresh_token as the one get as response from Jerry's login. We get the response:



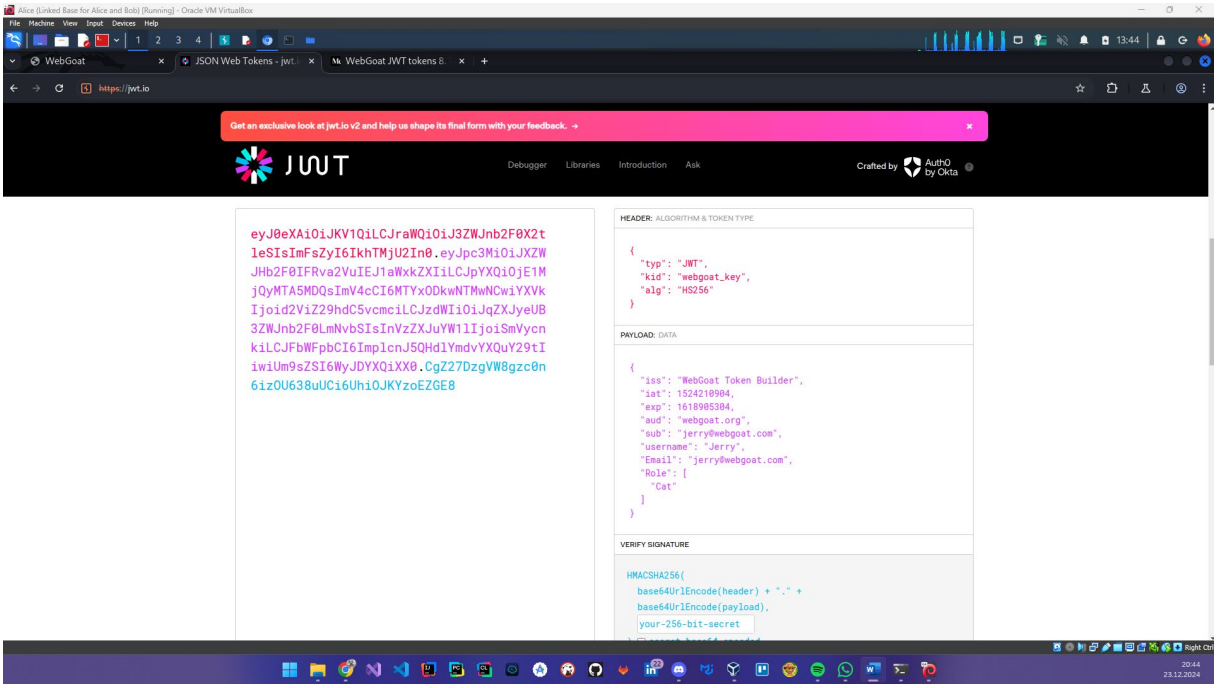
Let’s try this token:



We now finished this challenge.

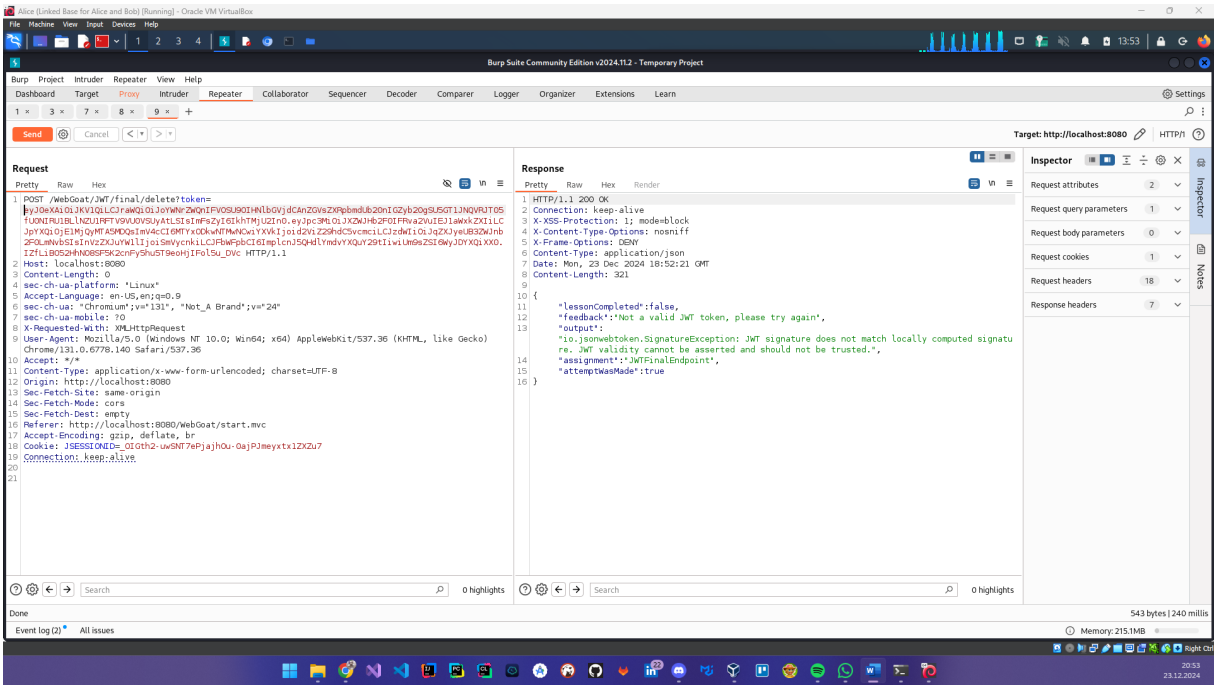


For the last challenge, we can use the previously noted endpoint that gives a Tom’s JWT which is being called when refreshing the page - /JWT/secret/gettoken. We see that its signature does not correspond to the expected signature, so this might not be a good approach. Let’s inspect the token:



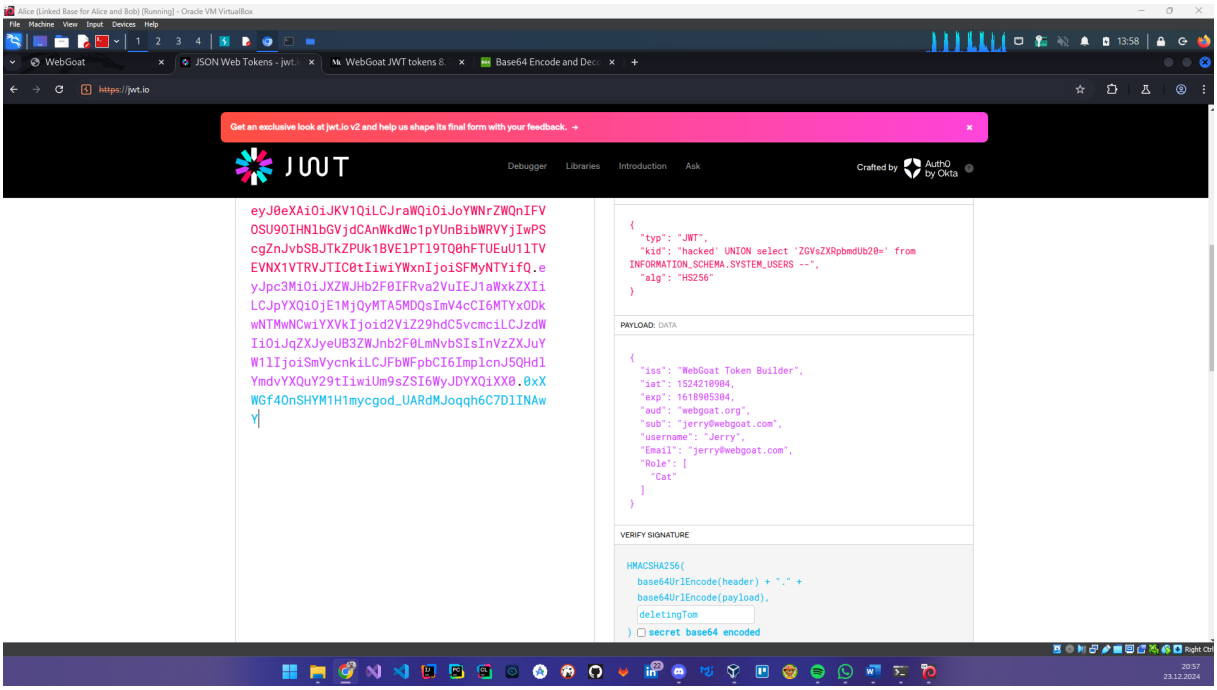
We see that this token was emitted to Jerry and that it has a special attribute in the header: **kid**. This needs to be changed to something as it seems. As the hints unveil, this param should be manipulated using SQL injection so let's put on the recommended hint string: **hacked' UNION select 'deletingTom' from INFORMATION_SCHEMA.SYSTEM_USERS**. Let's try to replicate this request.

The response:

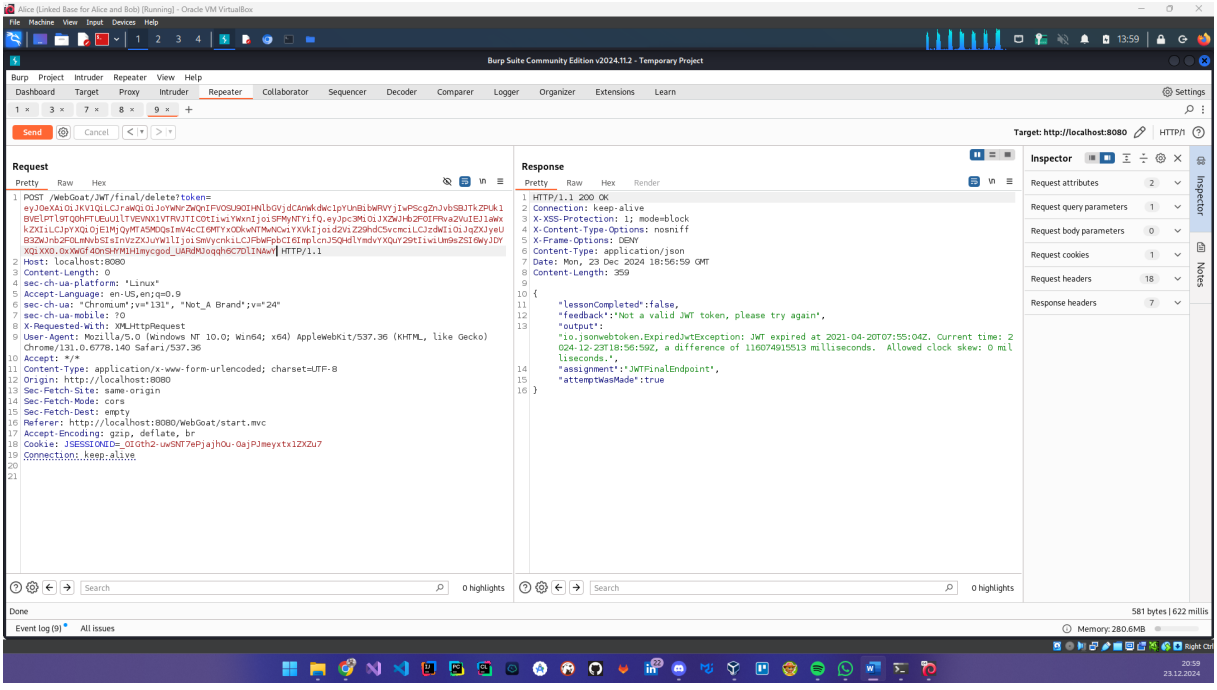


So, we did not get anything because the token is still invalid. Let's dive into the source code and find out that the key must be base64

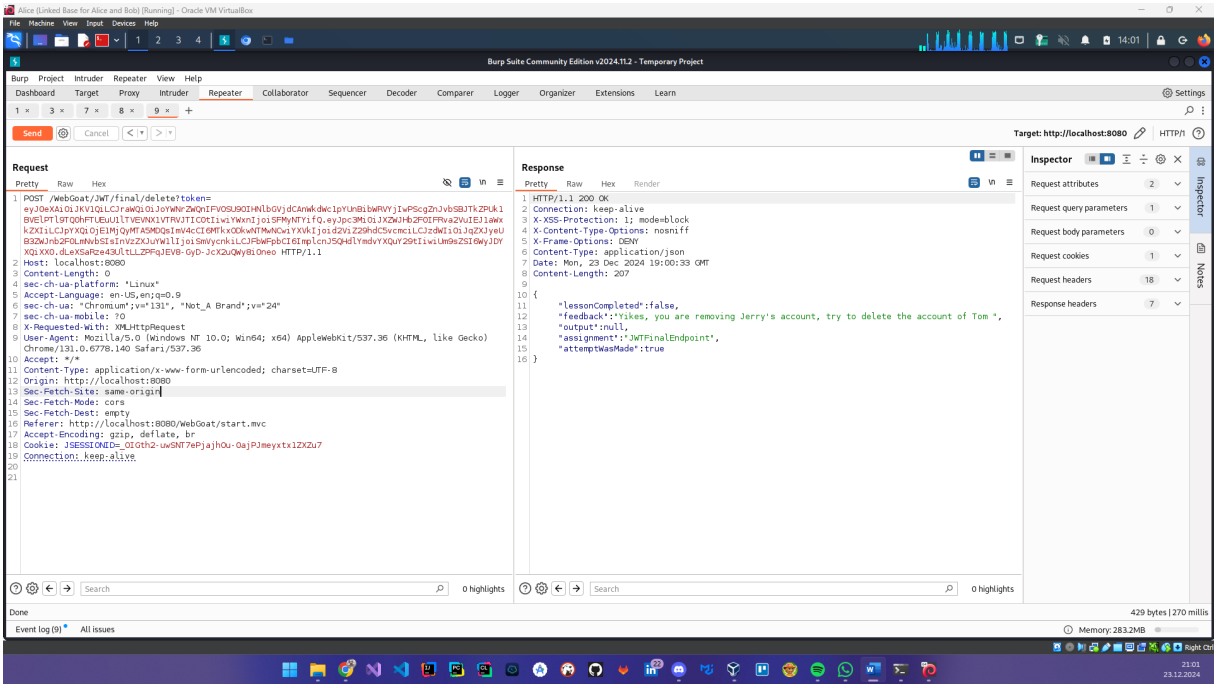
encoded so let’s tamper even more the JWT by signing it with the name of the key in plain and putting the base64 encoded string instead of the plaintext key in the query:



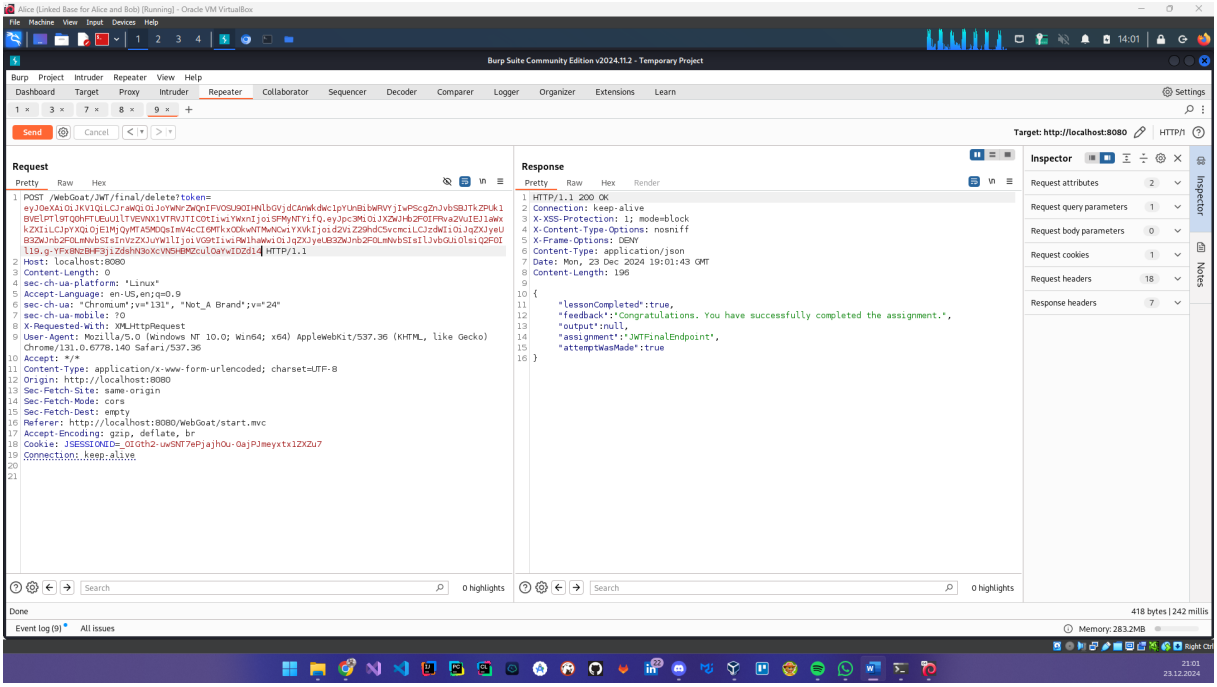
Let’s see the new response from Burp.

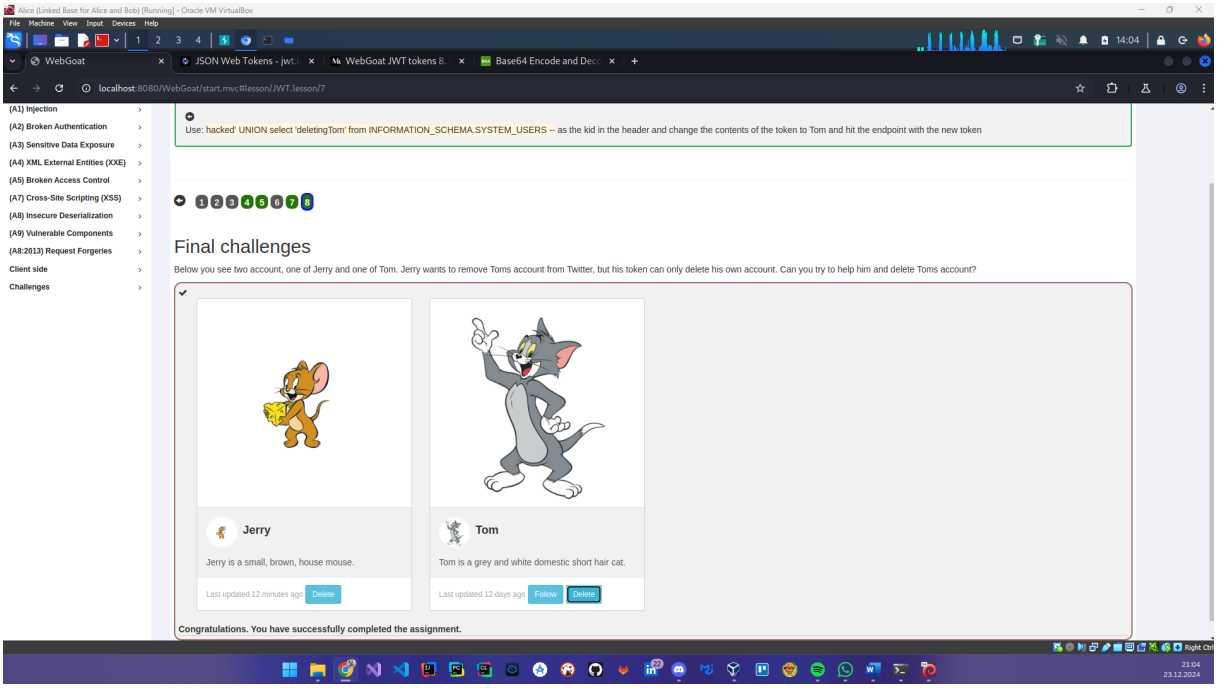


So, the only thing that is needed now is to change the timestamp. This can be easily done.



We forgot to change the username to Tom. Finally:





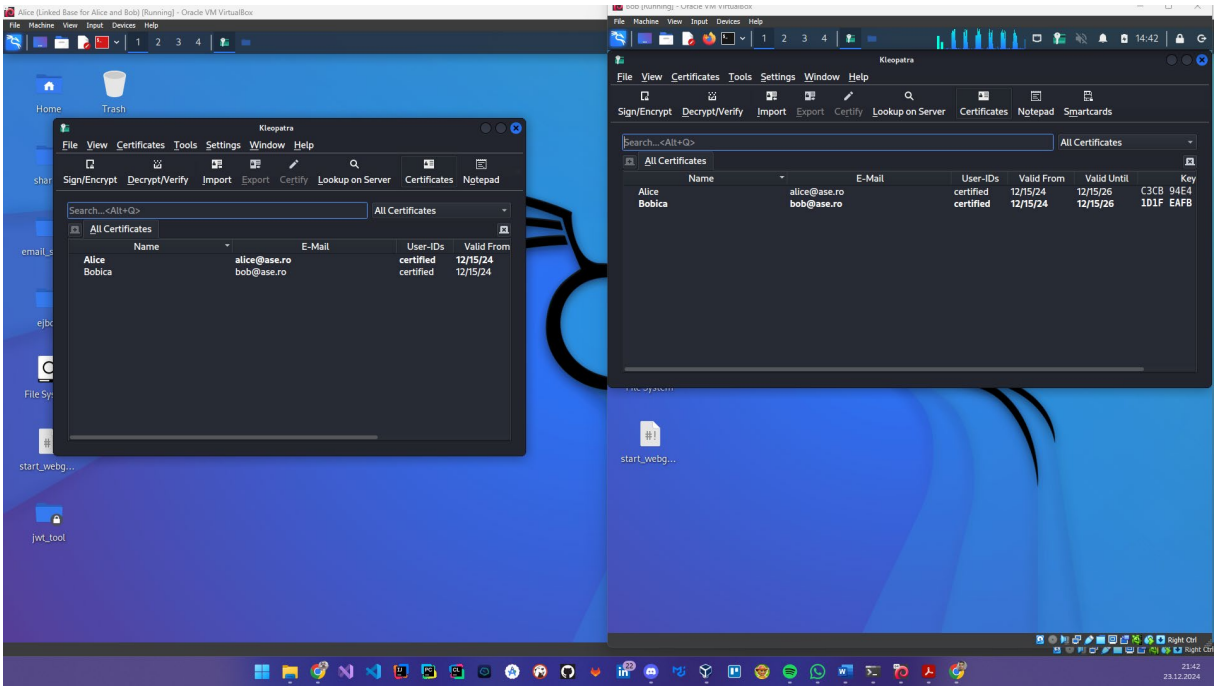
Day 2

PGP

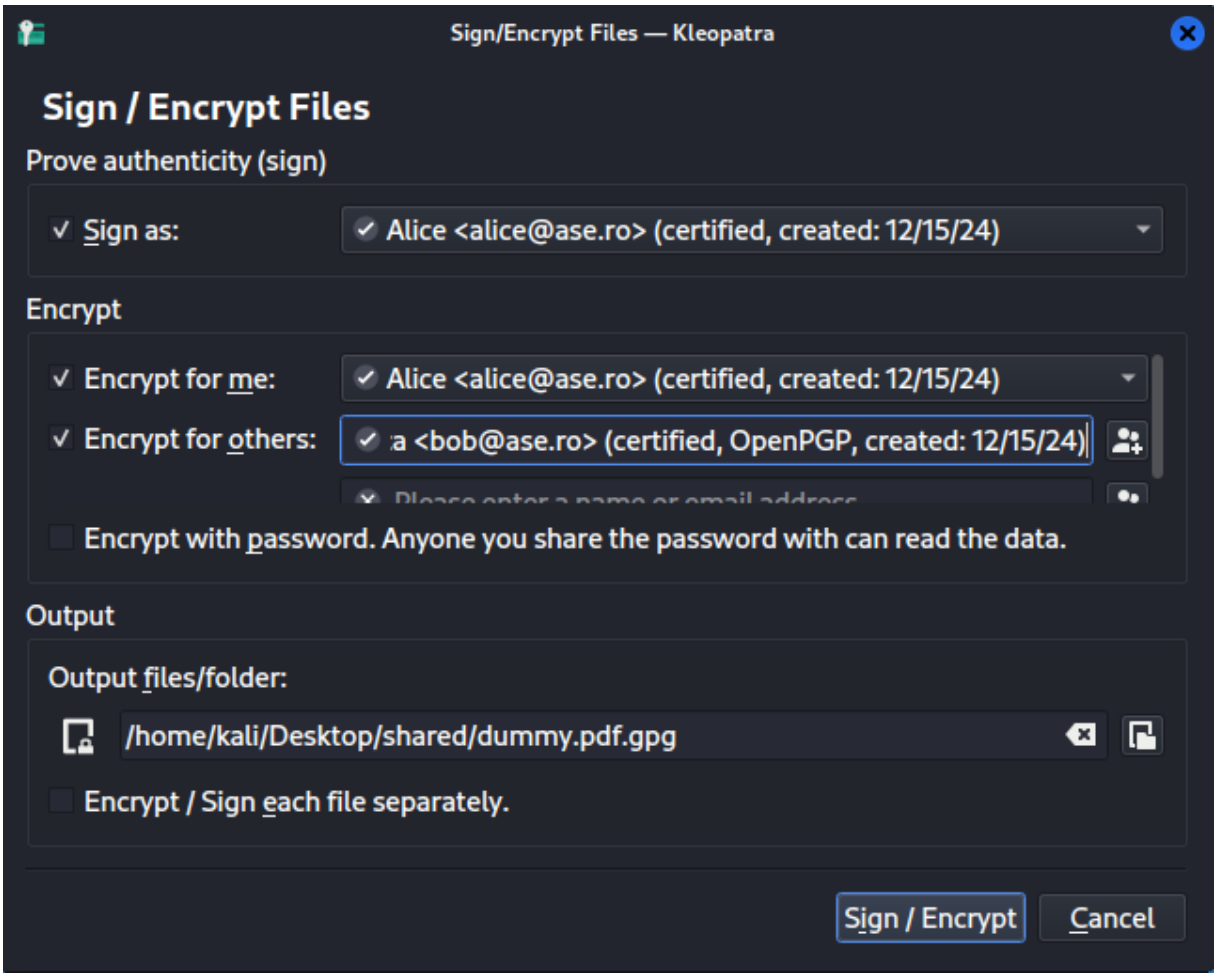
PGP - To do

- Install PGP App
- Generate Key & cert
- Sign
- Encrypt
- Exchange certs to create the Web Of Trust
- Validate a Signature

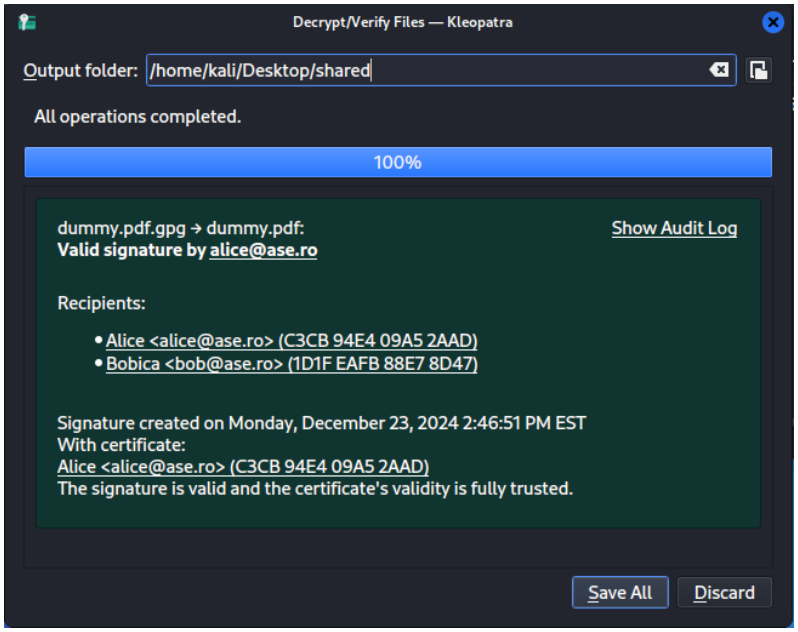
We have installed the Kleopatra PGP App in the live lab. We have generated two keys, and two certificates associated with them on different machines. For each web of trust, Alice, respectively, Bobică are the root of their webs.



In order to encrypt something and also verify its authenticity we exchanged the certificates between the machines in the live lab via the **Import/Export** options and saving the identities on a shared space. Let’s encrypt some dummy document from Alice’s machine and verify if it is trusted by the other party.



We specify that we want to encrypt the file for others too to make them decrypt the content even if they do not trust us. In our case it won't make a difference between we use only two parties. After the encryption is finished, we get a **dummy.pdf.gpg** file. We can decrypt/verify it on Bobică's end.



Everything is valid meaning that our exchange of certificates was done right.

TLS

TLS

- Demo IIS
- Demo apache2
- Useful commands:
  - systemctl restart apache2
  - systemctl stop apache2
  - apache2ctl configtest
- Useful files:
  - /etc/apache2/sites-available/default-ssl.conf
  - /etc/apache2/ssl.crt/
  - /etc/apache2/
  - /etc/apache2/mods-enabled/ssl.conf
  - /var/www/html

We have seen the IIS demo live in the lab. Now moving on to the apache2.

We must create a working apache2 SSL configuration using a certificate generated on our behalf.

Commands used:

```
// starting the apache2 service

┌──(root㉿kali)-[~]

└─# systemctl start apache2


// enabling ssl for apache2

┌──(root㉿kali)-[~]

└─# a2enmod ssl

Considering dependency mime for ssl:
```

```
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled

┌──(root㉿kali)-[~]
└─# ls /etc/apache2/ssl
ls: cannot access '/etc/apache2/ssl': No such file or directory
// creating a dir for our key-pair
┌──(root㉿kali)-[~]
└─# mkdir /etc/apache2/ssl

┌──(root㉿kali)-[~]
└─# cd /etc/apache2/ssl
// generating a key-pair
┌──(root㉿kali)-[/etc/apache2/ssl]
└─# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
server.key -out server.crt
....+...+...+..+......+.+......+........++++++++++++++++++++++++++++
++++++++++*.....+......+..............+....+.........+...+..+.....
.+......+.+...........+...++++++++++++++++++++++++++++++++++++++++*.
+.+..+..............+..................+...+...........+....+..
.........+......+.........+.........+.+..+...+......+......+.......+
..+.+...........+.....+...+...........+.+..+...........++++++

..+...+......+.......+...+.....++++++++++++++++++++++++++++++++++++
++*..........+...+.+...+.....+...+...+.......+........+.......+..+..
.....+........+..........+...+........+.+...++++++++++++++++++++++
+++++++++++++*.+..+.......+..+.+..+.+..+....+...+......+..........
..........+.+...+...........+......+..............+.....+...+.+...
..+....+.........+..+......+...........+.......+.........+..+...+..
....+...+..............+.+.........+..+......+...+....+...+.....+..
.+....+..+...........+.....++++++
```

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:RO

State or Province Name (full name) [Some-State]:Bucharest

Locality Name (eg, city) []:Sector 1

Organization Name (eg, company) [Internet Widgits Pty Ltd]:ASE

Organizational Unit Name (eg, section) []:ISM

Common Name (e.g. server FQDN or YOUR name) []:local

Email Address []:zecheruliviu21@stud.ase.ro

// instructing apache2 to pinpoint to our key-pair instead of the default one

┌──(root㉿kali)-[/etc/apache2/ssl]

└─# nano /etc/apache2/sites-available/default-ssl.conf

// enablind SSL virtual host

┌──(root㉿kali)-[/etc/apache2/ssl]

└─# a2ensite default-ssl.conf

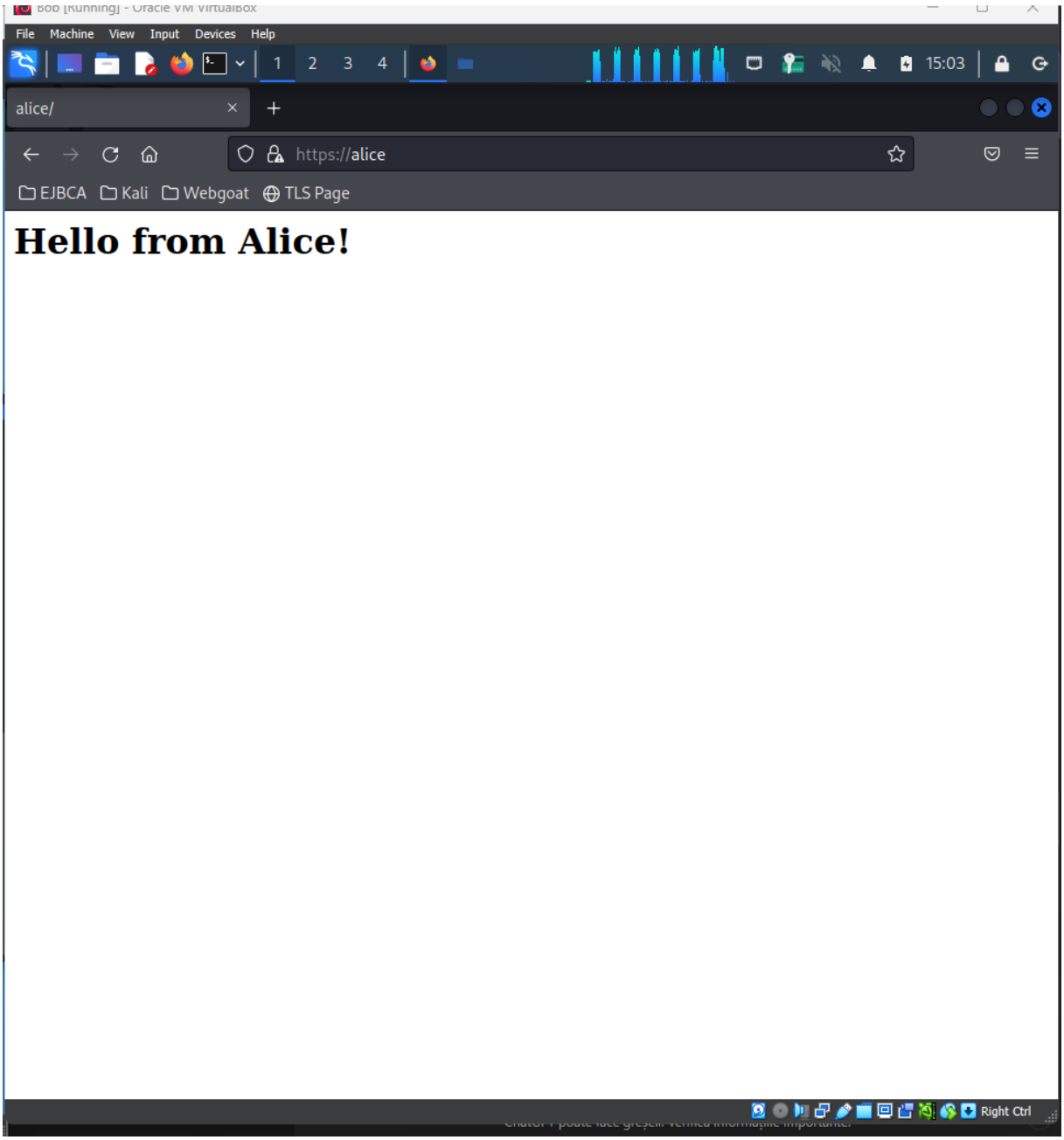
Site default-ssl already enabled

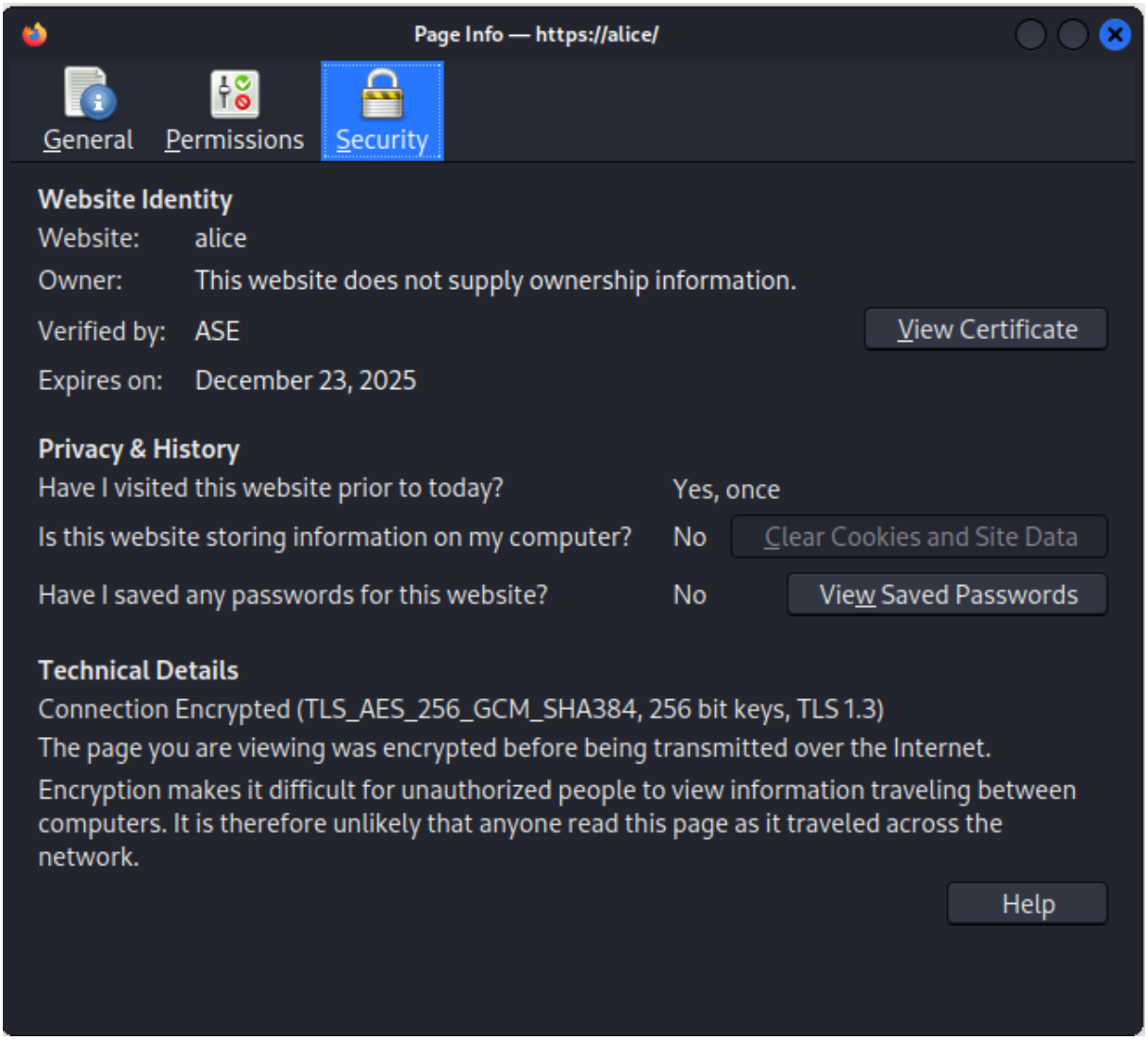
// restarting apache2 to apply modifications

┌──(root㉿kali)-[/etc/apache2/ssl]

└─# systemctl restart apache2

Let’s try to access Alice’s site from Bobică perspective over **HTTPS**. We only get a warning from Mozilla that the certificate is self-generated, but nonetheless our connection is secured.



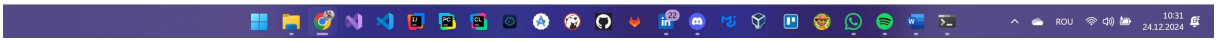


OAuth 2.0

TO DO

- Implement an OAuth 2.0 app
- Firstly, use the attached project
- Configure OAuth 2.0 client in Google Cloud Console
- Follow the instructions in readme
- Implement the same flow using other language (.net, java, etc.)

I will implement a similar app in .NET. We create a simple Razor Pages app. We first register our app into Google Cloud Console interface. Then we build a minimalistic server-side rendering webapp. The result is:



Zeek L.

Digitally signed by Zeek L.
Date: 2024.12.24 11:17:53 +02'00'