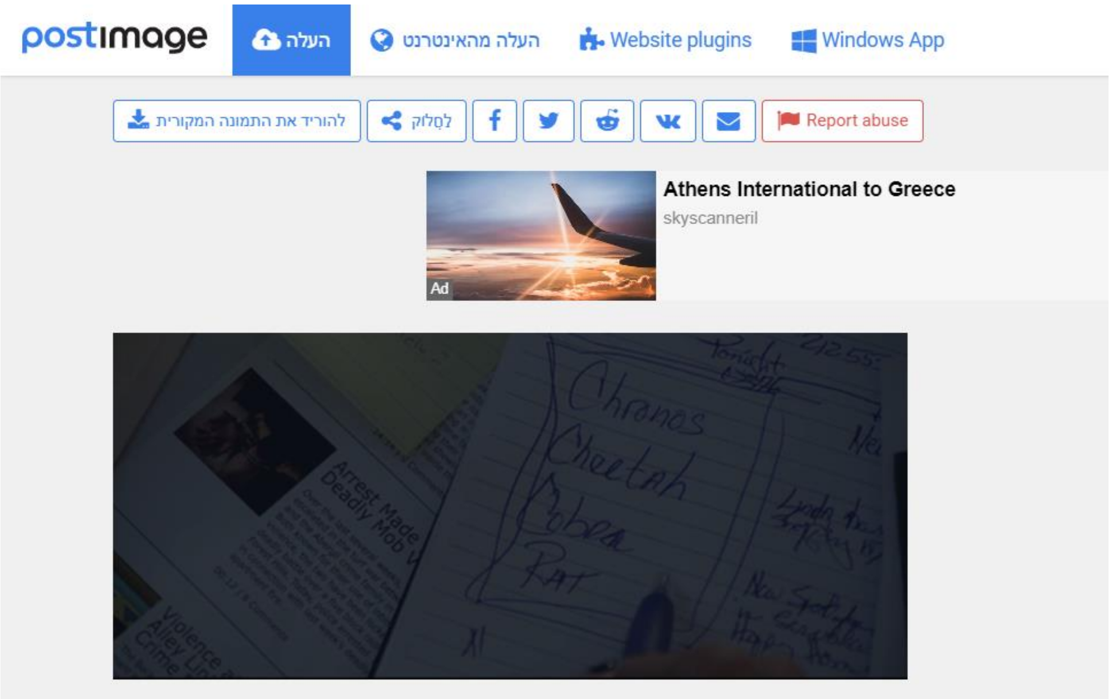
Hit&Run CTF Implementation Writeup

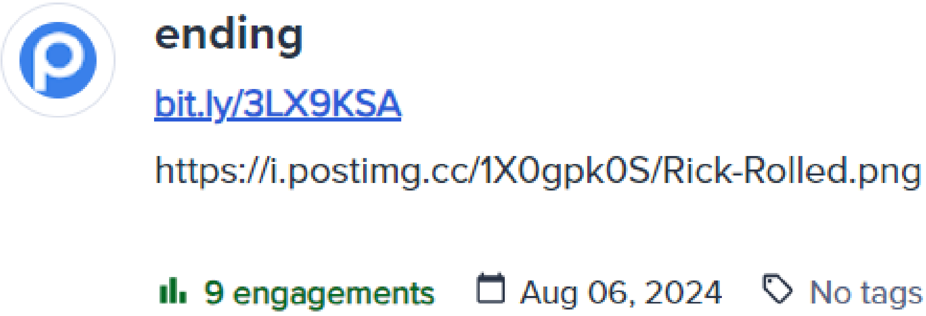
Roey Gross

The process of creating a CTF is reversed to the process of solving it. I took my development offer and started implementing from the last step to the first.

The final clue is an image from within the series and searching online for what's written in the image leads to the end of the investigation. So, no development on this stage.

I uploaded the image of the final clue and created a tiny URL for it.





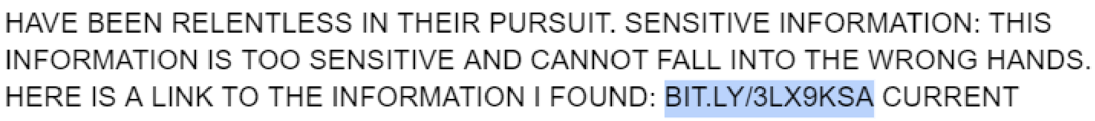
ending

bit.ly/3LX9KSA

https://i.postimg.cc/1X0gpk0S/Rick-Rolled.png

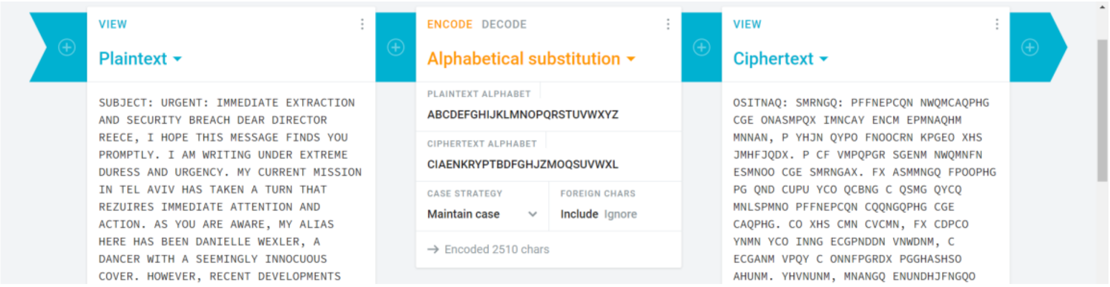
9 engagements Aug 06, 2024 No tags

I created a google docs file, wrote a story and inserted the tiny URL somewhere inside. Here is a snippet from the story:



HAVE BEEN RELENTLESS IN THEIR PURSUIT. SENSITIVE INFORMATION: THIS INFORMATION IS TOO SENSITIVE AND CANNOT FALL INTO THE WRONG HANDS. HERE IS A LINK TO THE INFORMATION I FOUND: BIT.LY/3LX9KSA CURRENT

Then I encrypted the story with substitution cipher while the substitution key was CIAENKRYPT (BDFGHJZMOQSUVWXL):



VIEW

Plaintext

```
SUBJECT: URGENT: IMMEDIATE EXTRACTION
AND SECURITY BREACH DEAR DIRECTOR
REECE, I HOPE THIS MESSAGE FINDS YOU
PROMPTLY. I AM WRITING UNDER EXTREME
DURESS AND URGENCY. MY CURRENT MISSION
IN TEL AVIV HAS TAKEN A TURN THAT
REZUIRES IMMEDIATE ATTENTION AND
ACTION. AS YOU ARE AWARE, MY ALIAS
HERE HAS BEEN DANIELLE WEXLER, A
DANCER WITH A SEEMINGLY INNOCUOUS
COVER. HOWEVER, RECENT DEVELOPMENTS
```

ENCODE DECODE

Alphabetical substitution

PLAINTEXT ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

CIPHERTEXT ALPHABET: CIAENKRYPTBDFGHJZMOQSUVWXL

CASE STRATEGY: Maintain case

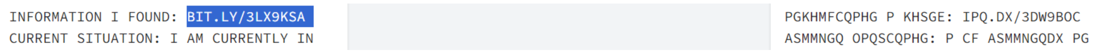
FOREIGN CHARS: Include Ignore

→ Encoded 2510 chars

VIEW

Ciphertext

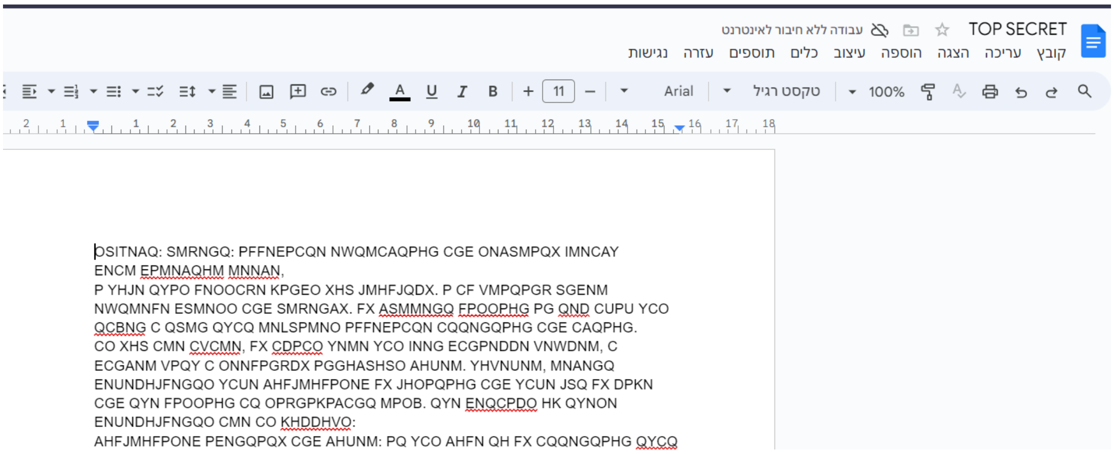
```
OSITNAQ: SMRNGQ: PFFNEPCQN NWQMCAQPHG
CGE ONASMPQX IMNCAY ENCM EPMNAQHM
MNNAN, P YHJN QYPO FNOOCRN KPGEO XHS
JMHFJQDX. P CF VMPQPGR SGENM NWQMNFN
ESMNOO CGE SMRNGAX. FX ASMMNGQ FPOOPHG
PG QND CUPU YCO QCBNG C QSMG QYCQ
MNLSPMNO PFFNEPCQN CQQNGQPHG CGE
CAQPHG. CO XHS CMN CVCMN, FX CDPCO
YNMN YCO INNG ECGPNDDN VNWDNM, C
ECGANM VPQY C ONNFPGRDX PGGHASHSO
AHUNM. YHVNUNM, MNANGQ ENUNDHJFNGQO
```



```
INFORMATION I FOUND: BIT.LY/3LX9KSA
CURRENT SITUATION: I AM CURRENTLY IN
```

```
PGKHMFCQPHG P KHSGE: IPQ.DX/3DW9BOC
ASMMNGQ OPQSCQPHG: P CF ASMMNGQDX PG
```

Then I copied the ciphertext to the docs and called it TOP SECRET.



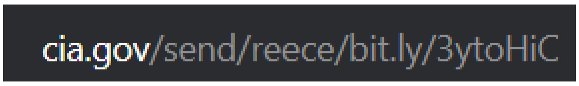
OSITNAQ: SMRNGQ: PFFNEPCQN NWQMCAQPHG CGE ONASMPQX IMNCAY
ENCM EPMNAQHM MNNAN,
P YHJN QYPO FNOOCRN KPGEO XHS JMHFJQDX. P CF VMPQPGR SGENM
NWQMNFN ESMNOO CGE SMRNGAX. FX ASMMNGQ FPOOPHG PG QND CUPU YCO
QCBNG C QSMG QYCQ MNLSPMNO PFFNEPCQN CQQNGQPHG CGE CAQPHG.
CO XHS CMN CVCMN, FX CDPCO YNMN YCO INNG ECGPNDDN VNWDNM, C
ECGANM VPQY C ONNFPGRDX PGGHASHSO AHUNM. YHVNUNM, MNANGQ
ENUNDHJFNGQO YCUN AHFJMHFPONE FX JHOPQPHG CGE YCUN JSQ FX DPKN
CGE QYN FPOOPHG CQ OPRGPKPACGQ MPOB. QYN ENQCPDO HK QYNON
ENUNDHJFNGQO CMN CO KHDDHVO:
AHFJMHFPONE PENGQPQX CGE AHUNM: PQ YCO AHFN QH FX CQQNGQPHG QYCQ

I created a tiny URL for the docs.

The next stage was creating a TLS communication which hides in it the URL to the docs. The TLS communication is encrypted. First let me explain how I created the pcapng file.

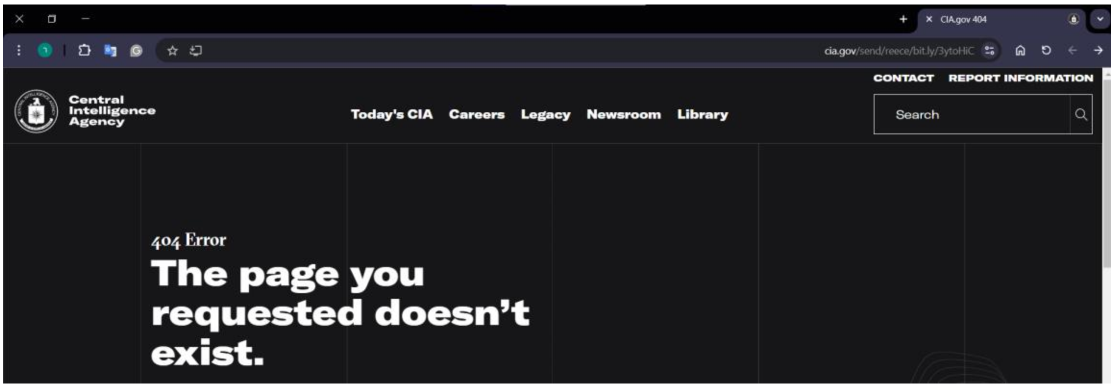
CIA's website uses TLS, so when I browse on chrome any page of CIA, it will use TLS.

I opened chrome browser, started the capture on Wireshark and searched this URL (which contains a link to the docs file).



```
cia.gov/send/reece/bit.ly/3ytoHiC
```

This page of course doesn’t exist.



404 Error

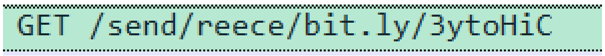
The page you requested doesn’t exist.

Then I stopped the capture.

Chrome browser did several things that has been captured by Wireshark.
The capture contains DNS search for the CIAs IP, then it starts a TCP connection with www.cia.gov. Lastly it uses TLS1.3 protocol to encrypt the communication.

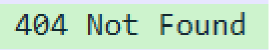
Using the sslkeylog file on wireshark allows to see the encrypted communication (Disscused in the writeup).

The encrypted data is an http request for the page.



```
GET /send/reece/bit.ly/3ytoHiC
```

CIA website sent back:



```
404 Not Found
```

And sent the data of "page not found" page.

What I wanted from this capture is the GET request of the docs url, which is encrypted by TLS1.3.

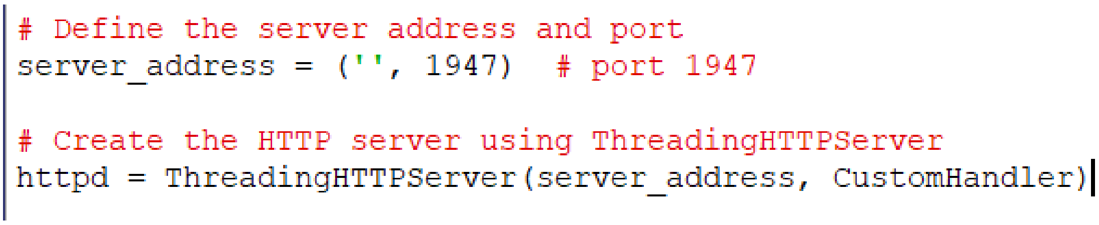
Decrypting can be done by using SSL keys. The solver needs to find the keys and add them in wireshark so the encrypted packets will be decrypted automatically.

I created a file named server.py. I wrote python code that runs a server on local host on port 1947 and listens to input. If the input is keylogfile.log it returns the keys, else, it returns a clue. After developing I noticed that most people do write keylogfile but they forget the .log, so the clue is "File Extensions Matters!".

Explanation of the python code:

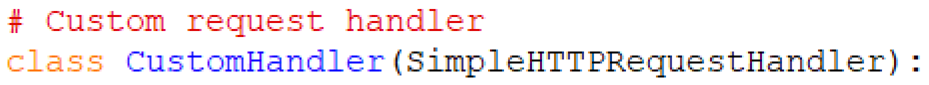
I used a special library for http servers called http.server.

The server is created on local host on port 1947.



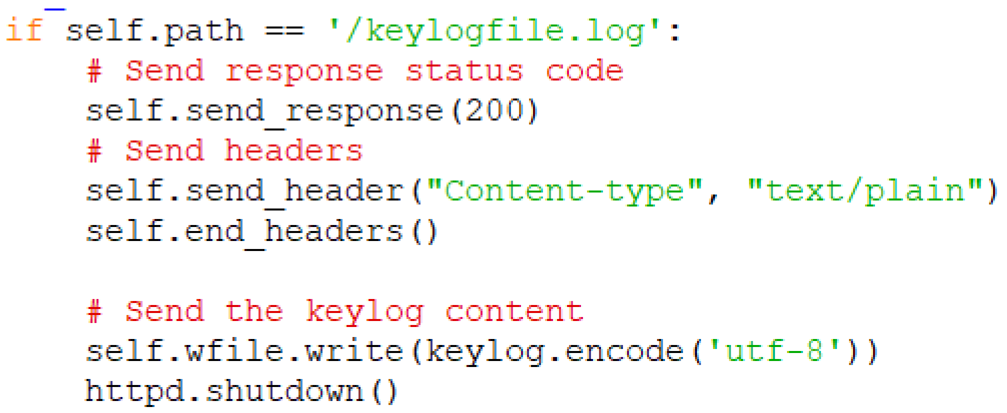
```
# Define the server address and port
server_address = ('', 1947)  # port 1947

# Create the HTTP server using ThreadingHTTPServer
httpd = ThreadingHTTPServer(server_address, CustomHandler)
```

Creating the server requires a handler for handling the client input.



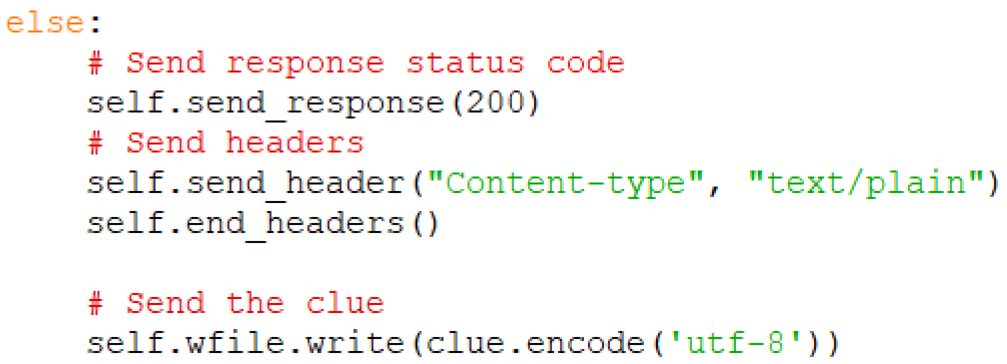
```
# Custom request handler
class CustomHandler(SimpleHTTPRequestHandler):
```

If the user sent keylogfile.log, the server will send back the string of the keylog. Then it will close itself:



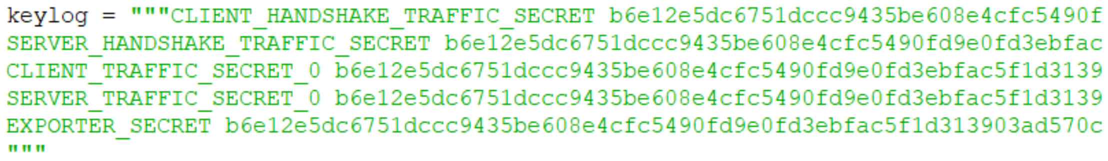
```
if self.path == '/keylogfile.log':
    # Send response status code
    self.send_response(200)
    # Send headers
    self.send_header("Content-type", "text/plain")
    self.end_headers()

    # Send the keylog content
    self.wfile.write(keylog.encode('utf-8'))
    httpd.shutdown()
```

Else, it will send the clue and wait for another input:



```
else:
    # Send response status code
    self.send_response(200)
    # Send headers
    self.send_header("Content-type", "text/plain")
    self.end_headers()

    # Send the clue
    self.wfile.write(clue.encode('utf-8'))
```

The keylog string is:



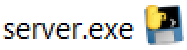
```
keylog = """CLIENT_HANDSHAKE_TRAFFIC_SECRET b6e12e5dc6751dccc9435be608e4cfc5490f
SERVER_HANDSHAKE_TRAFFIC_SECRET b6e12e5dc6751dccc9435be608e4cfc5490fd9e0fd3ebfac
CLIENT_TRAFFIC_SECRET_0 b6e12e5dc6751dccc9435be608e4cfc5490fd9e0fd3ebfac5f1d3139
SERVER_TRAFFIC_SECRET_0 b6e12e5dc6751dccc9435be608e4cfc5490fd9e0fd3ebfac5f1d3139
EXPORTER_SECRET b6e12e5dc6751dccc9435be608e4cfc5490fd9e0fd3ebfac5f1d313903ad570c
"""
```

Its not encrypted, and with some reversing of server.exe the string can be revealed. Anyways, only 1 out of 5 friends of mine tried reversing the exe, most of them had in mind that this is a network ctf and found the right port and sent the right string. I know how to add some anti-reversing but I decided to focus on networks.

Then I compiled the py file to exe file with pyinstaller command:

```
pyinstaller --onefile server.py
```

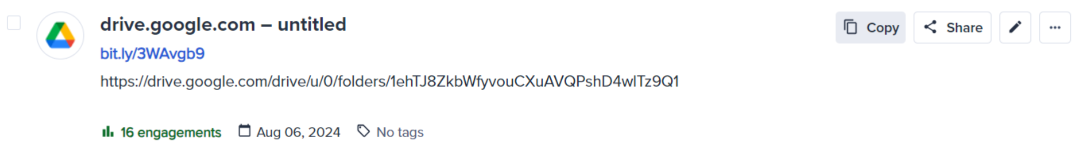
Which created:



server.exe

Now server.exe and the capture are ready. I renamed server.exe to CIA_YEAR.exe. I created a folder on google drive and put over there the exe and the pcapng files. (Google sent me a mail that they have removed the EXE for security reasons, so I zipped it and uploaded it again).

Then I created a tiny url to the folder.



drive.google.com – untitled

bit.ly/3WAvgb9

https://drive.google.com/drive/u/0/folders/1ehTJ8ZkbWfyvouCXuAVQPshD4wlTz9Q1

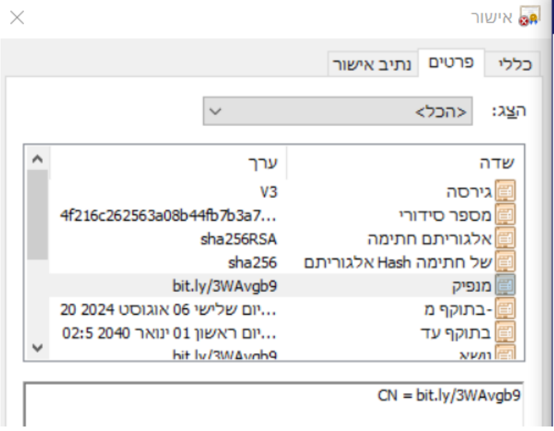
16 engagements Aug 06, 2024 No tags

This url to the folder will be inside a certificate. I created a certificate which contains the url, as the CN, with these commands:

```
C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x64
makecert -sky signature -r -n "CN=bit.ly/3WAvgb9" -pe -a sha256 -len 2048 -ss MY -cy authority RootCert.cer
```

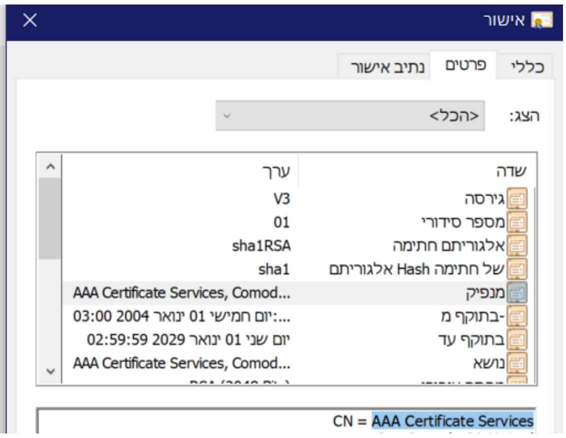
makecert creates a certificate with customized properties. The important flag here is -n which defines the name of the provider. I defined it as the tiny url.

Here is how a real certificate looks like for comparison:



שדה	ערך
גירסה	V3
מספר סידורי	4f216c262563a08b44fb7b3a7...
אלגוריתם חתימה	sha256RSA
אלגוריתם Hash של חתימה	sha256
מנפיק	bit.ly/3WAvgb9
בתוקף מ-	יום שלישי 06 אוגוסט 2024 20...
בתוקף עד	יום ראשון 01 ינואר 2040 02:5...
נושא	bit.ly/3WAvgb9

CN = bit.ly/3WAvgb9

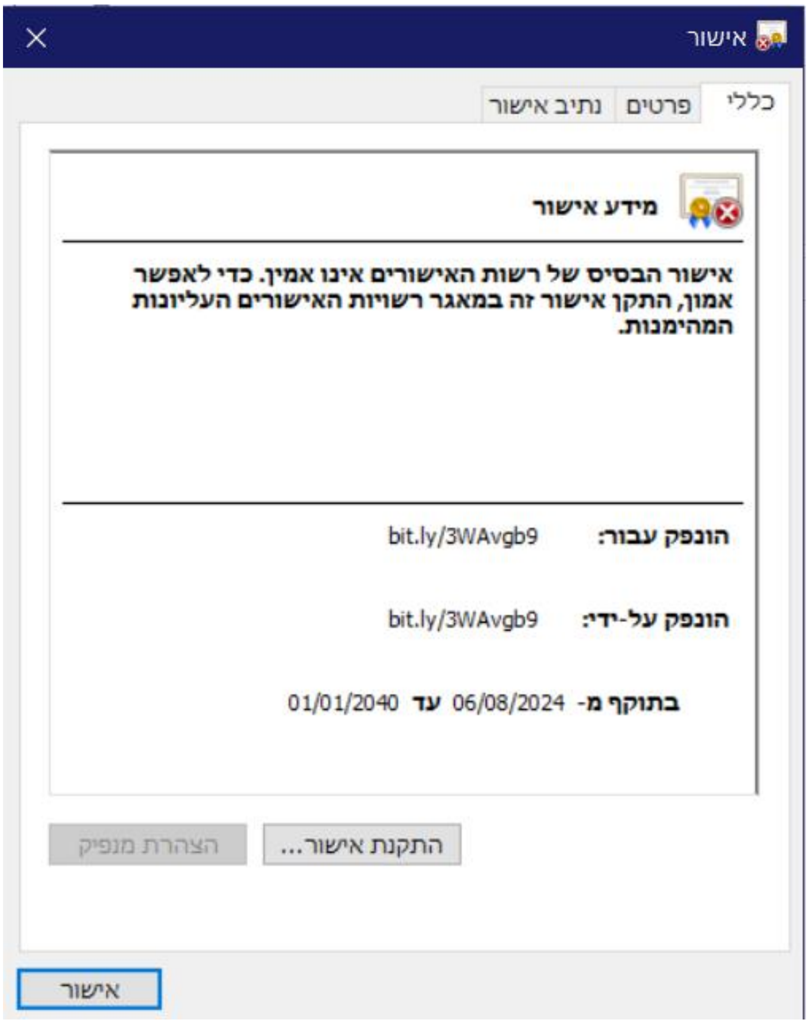


שדה	ערך
גירסה	V3
מספר סידורי	01
אלגוריתם חתימה	sha1RSA
אלגוריתם Hash של חתימה	sha1
מנפיק	AAA Certificate Services, Comod...
בתוקף מ-	יום חמישי 01 ינואר 2004 03:00:...
בתוקף עד	יום שני 01 ינואר 2029 02:59:59
נושא	AAA Certificate Services, Comod...

CN = AAA Certificate Services

The CN really should contain real information about the provider.

There are several fields which can be changed, but I chose CN because its more visible after clicking on the certificate:



מידע אישור

אישור הבסיס של רשות האישורים אינו אמין. כדי לאפשר אמון, התקן אישור זה במאגר רשויות האישורים העליונות המהימנות.

הונפק עבור: bit.ly/3WAvgb9

הונפק על-ידי: bit.ly/3WAvgb9

בתוקף מ- 06/08/2024 עד 01/01/2040

(There is an easier way creating certificates using openssl but I already had this Windows Kit commands from a another project. Download Windows SDK signing tools to use these commands. (Pay attention to the path because it changes according to the version.))

Finally, The solvers get the certificate, where they start helping Segev on his journey of discovring the truth.