CSIT The InfoSecurity Challenge 2020

CSIT is organising The InfoSecurity Challenge (TISC) — a 48-hour challenge which you can attempt anytime between 8 Aug-7 Sep 2020 — filled with cybersecurity and programming questions.

There are prizes to be won:

- 1st place: $3,000
- 2nd place: $2,000
- 3rd place: $1,000
- 4th place: $500
- 5th place: $250
- 6th-10th place: $100

This challenge is open to all Singapore citizens.

For more information, please visit: https://www.tisc.csit-events.sg

Event Details

I started my session at around 9pm, 9/8/2020. I ended up spending way too much time idling, setting things up and not being able to progress (especially on STAGE 2). I gave up about 24 hours in, as I had other work to do (like School). Nevertheless, I still learnt some things, such as practicing how to identify encoding/compression types, UPX Packing Golang, Reverse Engineering using Cutter.

I do not guarantee that the flags I have would be suitable for your submission, nor do I encourage anyone to steal my flags and submit them without understanding of how to get the flag. These writeups are mainly for archival records and for learning purposes

My Stuff

I only could do 2 challenges

1. STAGE 0 (Test Stage): Introduction
   - Do the survey to get the flag `TISC20{finished_the_form_973926492}`
2. STAGE 1: What is this thing
   - I took 6 hours or so.

I tried to do STAGE 2 and even got a RSA Key, but for some reason it didn't work (probably it was the incorrect one)

STAGE 1: What is this thing?

STAGE ONE

Description

You are working as a cyber security analyst for TISC (The InfoSecurity Company). A client has recently approached us after they were struck by ransomware on their database servers!

The last script the sysadmin remembered running was something he took off a stackoverflow page: stackoverflowed.png attached below! This lead to a github page: https://github.com/divoc-91/AnorocWare-Server-Patch We also found some deleted files: Some sort of transactional diagram

We noticed unusual network activity around the time that the user reported being ransomware-d. There were files being sent and received, some of which we were unable to inspect. We suspect they are using a simple password (6 characters, hexadecimal) on the zip files as well as several layers of encoding / compression so it cannot be automatically unpacked by the company's firewall or antivirus. Think of it roughly like zipping a file many times. We need your help to write some code that will get to the bottom of this!

Connect using

```
nc fqybysahpvift1nqtwywevlr7n50zdzp.ctf.sg 31081
```

You will need your SUBMISSION_TOKEN too! (find below)

REMINDER!

- Save any code you wrote and take screenshots of your work, THESE WILL NEED TO BE SUBMITTED IN YOUR WRITEUP FOR YOUR ATTEMPT TO BE CONSIDERED VALID!
- Writeups should be sent from the email address used to sign up for TISC to tisc_contactus@csit.gov.sg within 48h after the end of your 48h session.
- Emails should have "tiscwriteup your_fullname_goes_here" as the subject.
- You should not share your writeups with anyone else or post them publicly before 08 Sept 2020 0000hrs.
- The clarity of your documentation will contribute to a better evaluation of your work.

Attached Files

planBscheme.png

stackoverflowed.png

suspectedscript.png

Solution

Use fcrackzip to bruteforce the password of the zip file and extract out `temp.mess`.

Make a python program that can identify if a file is `gzip`/`xz`/`bzip`/`zlib` compressed, Base64, a regular hexdump, a list of integers. Then either extract it out (for `gzip`/`xz`/`bzip`/`zlib`), convert into raw data (Base64, hexdump), or convert each integer in the list of integers to a character and write all of the characters to a raw file.

Repeat this process recursively on `temp.mess` and you will get the flag

My Thought Process

Cracking Zip

Firstly, I netcat using the given command

```
~$ nc fqybysahpvift1nqtwywevlr7n50zdzp.ctf.sg 31081


$$$$$$$$\ $$$$$$\  $$$$$$\   $$$$$$\
\__$$  __|\_$$  _|$$  __$$\ $$  __$$\
   $$ |     $$ |  $$ /  \__|$$ /  \__|
   $$ |     $$ |  \$$$$$$\  $$ |
   $$ |     $$ |   \____$$\ $$ |
   $$ |     $$ |  $$\   $$ |$$ |  $$\
   $$ |   $$$$$$\ \$$$$$$  |\$$$$$$  |
   \__|   \______| \______/  \______/

CSIT's The Infosecurity Challenge 2020
https://play.tisc.csit-events.sg/

CHALLENGE 1: What is this thing?
======================================

SUBMISSION_TOKEN? PUTINYOURTOKENHERE

We noticed unusually network activity around the time that the user
reported being ransomware-d.
There were files being sent and recieved, some of which we were unable to
inspect.
Could you try to decode this?

Reminder! SAVE ANY CODE YOU WROTE / TAKE SCREENSHOTS OF YOUR WORK, THIS
WILL NEED TO BE SUBMITTED IN YOUR WRITEUP!
CLARITY OF DOCUMENTATION WILL CONTRIBUTE TO A BETTER EVALUATION OF YOUR
WRITEUP.

The file is hosted at
http://fqybysahpvift1nqtwywevlr7n50zdzp.ctf.sg:31080/04979fdc74cffb769440fd
8c69a7c5c0.zip .


Flag?
```

After that I downloaded the zip file and found a temp.mess file. However, extracting it seems to require a password. Since the challenge said the password was 6 characters and hexadecimal, I used fcrackzip to get the password. It took 20 min or so?

```
$ fcrackzip -b -u -v -l 6 -c ':0123456789ABCDEF'
04979fdc74cffb769440fd8c69a7c5c0.zip
found file 'temp.mess', (size cp/uc 245827/245740, flags 3, chk a499)
$ fcrackzip -b -u -v -l 6 -c ':0123456789abcdef'
04979fdc74cffb769440fd8c69a7c5c0.zip   found file 'temp.mess', (size cp/uc
245827/245740, flags 3, chk a499)
checking pw 98967f

PASSWORD FOUND!!!!: pw == 9a035a
```

Basic analysis of `temp.mess` show that it is gzip compressed data

```
$ file temp.mess
temp.mess: gzip compressed data, last modified: Wed Aug  5 11:28:11 2020,
max compression, original size modulo 2^32 245647
$ binwalk temp.mess

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------
-----
0             0x0             gzip compressed data, maximum compression,
last modified: 2020-08-05 11:28:11
15            0xF             gzip compressed data, maximum compression,
last modified: 2020-08-05 11:28:11

$ mv temp.mess temp.gz
$ gunzip -d temp.gz
$
```

On extracting it seems like you get another gzip file. So I extracted that

```
$ file temp
temp: gzip compressed data, last modified: Wed Aug  5 11:28:11 2020, max
compression, original size modulo 2^32 245554
$ mv temp temp.gz && gunzip -d temp.gz
$ file temp
temp: zlib compressed data
```

Extracting zlib data

To extract the zlib data, I used zlib-flate (`sudo apt install qpdf` first to use it)

```
$ zlib-flate -uncompress < temp | cat > tempdata.txt
```

Extracting hexdump and list of integers (JSON Data)

Opening the text file makes it seem like base64 data. However, decoding it doesn't seem to give any useful data

```
$ base64 -d tempdata.txt > tempdata-b64decoded
$ file tempdata-b64decoded
tempdata-b64decoded: data
$ binwalk tempdata-b64decoded

DECIMAL       HEXADECIMAL     DESCRIPTION
----------------------------------------------------------------------------
-----
```

Another possible form of data is Hex. I coded a simple python program to convert the hex string into raw byte data

```
def reverseHexDump(filename, outFile = "temp"):
    with open(filename) as f:
        data = f.read()

    # Read 2 characters at a time
    intData = []
    charAtTime = 2
    for i in range(0,len(data),charAtTime):
        intData.append(int( data[i:i+charAtTime] ,16))

    byteData = bytes(intData)
    with open(outFile,'wb') as f:
        f.write(byteData)

reverseHexDump('tempdata.txt')
```

```
$ file temp
temp: JSON data
```

Moreover, it's actually a list of integers. On converting them to ASCII, I get garbage text, so I put this text into a new file with another python code

```
def readInJson(file, out):
    with open(file) as f:
        data = f.read()
    processedData = eval(data)

    strData = ""
    for i in processedData:
        strData += chr(i)
    with open(out,'w') as f:
        f.write(strData)
readInJson('temp','temp')
```

Extrcating `xz`

After that extract it out again

```
$ file temp
temp: XZ compressed data
$ mv temp temp.xz
$ xz -d temp.xz
$ file temp
temp: gzip compressed data, last modified: Wed Aug  5 11:28:11 2020, max
compression, original size modulo 2^32 215096
```

After that I extracted it with `gunzip` and `xz` again to get a text file, with a bunch of integers this time

```
$ file temp
temp: ASCII text, with very long lines, with no line terminators
$ head -c 100 temp
3738396338346237363535303163334656630323638636262383465306165633132316238356
237303737373737373730626236
```

I guessed that these integers could mean more byte data so I converted them into byte data and wrote the bytes into raw file

```
def readInIntegers(file,out):
    with open(file) as f:
        data = f.read()
    intData = int(data)
    #print(intData)
    byteData = bytes([intData])
    with open(out,'wb') as f:
        f.write(byteData)
readInIntegers('temp', 'temp1')
```

```
$ file temp1
temp1: ASCII text, with very long lines, with no line terminators
$ head -c20 temp1
[3738396338346237363
$ tail -c20 temp1
306533333239353537L]
$
```

Ok it's just the same number. After that I thought that this may just be hex data again, and converted it into raw byte data again (and that was JSON so I converted the strings inside and so on) After that there is way more tedious extraction.

Some other file types

After going through several times of extraction, there were some new file formats I figured out

One of them was bzip. I used `dtrx` to extract it (because I'm lazy)

```
$ file tempbzip
tempbzip: bzip2 compressed data, block size = 900k
```

Here's how to install `dtrx` by the way.

```
wget http://ftp.br.debian.org/debian/pool/main/d/dtrx/dtrx_7.1-2_all.deb
sudo dpkg -i dtrx_7.1-2_all.deb
sudo apt update
sudo apt install -f
```

Another type I was decoding (and struggling to do so) is a type of string with only alphanumeric characters.Then I realised it is Base64

```
$ cat temp1 | grep -o -P '.{0,3}[^0-9A-Za-z].{0,4}'
$ head -c20 temp1
UWxwb09URkJXU1pUV1V4
$ tail -c20 temp1
Ulwd29TQ1kwcklDQT0K
$ cat temp1 | base64 -d | base64 -d > tempbzip
$
```

Scripting

I started getting bored of doing things manually. Also I don't think thats what they wanted. I decided to start scripting the automatic extraction of files. There was probably an online extractor online that could do this for you, but ~I'm not very smart~ I wanted to challenge myself.

After that I let the program run until it crashes correctly, and I can get the flag

```
$ python solve.py
...
an integer is required
Traceback (most recent call last):
  File "solve.py", line 135, in <module>
    scriptRun('test')
  File "solve.py", line 129, in scriptRun
    raise Exception(str(e)+"\nCurrent counter at "+str(counter))
Exception: an integer is required
Current counter at 163
$ cat test163
{"anoroc": "v1.320", "secret":
"TISC20{q1_409393ef1b650092ce346a09205c40d6}", "desc": "Submit this.secret
to the TISC grader to complete challenge", "constants": [1116352408,
1899447441, 3049323471, 3921009573, 961987163, 1508970993, 2453635748,
2870763221], "sign": "boy0NNVUj2epGX0mAv5bo9nwBqqwcQ"}
```

Wow I took 6.5 hours.

Flag

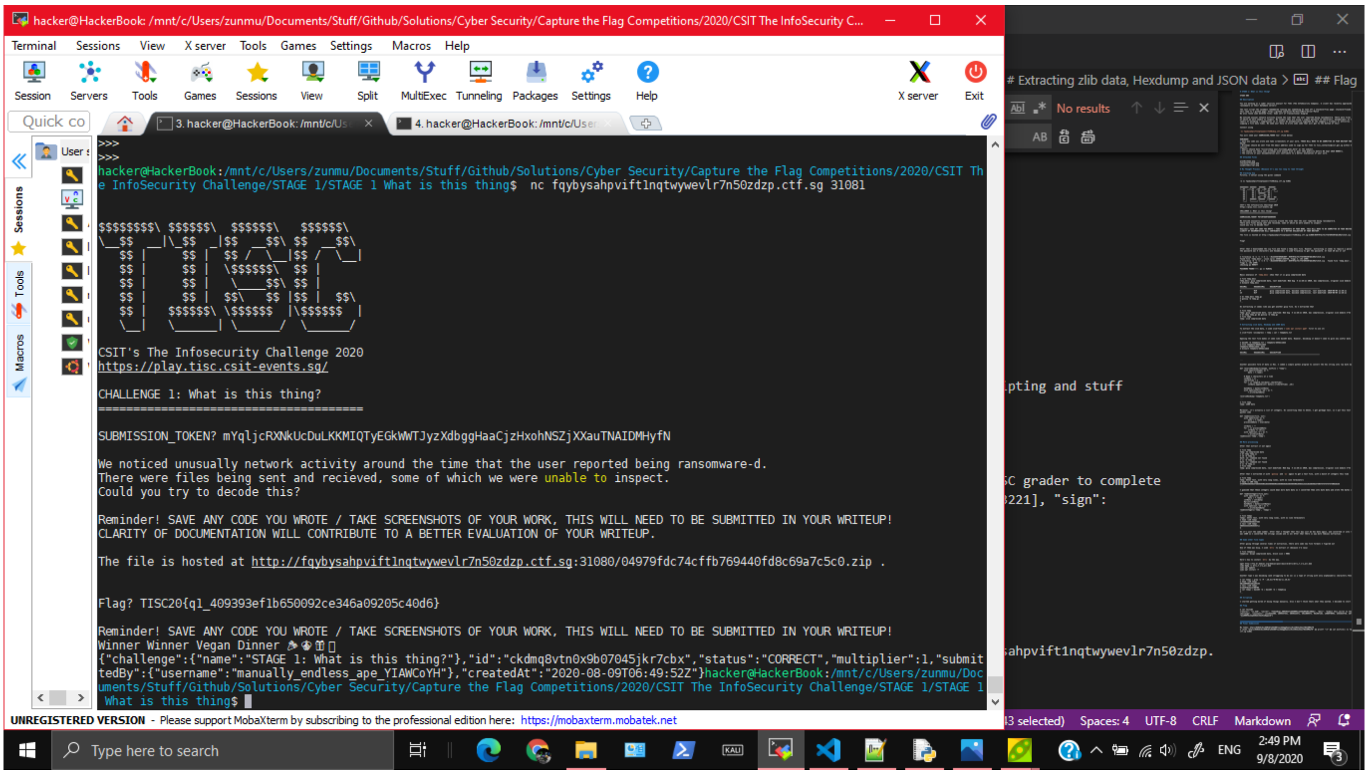
`TISC20{q1_409393ef1b650092ce346a09205c40d6}`

Final Submission

```
$  nc fqybysahpvift1nqtwywevlr7n50zdzp.ctf.sg 31081
.
.
.

Flag? TISC20{q1_409393ef1b650092ce346a09205c40d6}

Reminder! SAVE ANY CODE YOU WROTE / TAKE SCREENSHOTS OF YOUR WORK, THIS
WILL NEED TO BE SUBMITTED IN YOUR WRITEUP!
Winner Winner Vegan Dinner 🎉🎊🎁🥳
{"challenge":{"name":"STAGE 1: What is this
thing?"},"id":"ckdmq8vtn0x9b07045jkr7cbx","status":"CORRECT","multiplier":1
,"submittedBy":
{"username":"manually_endless_ape_YIAWCoYH"},"createdAt":"2020-08-
09T06:49:52Z"}
```



```
>>>
>>>
hacker@HackerBook:/mnt/c/Users/zunmu/Documents/Stuff/Github/Solutions/Cyber Security/Capture the Flag Competitions/2020/CSIT Th
e InfoSecurity Challenge/STAGE 1/STAGE 1 What is this thing$  nc fqybysahpviftlnqtwywevlr7n50zdzp.ctf.sg 31081


$$$$$$$$\ $$$$$$\  $$$$$$\   $$$$$$\
\__$$  __|\_$$  _|$$  __$$\ $$  __$$\
   $$ |     $$ |  $$ /  \__|$$ /  \__|
   $$ |     $$ |  \$$$$$$\  $$ |
   $$ |     $$ |   \____$$\ $$ |
   $$ |     $$ |  $$\   $$ |$$ |  $$\
   $$ |   $$$$$$\ \$$$$$$  |\$$$$$$  |
   \__|   \______| \______/  \______/

CSIT's The Infosecurity Challenge 2020
https://play.tisc.csit-events.sg/

CHALLENGE 1: What is this thing?
======================================

SUBMISSION_TOKEN? mYqljcRXNkUcDuLKKMIQTyEGkWWTJyzXdbggHaaCjzHxohNSZjXXauTNAIDMHyfN

We noticed unusually network activity around the time that the user reported being ransomware-d.
There were files being sent and recieved, some of which we were unable to inspect.
Could you try to decode this?

Reminder! SAVE ANY CODE YOU WROTE / TAKE SCREENSHOTS OF YOUR WORK, THIS WILL NEED TO BE SUBMITTED IN YOUR WRITEUP!
CLARITY OF DOCUMENTATION WILL CONTRIBUTE TO A BETTER EVALUATION OF YOUR WRITEUP.

The file is hosted at http://fqybysahpviftlnqtwywevlr7n50zdzp.ctf.sg:31080/04979fdc74cffb769440fd8c69a7c5c0.zip .


Flag? TISC20{q1_409393ef1b650092ce346a09205c40d6}

Reminder! SAVE ANY CODE YOU WROTE / TAKE SCREENSHOTS OF YOUR WORK, THIS WILL NEED TO BE SUBMITTED IN YOUR WRITEUP!
Winner Winner Vegan Dinner
{"challenge":{"name":"STAGE 1: What is this thing?"},"id":"ckdmq8vtn0x9b07045jkr7cbx","status":"CORRECT","multiplier":1,"submit
tedBy":{"username":"manually_endless_ape_YIAWCoYH"},"createdAt":"2020-08-09T06:49:52Z"}hacker@HackerBook:/mnt/c/Users/zunmu/Doc
uments/Stuff/Github/Solutions/Cyber Security/Capture the Flag Competitions/2020/CSIT The InfoSecurity Challenge/STAGE 1/STAGE 1
 What is this thing$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
# Extracting zlib data, Hexdump and JSON data > ## Flag
2:49 PM
9/8/2020

Final Python Solution

```
# Run in python 2
ZIP_FILE = "04979fdc74cffb769440fd8c69a7c5c0.zip"
PASSWORD = "9a035a"

import os
def route(inname,out):
    if inname != out:
        os.system("mv "+inname+" "+out)

def unzip(filename, out="temp.mess"):
    os.system("unzip"+" -P "+PASSWORD+" "+ZIP_FILE)
    route("temp.mess",out)

def unGZ(filename, out):
    os.system("gunzip "+filename)
    route(filename.split('.')[0],out)
def unXz(filename, out):
    command = "xz -d "+filename
    print('Command:' + command)
    os.system(command)
    route(filename.split('.')[0],out)

def unZlib(filename,out):
    command = 'zlib-flate -uncompress < '+filename+' | cat > '+out
    os.system(command)

def reverseHexDump(filename, outFile = "rawFile"):
    with open(filename) as f:
        data = f.read()

    # Read 2 characters at a time
    intData = []
    charAtTime = 2
    for i in range(0,len(data),charAtTime):
        intData.append(int( data[i:i+charAtTime] ,16))

    byteData = bytes(intData)
    with open(outFile,'wb') as f:
        f.write(byteData)

def readInJson(file, out):
    with open(file) as f:
        data = f.read()
    processedData = eval(data)

    strData = ""
    for i in processedData:
        strData += chr(i)
    with open(out,'w') as f:
        f.write(strData)

############################################################################
##

def legacy():
    os.system('rm temp*')

    unzip(ZIP_FILE,'temp.gz')
    unGZ('temp.gz','temp.gz')
    unGZ('temp.gz','temp.zlib')
    unZlib('temp.zlib','temp')
    reverseHexDump('temp','temp.json')
    readInJson('temp.json', 'temp.xz')
    os.system('rm temp')
    unXz('temp.xz', 'temp.gz')
    unGZ('temp.gz','temp.xz')
    unXz('temp.xz', 'temp')

    reverseHexDump('temp','temp.json')
    readInJson('temp.json', 'temp')
    reverseHexDump('temp', 'temp')
    readInJson('temp', 'temp.zlib')

    unZlib('temp.zlib','temp.bzip')
    os.system("dtrx temp.bzip && mv temp.bzip.1 temp.zlib")
    unZlib('temp.zlib','temp')
    reverseHexDump('temp','temp.json')
    readInJson('temp.json', 'tempb64')
    readInJson('temp.json', 'test')
    #os.system('$ cat tempb64 | base64 -d > test')

    #os.system('rm temp*')
```

```
def singleExtraction(filename, outfile):
    cmd = os.popen('file '+filename).read()
    cmdOut = cmd.split(':')[1]
    print(cmdOut)

    if 'gzip compressed data' in cmdOut:
        #unGZ(filename,outfile)
        os.system("dtrx "+filename+" && mv "+filename+".1 "+outfile)

    elif 'XZ compressed data' in cmdOut:
        os.system('mv '+filename+' '+filename+'.xz')
        unXz(filename+'.xz', outfile)

    elif 'zlib compressed data' in cmdOut:
        unZlib(filename, outfile)

    elif 'ASCII text, with very long lines' in cmdOut:
        with open(filename) as f:
            data = f.read()
        b64Identify = False
        for i in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
            if i in data: b64Identify = True
        if b64Identify: #base64
           os.system("cat "+filename+" | base64 -d > "+outfile)
        else:
            reverseHexDump(filename,outfile)

    elif 'JSON data' in cmdOut:
        readInJson(filename, outfile)

    elif 'bzip2 compressed data' in cmdOut:
        os.system("dtrx "+filename+" && mv "+filename+".1 "+outfile)

    else:
        os.system("cp "+filename+" undoneyet")
        raise Exception("Not implemented file type of "+cmdOut)


def scriptRun(ogFile):
    os.system("cp "+ogFile+" "+ogFile+str(0))
    counter = 0
    while True:
        inf , outf = ogFile+str(counter), ogFile+str(counter+1)
        try:
            singleExtraction(inf, outf)
        except Exception as e:
            print(str(e))
            raise Exception(str(e)+"\nCurrent counter at "+str(counter))
        counter += 1

if True:
    legacy()
#singleExtraction('tempb64', 'temp8')
```

```
scriptRun('test')
#reverseHexDump('temp1','temp.json')
```

STAGE 2: Find me some keys

STAGE TWO

Description

YOU WILL NEED ENCRYPTED.ZIP ATTACHED BELOW

You have recovered the ransomware that has been running on the computers. First, you need to figure out how the ransomware works, what encryption algorithm it uses.

Please handle the anorocware executable with caution, we have provided a Dockerfile for you to containerise the ransomware. You may also choose to use a Virtual Machine.

Find the public key that is embedded in the binary. Hash the base64 string of the public key using sha256 and submit flag as TISC20{sha256 hash of the base64 encoded public key}

ANSWER FORMAT EXAMPLE:

The key file will look something like this but longer: `LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0NCmMyOXRaU0JpWVhOblE5qUWdjM1J5YVc1bklHZHZaWE1nYUdWeVpRPT0NCi0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLQ==`

Internally it contains something like this:

```
-----BEGIN PUBLIC KEY-----
c29tZSBiYXNlNjQgc3RyaW5nIGdvZXMgaGVyZQ==
-----END PUBLIC KEY-----
```

You will need to perform a sha256 hash on the whole key file, i.e. `sha256("LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0NCmMyOXRaU0JpWVhOblE5qUWdjM1J5YVc1bklHZHZaWE1nYUdWeVpRPT0NCi0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLQ==")` to obtain `799f319d8400243a8331d70ae7df4bacf0290dca303b138b03263655e9548caf`

Then submit flag as `TISC20{799f319d8400243a8331d70ae7df4bacf0290dca303b138b03263655e9548caf}`

REMINDER!

Save any code you wrote and take screenshots of your work within your allocated 48h session, THESE WILL NEED TO BE SUBMITTED IN YOUR WRITEUP FOR YOUR ATTEMPT TO BE CONSIDERED VALID! Writeups should be sent from the email address used to sign up for TISC to tisc_contactus@csit.gov.sg within 48h after the end of your 48h session. Emails should have "tiscwriteup your_fullname_goes_here" as the subject. You should not share your writeups with anyone else or post them publicly before 08 Sept 2020 0000hrs. The clarity of your documentation will contribute to a better evaluation of your work.

Attached Files

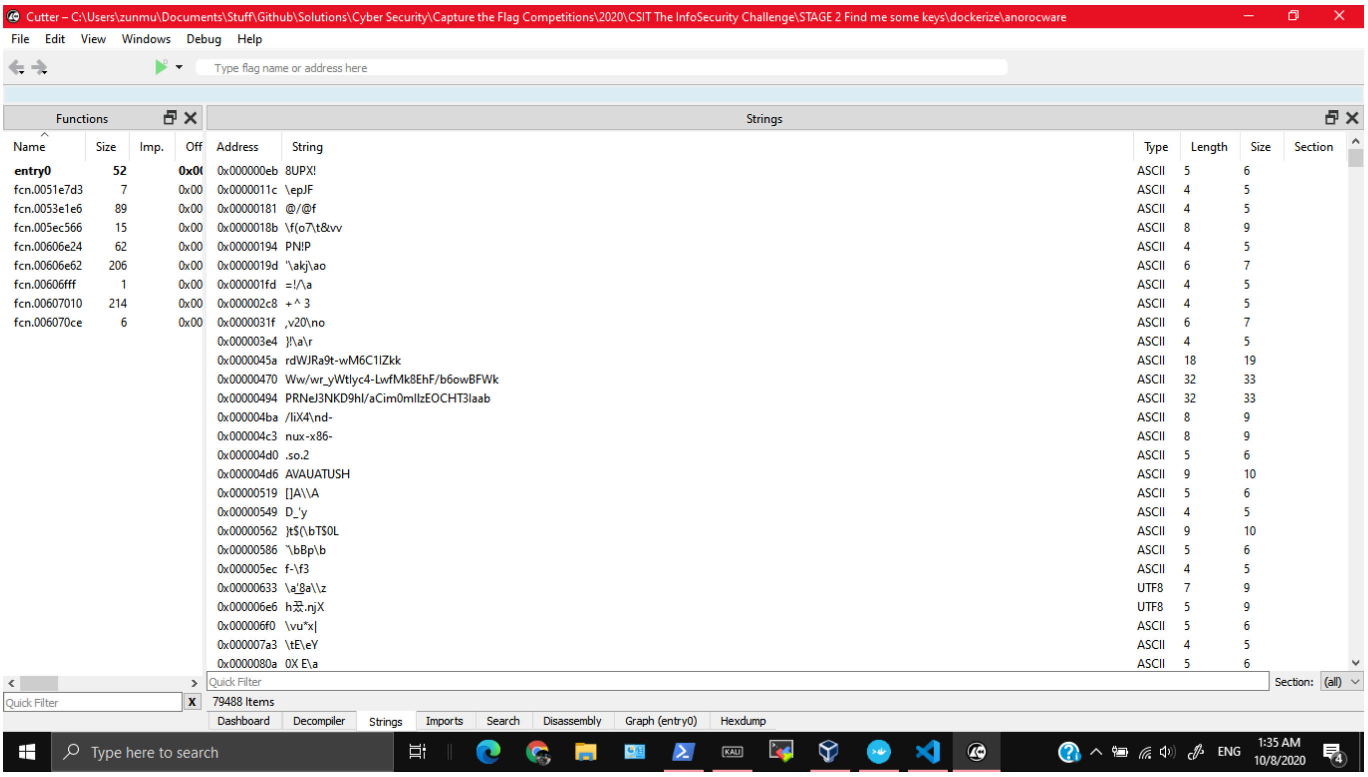
encrypted.zip

Solution

Finding out it's UPX packed

I'm using Windows, so I need to setup docker first. ~~Totally didn't spend 1h 45 min trying to extract a linux VM becuase my original one broke and failing~~. A good way to do it is to `choco install docker-toolbox`. It sets up the server for you, so all you need to do is to open a program, and you get access to a command prompt with docker access.

Firstly, I decompiled the program using Cutter (which uses Ghidra's Decompiler). Cutter is a good tool for Windows (Don't have to deal with Java and cumbersome UI and setup). It doesn't seem like it has any meaningful strings.

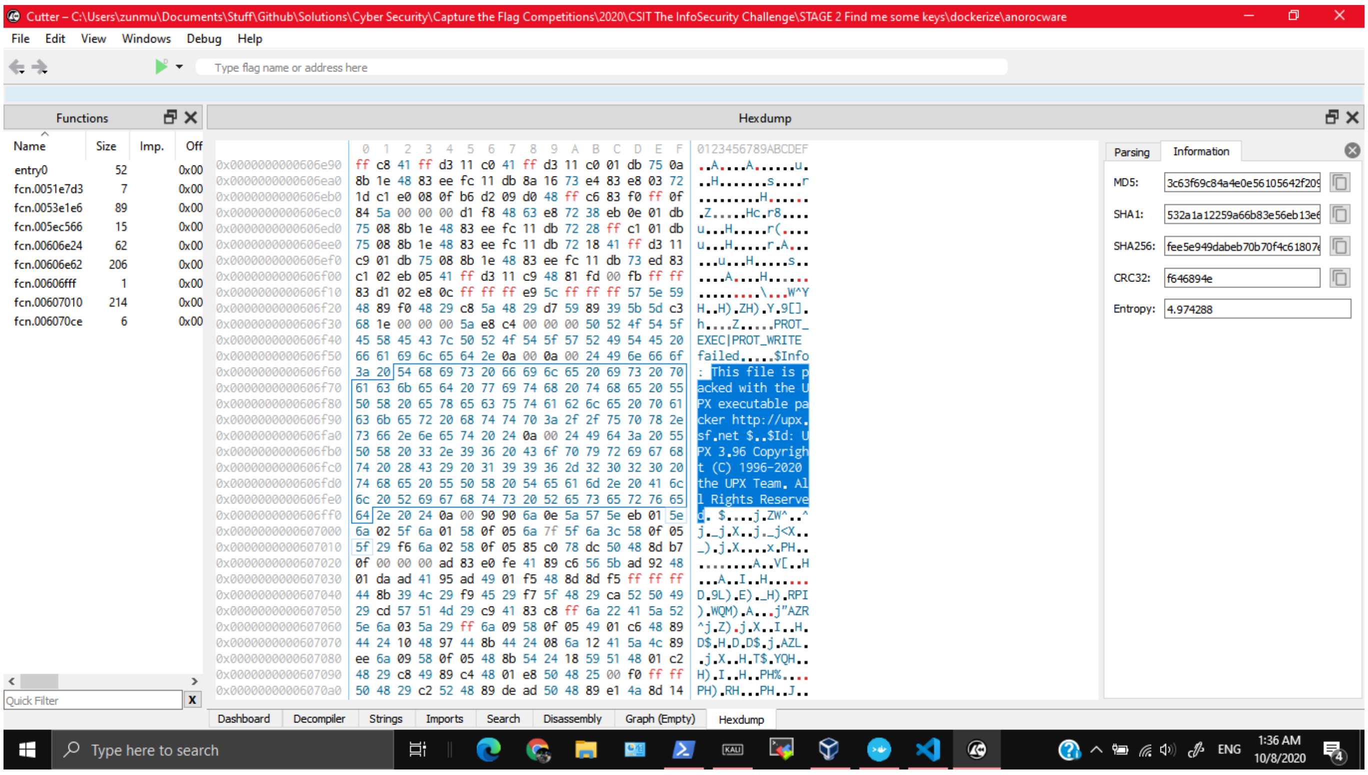

Cutter – C:\Users\zunmu\Documents\Stuff\Github\Solutions\Cyber Security\Capture the Flag Competitions\2020\CSIT The InfoSecurity Challenge\STAGE 2 Find me some keys\dockerize\anorocware

Name	Size	Imp.	Off
entry0	**52**		**0x0(**
fcn.0051e7d3	7		0x00
fcn.0053e1e6	89		0x00
fcn.005ec566	15		0x00
fcn.00606e24	62		0x00
fcn.00606e62	206		0x00
fcn.00606fff	1		0x00
fcn.00607010	214		0x00
fcn.006070ce	6		0x00

Address	String	Type	Length	Size	Section
0x000000eb	8UPX!	ASCII	5	6	
0x0000011c	\epJF	ASCII	4	5	
0x00000181	@/@f	ASCII	4	5	
0x0000018b	\f(o7\t&vv	ASCII	8	9	
0x00000194	PN!P	ASCII	4	5	
0x0000019d	'\akj\ao	ASCII	6	7	
0x000001fd	=!/\a	ASCII	4	5	
0x000002c8	+ ^ 3	ASCII	4	5	
0x0000031f	,v20\no	ASCII	6	7	
0x000003e4	}!\a\r	ASCII	4	5	
0x0000045a	rdWJRa9t-wM6C1lZkk	ASCII	18	19	
0x00000470	Ww/wr_yWtlyc4-LwfMk8EhF/b6owBFWk	ASCII	32	33	
0x00000494	PRNeJ3NKD9hl/aCim0mllzEOCHT3laab	ASCII	32	33	
0x000004ba	/liX4\nd-	ASCII	8	9	
0x000004c3	nux-x86-	ASCII	8	9	
0x000004d0	.so.2	ASCII	5	6	
0x000004d6	AVAUATUSH	ASCII	9	10	
0x00000519	[]A\\A	ASCII	5	6	
0x00000549	D_'y	ASCII	4	5	
0x00000562	}t$(\bT$0L	ASCII	9	10	
0x00000586	`\bBp\b	ASCII	5	6	
0x000005ec	f-\f3	ASCII	4	5	
0x00000633	\a'8a\\z	UTF8	7	9	
0x000006e6	h买.njX	UTF8	5	9	
0x000006f0	\vu*x|	ASCII	5	6	
0x000007a3	\tE\eY	ASCII	4	5	
0x0000080a	0X E\a	ASCII	5	6	

79488 Items

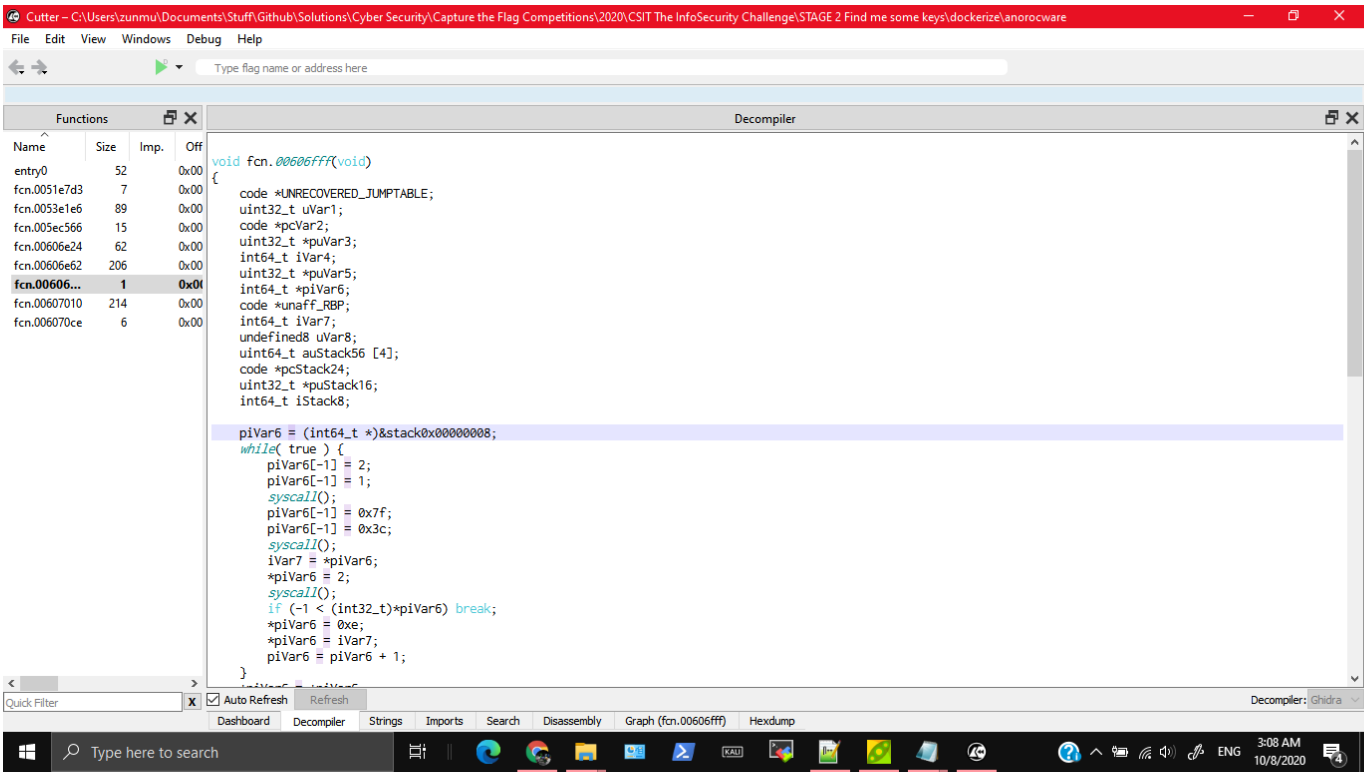
However, on looking at the hex dump, it seems like it is packed with UPX packer? That seems interesting (maybe it's why the decompiler couldn't fully decompile ~~or it's just me being lazy to read~~).


Cutter – C:\Users\zunmu\Documents\Stuff\Github\Solutions\Cyber Security\Capture the Flag Competitions\2020\CSIT The InfoSecurity Challenge\STAGE 2 Find me some keys\dockerize\anorocware
File
Edit
View
Windows
Debug
Help
Type flag name or address here
Functions
Hexdump
Parsing
Information
Quick Filter
Dashboard
Decompiler
Strings
Imports
Search
Disassembly
Graph (Empty)
Hexdump
Type here to search
ENG
1:36 AM
10/8/2020

Name	Size	Imp.	Off
entry0	52		0x00
fcn.0051e7d3	7		0x00
fcn.0053e1e6	89		0x00
fcn.005ec566	15		0x00
fcn.00606e24	62		0x00
fcn.00606e62	206		0x00
fcn.00606fff	1		0x00
fcn.00607010	214		0x00
fcn.006070ce	6		0x00

```
                    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0x0000000000606e90  ff c8 41 ff d3 11 c0 41 ff d3 11 c0 01 db 75 0a  ..A....A......u.
0x0000000000606ea0  8b 1e 48 83 ee fc 11 db 8a 16 73 e4 83 e8 03 72  ..H.......s....r
0x0000000000606eb0  1d c1 e0 08 0f b6 d2 09 d0 48 ff c6 83 f0 ff 0f  .........H......
0x0000000000606ec0  84 5a 00 00 00 d1 f8 48 63 e8 72 38 eb 0e 01 db  .Z.....Hc.r8....
0x0000000000606ed0  75 08 8b 1e 48 83 ee fc 11 db 72 28 ff c1 01 db  u...H.....r(....
0x0000000000606ee0  75 08 8b 1e 48 83 ee fc 11 db 72 18 41 ff d3 11  u...H.....r.A...
0x0000000000606ef0  c9 01 db 75 08 8b 1e 48 83 ee fc 11 db 73 ed 83  ...u...H.....s..
0x0000000000606f00  c1 02 eb 05 41 ff d3 11 c9 48 81 fd 00 fb ff ff  ....A....H......
0x0000000000606f10  83 d1 02 e8 0c ff ff ff e9 5c ff ff ff 57 5e 59  .........\...W^Y
0x0000000000606f20  48 89 f0 48 29 c8 5a 48 29 d7 59 89 39 5b 5d c3  H..H).ZH).Y.9[].
0x0000000000606f30  68 1e 00 00 00 5a e8 c4 00 00 00 50 52 4f 54 5f  h....Z.....PROT_
0x0000000000606f40  45 58 45 43 7c 50 52 4f 54 5f 57 52 49 54 45 20  EXEC|PROT_WRITE 
0x0000000000606f50  66 61 69 6c 65 64 2e 0a 00 0a 00 24 49 6e 66 6f  failed.....$Info
0x0000000000606f60  3a 20 54 68 69 73 20 66 69 6c 65 20 69 73 20 70  : This file is p
0x0000000000606f70  61 63 6b 65 64 20 77 69 74 68 20 74 68 65 20 55  acked with the U
0x0000000000606f80  50 58 20 65 78 65 63 75 74 61 62 6c 65 20 70 61  PX executable pa
0x0000000000606f90  63 6b 65 72 20 68 74 74 70 3a 2f 2f 75 70 78 2e  cker http://upx.
0x0000000000606fa0  73 66 2e 6e 65 74 20 24 0a 00 24 49 64 3a 20 55  sf.net $..$Id: U
0x0000000000606fb0  50 58 20 33 2e 39 36 20 43 6f 70 79 72 69 67 68  PX 3.96 Copyrigh
0x0000000000606fc0  74 20 28 43 29 20 31 39 39 36 2d 32 30 32 30 20  t (C) 1996-2020 
0x0000000000606fd0  74 68 65 20 55 50 58 20 54 65 61 6d 2e 20 41 6c  the UPX Team. Al
0x0000000000606fe0  6c 20 52 69 67 68 74 73 20 52 65 73 65 72 76 65  l Rights Reserve
0x0000000000606ff0  64 2e 20 24 0a 00 90 90 6a 0e 5a 57 5e eb 01 5e  d. $....j.ZW^..^
0x0000000000607000  6a 02 5f 6a 01 58 0f 05 6a 7f 5f 6a 3c 58 0f 05  j._j.X..j._j<X..
0x0000000000607010  5f 29 f6 6a 02 58 0f 05 85 c0 78 dc 50 48 8d b7  _).j.X....x.PH..
0x0000000000607020  0f 00 00 00 ad 83 e0 fe 41 89 c6 56 5b ad 92 48  ........A..V[..H
0x0000000000607030  01 da ad 41 95 ad 49 01 f5 48 8d 8d f5 ff ff ff  ...A..I..H......
0x0000000000607040  44 8b 39 4c 29 f9 45 29 f7 5f 48 29 ca 52 50 49  D.9L).E)._H).RPI
0x0000000000607050  29 cd 57 51 4d 29 c9 41 83 c8 ff 6a 22 41 5a 52  ).WQM).A...j"AZR
0x0000000000607060  5e 6a 03 5a 29 ff 6a 09 58 0f 05 49 01 c6 48 89  ^j.Z).j.X..I..H.
0x0000000000607070  44 24 10 48 97 44 8b 44 24 08 6a 12 41 5a 4c 89  D$.H.D.D$.j.AZL.
0x0000000000607080  ee 6a 09 58 0f 05 48 8b 54 24 18 59 51 48 01 c2  .j.X..H.T$.YQH..
0x0000000000607090  48 29 c8 49 89 c4 48 01 e8 50 48 25 00 f0 ff ff  H).I..H..PH%....
0x00000000006070a0  50 48 29 c2 52 48 89 de ad 50 48 89 e1 4a 8d 14  PH).RH...PH..J..
```

MD5:	3c63f69c84a4e0e56105642f20
SHA1:	532a1a12259a66b83e56eb13e
SHA256:	fee5e949dabeb70b70f4c61807
CRC32:	f646894e
Entropy:	4.974288


Cutter – C:\Users\zunmu\Documents\Stuff\Github\Solutions\Cyber Security\Capture the Flag Competitions\2020\CSIT The InfoSecurity Challenge\STAGE 2 Find me some keys\dockerize\anorocware
File
Edit
View
Windows
Debug
Help
Type flag name or address here
Functions
Decompiler
Quick Filter
Auto Refresh
Refresh
Decompiler: Ghidra
Dashboard
Decompiler
Strings
Imports
Search
Disassembly
Graph (fcn.00606fff)
Hexdump
Type here to search
ENG
3:08 AM
10/8/2020

Name	Size	Imp.	Off
entry0	52		0x00
fcn.0051e7d3	7		0x00
fcn.0053e1e6	89		0x00
fcn.005ec566	15		0x00
fcn.00606e24	62		0x00
fcn.00606e62	206		0x00
fcn.00606...	1		0x00
fcn.00607010	214		0x00
fcn.006070ce	6		0x00

```c
void fcn.00606fff(void)
{
    code *UNRECOVERED_JUMPTABLE;
    uint32_t uVar1;
    code *pcVar2;
    uint32_t *puVar3;
    int64_t iVar4;
    uint32_t *puVar5;
    int64_t *piVar6;
    code *unaff_RBP;
    int64_t iVar7;
    undefined8 uVar8;
    uint64_t auStack56 [4];
    code *pcStack24;
    uint32_t *puStack16;
    int64_t iStack8;

    piVar6 = (int64_t *)&stack0x00000008;
    while( true ) {
        piVar6[-1] = 2;
        piVar6[-1] = 1;
        syscall();
        piVar6[-1] = 0x7f;
        piVar6[-1] = 0x3c;
        syscall();
        iVar7 = *piVar6;
        *piVar6 = 2;
        syscall();
        if (-1 < (int32_t)*piVar6) break;
        *piVar6 = 0xe;
        *piVar6 = iVar7;
        piVar6 = piVar6 + 1;
    }
```

I set up the container by running `docker build dockerize`. After that I ran some basic commands to figure out what it was doing (since I can't actually read the decompiled code properly). By the way if you want to get a terminal out of the container run `docker run -i -t CONTAINER_ID bash`

```
$ docker run anorocware './anorocware'
:(
$ docker run anorocware 'ltrace ./anorocware'
Couldn't find .dynsym or .dynstr in "/proc/8/exe"
$ docker run anorocware 'strace anorocware' > strace.log
```

After some googling I realised you could easily unpack the executable with a linux command. I installed upx in the docker container (actually in normal linux works too) and unpacked the executable.

```
root@23bbdce9e1da:~# apt install upx
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'upx-ucl' instead of 'upx'
The following additional packages will be installed:
  libucl1
The following NEW packages will be installed:
  libucl1 upx-ucl
0 upgraded, 2 newly installed, 0 to remove and 4 not upgraded.
Need to get 417 kB of archives.
After this operation, 2158 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
.
.
.
.
Processing triggers for libc-bin (2.31-0ubuntu9) ...
root@23bbdce9e1da:~# upx
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95        Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th
2018

Usage: upx [-123456789dlthVL] [-qvfk] [-o file] file..

Commands:
  -1     compress faster                   -9    compress better
  -d     decompress                        -l    list compressed file
  -t     test compressed file              -V    display version number
  -h     give more help                    -L    display software license
Options:
  -q     be quiet                          -v    be verbose
  -oFILE write output to 'FILE'
  -f     force compression of suspicious files
  -k     keep backup files
file..   executables to (de)compress

Type 'upx --help' for more detailed help.

UPX comes with ABSOLUTELY NO WARRANTY; for details visit
https://upx.github.io
root@23bbdce9e1da:~# upx -d anorocware
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95        Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th
2018

        File size         Ratio      Format      Name
```

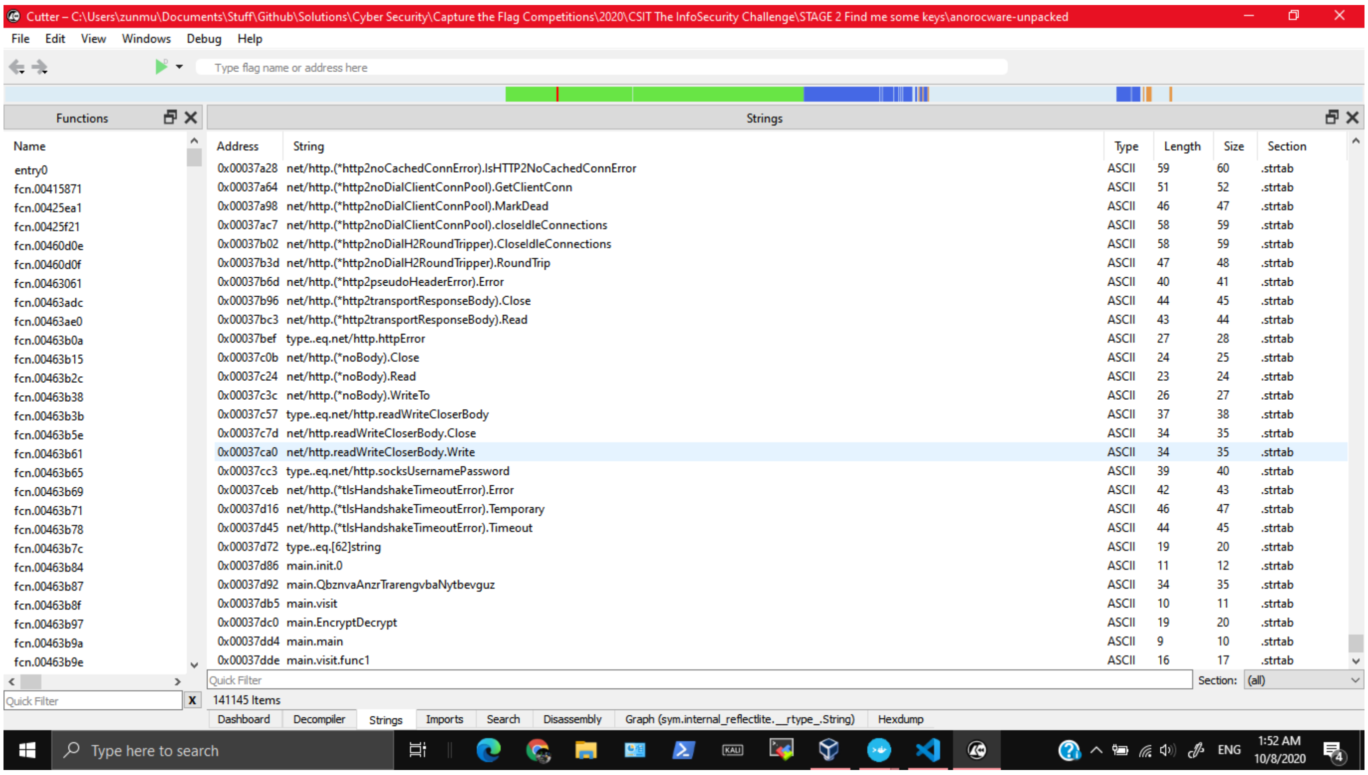
```
    --------------------   ------   -----------   -----------
    7406375 <-   3993332   53.92%   linux/amd64   anorocware

Unpacked 1 file.
root@23bbdce9e1da:~# ls
anorocware  clients.db.anoroc  email  images  keydetails-enc.txt
ransomnote-anoroc.txt  secret_investments.db.anoroc
```

Strings

After that I copied the unpacked executable out from the docker container and analysed it in Cutter. The results are much more promising. The decompiled code is more readable.

```
$ docker cp reverent_allen:/root/anorocware anorocware-unpacked
```



Cutter – C:\Users\zunmu\Documents\Stuff\Github\Solutions\Cyber Security\Capture the Flag Competitions\2020\CSIT The InfoSecurity Challenge\STAGE 2 Find me some keys\anorocware-unpacked

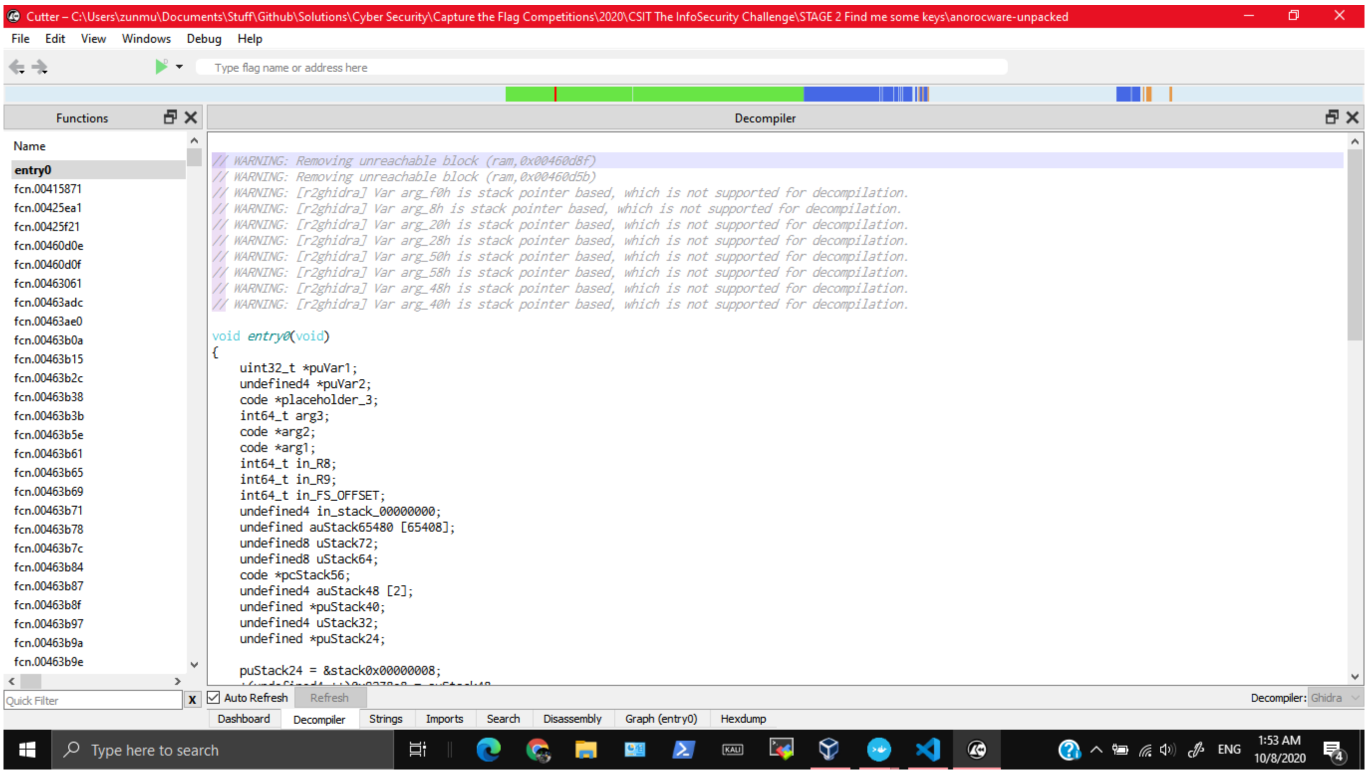
Functions

Name
entry0
fcn.00415871
fcn.00425ea1
fcn.00425f21
fcn.00460d0e
fcn.00460d0f
fcn.00463061
fcn.00463adc
fcn.00463ae0
fcn.00463b0a
fcn.00463b15
fcn.00463b2c
fcn.00463b38
fcn.00463b3b
fcn.00463b5e
fcn.00463b61
fcn.00463b65
fcn.00463b69
fcn.00463b71
fcn.00463b78
fcn.00463b7c
fcn.00463b84
fcn.00463b87
fcn.00463b8f
fcn.00463b97
fcn.00463b9a
fcn.00463b9e

Strings

Address	String	Type	Length	Size	Section
0x00037a28	net/http.(*http2noCachedConnError).IsHTTP2NoCachedConnError	ASCII	59	60	.strtab
0x00037a64	net/http.(*http2noDialClientConnPool).GetClientConn	ASCII	51	52	.strtab
0x00037a98	net/http.(*http2noDialClientConnPool).MarkDead	ASCII	46	47	.strtab
0x00037ac7	net/http.(*http2noDialClientConnPool).closeIdleConnections	ASCII	58	59	.strtab
0x00037b02	net/http.(*http2noDialH2RoundTripper).CloseIdleConnections	ASCII	58	59	.strtab
0x00037b3d	net/http.(*http2noDialH2RoundTripper).RoundTrip	ASCII	47	48	.strtab
0x00037b6d	net/http.(*http2pseudoHeaderError).Error	ASCII	40	41	.strtab
0x00037b96	net/http.(*http2transportResponseBody).Close	ASCII	44	45	.strtab
0x00037bc3	net/http.(*http2transportResponseBody).Read	ASCII	43	44	.strtab
0x00037bef	type..eq.net/http.httpError	ASCII	27	28	.strtab
0x00037c0b	net/http.(*noBody).Close	ASCII	24	25	.strtab
0x00037c24	net/http.(*noBody).Read	ASCII	23	24	.strtab
0x00037c3c	net/http.(*noBody).WriteTo	ASCII	26	27	.strtab
0x00037c57	type..eq.net/http.readWriteCloserBody	ASCII	37	38	.strtab
0x00037c7d	net/http.readWriteCloserBody.Close	ASCII	34	35	.strtab
0x00037ca0	net/http.readWriteCloserBody.Write	ASCII	34	35	.strtab
0x00037cc3	type..eq.net/http.socksUsernamePassword	ASCII	39	40	.strtab
0x00037ceb	net/http.(*tlsHandshakeTimeoutError).Error	ASCII	42	43	.strtab
0x00037d16	net/http.(*tlsHandshakeTimeoutError).Temporary	ASCII	46	47	.strtab
0x00037d45	net/http.(*tlsHandshakeTimeoutError).Timeout	ASCII	44	45	.strtab
0x00037d72	type..eq.[62]string	ASCII	19	20	.strtab
0x00037d86	main.init.0	ASCII	11	12	.strtab
0x00037d92	main.QbznvaAnzrTrarengvbaNytbevguz	ASCII	34	35	.strtab
0x00037db5	main.visit	ASCII	10	11	.strtab
0x00037dc0	main.EncryptDecrypt	ASCII	19	20	.strtab
0x00037dd4	main.main	ASCII	9	10	.strtab
0x00037dde	main.visit.func1	ASCII	16	17	.strtab

141145 Items



```
// WARNING: Removing unreachable block (ram,0x00460d8f)
// WARNING: Removing unreachable block (ram,0x00460d5b)
// WARNING: [r2ghidra] Var arg_f0h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_8h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_20h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_28h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_50h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_58h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_48h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_40h is stack pointer based, which is not supported for decompilation.

void entry0(void)
{
    uint32_t *puVar1;
    undefined4 *puVar2;
    code *placeholder_3;
    int64_t arg3;
    code *arg2;
    code *arg1;
    int64_t in_R8;
    int64_t in_R9;
    int64_t in_FS_OFFSET;
    undefined4 in_stack_00000000;
    undefined auStack65480 [65408];
    undefined8 uStack72;
    undefined8 uStack64;
    code *pcStack56;
    undefined4 auStack48 [2];
    undefined *puStack40;
    undefined4 uStack32;
    undefined *puStack24;

    puStack24 = &stack0x00000008;
```

There was also a potential long string `rdWJRa9t-wM6C1IZkkWw/wr_yWtIyc4-LwfMk8EhF/b6owBFWkPRNeJ3NKD9hl/aCim0mIlzEOCHT3IaabM` which could be the key but probably not.

The function names remind me a lot of Java. On some googling, it's actually possible to convert from Java to Linux Executable through the use of gcj and also some methods. Maybe it is possible to reverse the change?

Let's analyse the file. Running ltrace on it seems a lot more promising this time.

```
root@23bbdce9e1da:~# ltrace ./anorocware
malloc(56)
= 0xbca2a0
pthread_attr_init(0xbca2a0, 0, 65, 0xbca2d0)
= 0
pthread_attr_getstacksize(0xbca2a0, 0x7ffda6f2b060, 65, 0xbca2d0)
= 0
pthread_attr_destroy(0xbca2a0, 0x7ffda6f2b060, 1, 0xbca2d0)
= 0
free(0xbca2a0)
= <void>
mmap(0, 0x40000, 3, 34)
= 0x7f4a98da0000
mmap(0, 0x20000, 0, 34)
= 0x7f4a98d80000
mmap(0, 0x100000, 0, 34)
= 0x7f4a98c80000
mmap(0, 0x800000, 0, 34)
= 0x7f4a98480000
mmap(0, 0x4000000, 0, 34)
= 0x7f4a94480000
mmap(0, 0x20000000, 0, 34)
```

```
= 0x7f4a74480000
mmap(0xc000000000, 0x4000000, 0, 34)
= 0xc000000000
mmap(0xc000000000, 0x4000000, 3, 50)
= 0xc000000000
mmap(0, 0x2000000, 3, 34)
= 0x7f4a72480000
mmap(0, 0x210808, 3, 34)
= 0x7f4a7226f000
mmap(0x7f4a98d80000, 0x20000, 3, 50)
= 0x7f4a98d80000
.
.
.
000000c000057af8:  8afdad80b547b63b  43b2a29ce3427104
000000c000057b08:  a590fd826b98e69c  2cfec8a1b2b10275
000000c000057b18:  6a1ae8c9116e8fc8  43b2a29ce3427104
000000c000057b28:  00000007676000d0  00000007676000d0
000000c000057b38:  000000c0000ece38  000000c0000ecf90
000000c000057b48:  0000000000000004  00000007676000d0
000000c000057b58:  000000c0000ecf90  000000c0000ecd20
000000c000057b68:  000000000044996f <runtime.sighandler+1967>
0000000000010247
000000c000057b78:  002b000000000033  0000000000000004
000000c000057b88:  000000000000000e  0000000000000000
000000c000057b98:  00000007676000d0  000000c000057c80
000000c000057ba8:  0000000000000000  0000000000000000
000000c000057bb8:  0000000000000000
runtime.sigtramp(0x7, 0x0, 0xc000050000, 0x0, 0x8000, 0x4,
0x2cfec8a1b2b10275, 0x8afdad80b547b63b, 0x43b2a29ce3427104,
0xa590fd826b98e69c, ...)
        /usr/lib/go/src/runtime/sys_linux_amd64.s:389 +0x43

goroutine 12 [running]:
+++ killed by SIGSEGV +++
```

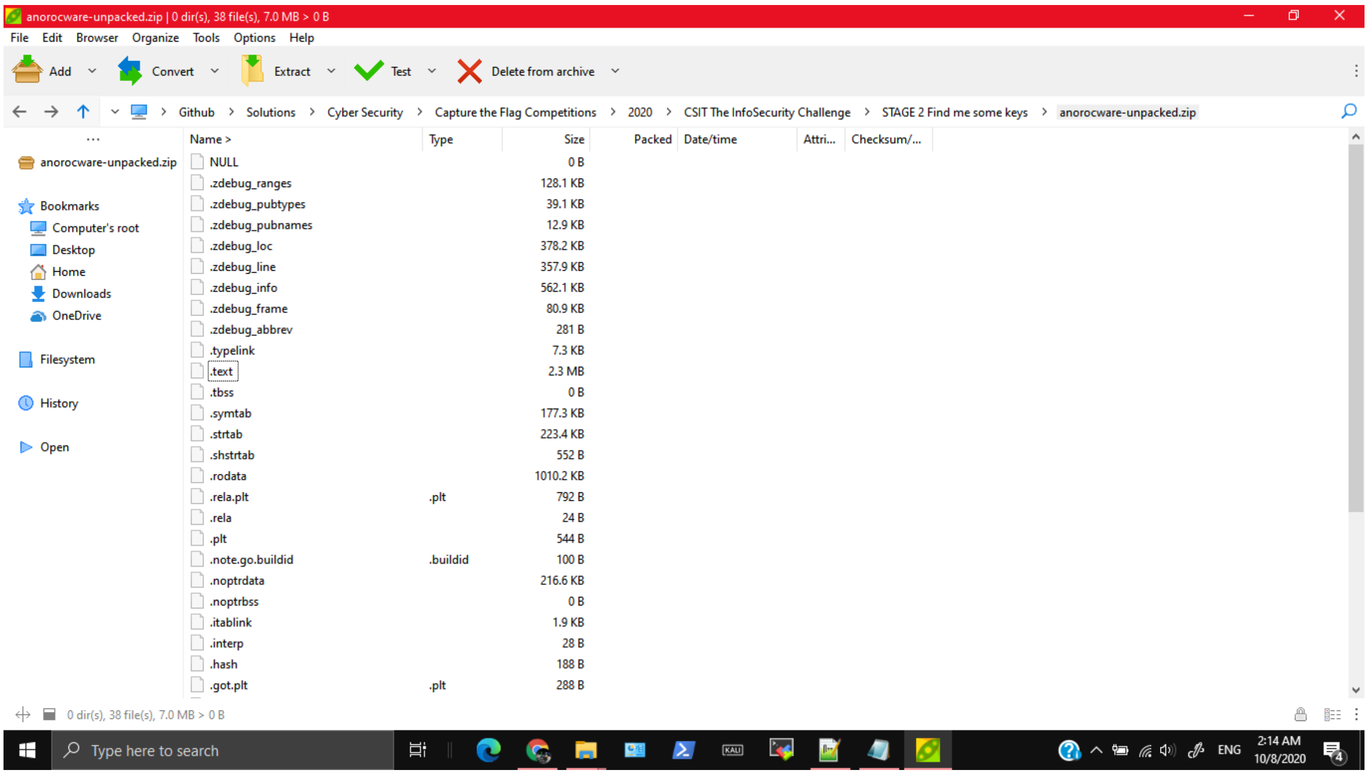
Static analysis on the file type is interesting

```
$ file anorocware-unpacked
anorocware-unpacked: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, Go
BuildID=rdWJRa9t-wM6C1IZkkWw/wr_yWtIyc4-
LwfMk8EhF/b6owBFWkPRNeJ3NKD9hl/aCim0mIlzEOCHT3IaabM, not stripped
$ binwalk anorocware-unpacked

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
-----
0             0x0             ELF, 64-bit LSB executable, AMD x86-64,
version 1 (SYSV)
3065246       0x2EC59E        Unix path:
/dev/stdin/etc/hosts1220703125610351525625: parsing :authorityAdditionalBad
```

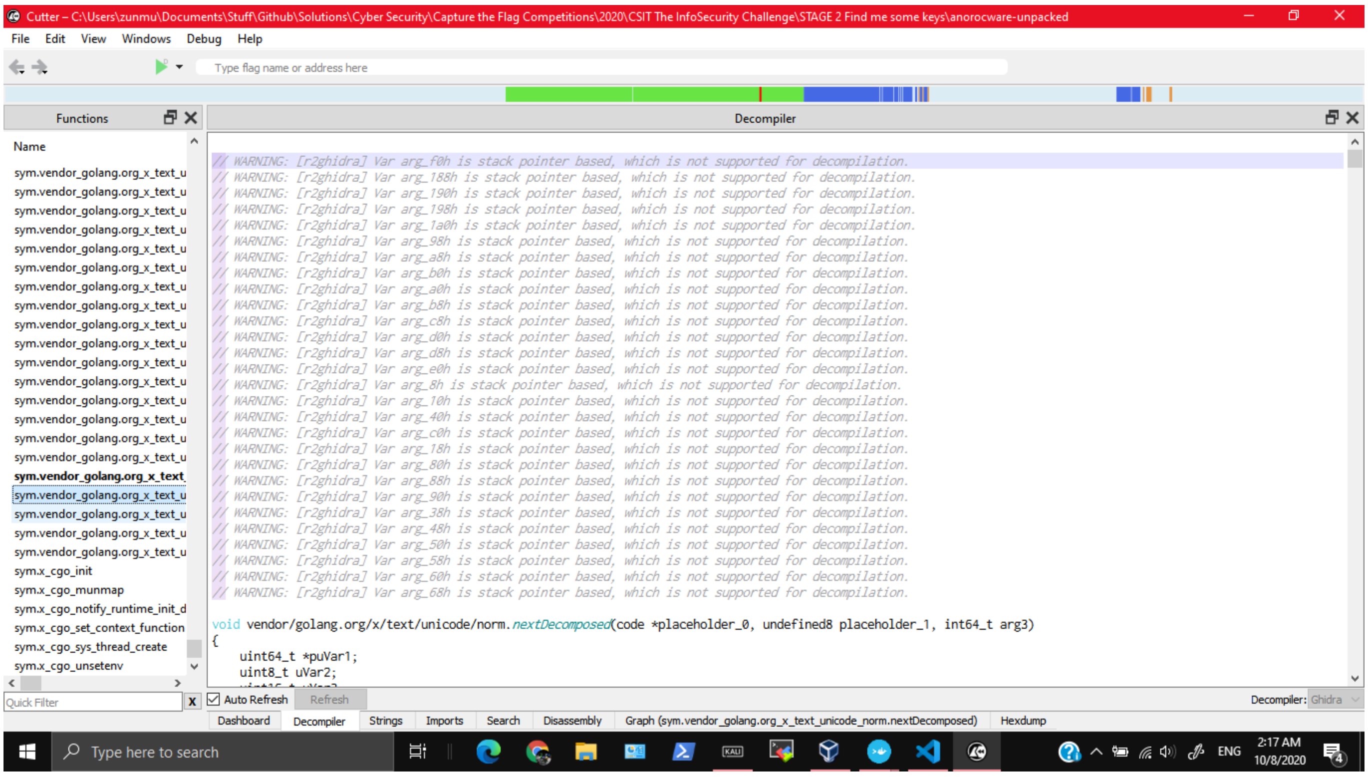
```
varintClassCHAOSClassCSNETConnectionContent-IdDSA-SHA2
3066237       0x2EC97D        Unix path:
/dev/stderr/dev/stdout/index.html30517578125: frame.sp=Bad GatewayBad
RequestClassHESIODCookie.PathENABLE_PUSHEND_HEADERSEarly H
3069522       0x2ED652        Unix path:
/etc/protocols/etc/ssl/certs/proc/self/exe/var/ssl/certs1907348632812595367
431640625: extra text: <not Stringer>Accept-CharsetCo
3070519       0x2EDA37        Unix path: /etc/machine-
id/etc/mdns.allow/etc/mime.types0601021504Z0700476837158203125: cannot
parse :ValidateLabels<invalid Value>ASCII_He
3072972       0x2EE3CC        Unix path:
/etc/ssl/cert.pem0123456789ABCDEFX0123456789abcdefx060102150405Z07001192092
8955078125596046447753906255: missing method ; SameSit
3077222       0x2EF466        Unix path:
/usr/share/zoneinfo/37252902984619140625Egyptian_HieroglyphsIDS_Trinary_Ope
ratorInsufficient StorageMAX_HEADER_LIST_SIZEMeroitic
3079808       0x2EFE80        Unix path: /etc/apache/mime.types/etc/ssl/ca-
bundle.pem/lib/time/zoneinfo.zip/usr/local/share/certs4656612873077392578l2
5DEBUG_HTTP2_GOROUT
3081372       0x2F049C        Unix path:
/etc/apache2/mime.types/etc/pki/tls/cacert.pem23283064365386962890625<inval
id reflect.Value>CLIENT_TRAFFIC_SECRET_0Logical_Order
3083189       0x2F0BB5        Unix path:
/usr/share/lib/zoneinfo//var/lib/dbus/machine-
id116415321826934814453125582076609134674072265625Request Entity Too
Large", requi
3095270       0x2F3AE6        Unix path: /etc/pki/tls/certs/ca-
bundle.crt28421709430404007434844970703125: day-of-year does not match
dayMapIter.Value called before Next
3117838       0x2F930E        Unix path: /etc/pki/ca-
trust/extracted/pem/tls-ca-bundle.pemECDSA signature contained zero or
negative valuescasgstatus: waiting for Gwaiti
3127445       0x2FB895        Base64 standard index table
3409752       0x340758        Intel x86 or x64 microcode, pf_mask 0x00,
1A00-01-03, size 259
3451582       0x34AABE        VxWorks symbol table, big endian, first
entry: [type: initialized data, code address: 0x800, symbol address: 0x100]
3453088       0x34B0A0        Unix path: /usr/lib/go
3521376       0x35BB60        SHA256 hash constants, little endian
3521392       0x35BB70        SHA256 hash constants, little endian
3883602       0x3B4252        Cisco IOS microcode, for ""
5111920       0x4E0070        Unix path: /usr/lib/go/src/os/executable.go
5142496       0x4E77E0        Unix path:
/sys/kernel/mm/transparent_hugepage/hpage_pmd_size
5160905       0x4EBFC9        Boot section Start 0x42424242 End 0x42424242
5160909       0x4EBFCD        Boot section Start 0x42424242 End 0x0
5160913       0x4EBFD1        Boot section Start 0x0 End 0x0
5160917       0x4EBFD5        Boot section Start 0x0 End 0x0
5901625       0x5A0D39        Unix path: /usr/lib/go/src/runtime/runtime-
gdb.py
6996591       0x6AC26F        mcrypt 2.2 encrypted data, algorithm:
blowfish-448, mode: CBC, keymode: 4bit
7090183       0x6C3007        mcrypt 2.2 encrypted data, algorithm:
blowfish-448, mode: CBC, keymode: 8bit
```

On a whim based on this resource I decided to try unzipping the file. To my surprise it's extractable (even though no zip file header or something)!


anorocware-unpacked.zip | 0 dir(s), 38 file(s), 7.0 MB > 0 B
File Edit Browser Organize Tools Options Help
Add Convert Extract Test Delete from archive
Github > Solutions > Cyber Security > Capture the Flag Competitions > 2020 > CSIT The InfoSecurity Challenge > STAGE 2 Find me some keys > anorocware-unpacked.zip
anorocware-unpacked.zip
Bookmarks
Computer's root
Desktop
Home
Downloads
OneDrive
Filesystem
History
Open
0 dir(s), 38 file(s), 7.0 MB > 0 B
Type here to search
ENG
2:14 AM
10/8/2020

Name	Type	Size	Packed	Date/time	Attri...	Checksum/...
NULL		0 B				
.zdebug_ranges		128.1 KB				
.zdebug_pubtypes		39.1 KB				
.zdebug_pubnames		12.9 KB				
.zdebug_loc		378.2 KB				
.zdebug_line		357.9 KB				
.zdebug_info		562.1 KB				
.zdebug_frame		80.9 KB				
.zdebug_abbrev		281 B				
.typelink		7.3 KB				
.text		2.3 MB				
.tbss		0 B				
.symtab		177.3 KB				
.strtab		223.4 KB				
.shstrtab		552 B				
.rodata		1010.2 KB				
.rela.plt	.plt	792 B				
.rela		24 B				
.plt		544 B				
.note.go.buildid	.buildid	100 B				
.noptrdata		216.6 KB				
.noptrbss		0 B				
.itablink		1.9 KB				
.interp		28 B				
.hash		188 B				
.got.plt	.plt	288 B				

Ok I misjudged the code. It looks more like Golang now, because the function names literally have `golang` in them. Welp no easy way out.

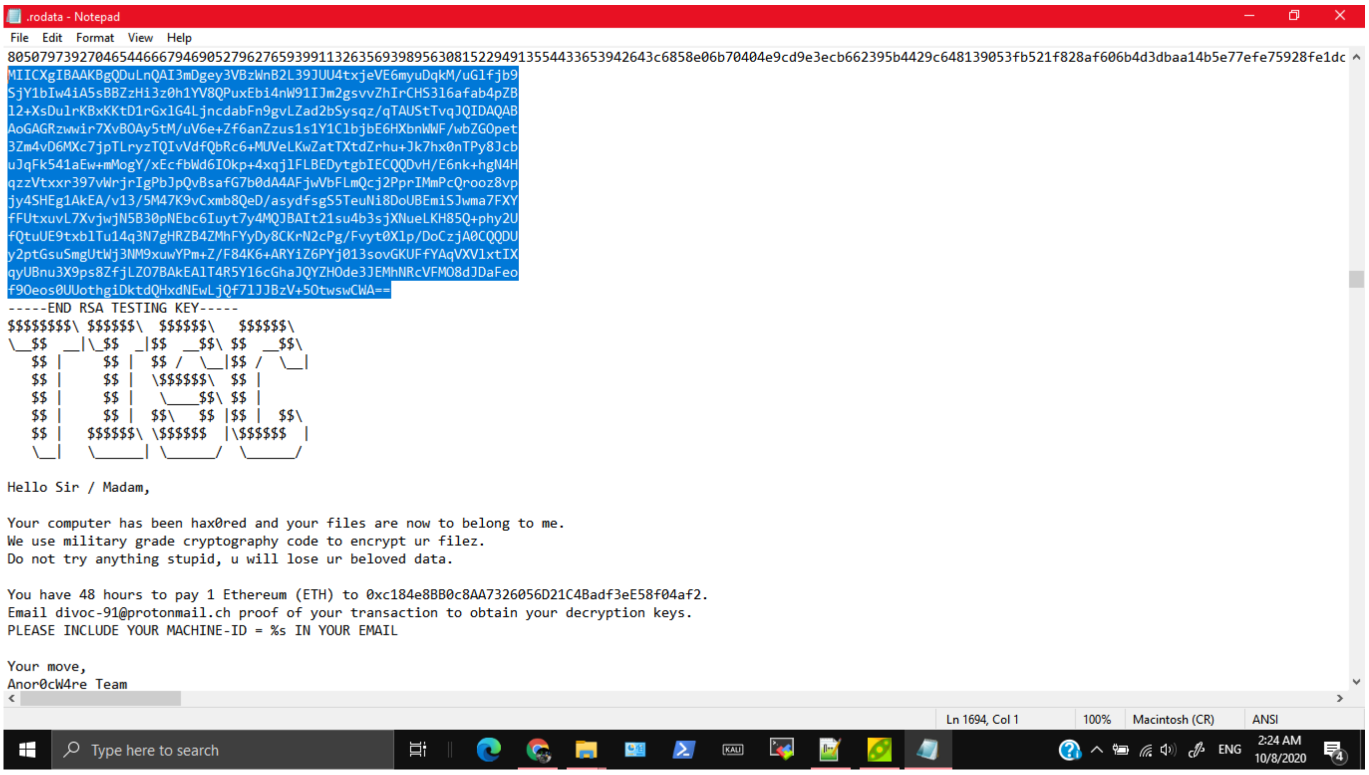

Cutter – C:\Users\zunmu\Documents\Stuff\Github\Solutions\Cyber Security\Capture the Flag Competitions\2020\CSIT The InfoSecurity Challenge\STAGE 2 Find me some keys\anorocware-unpacked
File Edit View Windows Debug Help
Type flag name or address here
Functions
Decompiler
Name
Quick Filter
Auto Refresh
Refresh
Dashboard
Decompiler
Strings
Imports
Search
Disassembly
Graph (sym.vendor_golang.org_x_text_unicode_norm.nextDecomposed)
Hexdump
Decompiler: Ghidra
Type here to search
ENG
2:17 AM
10/8/2020

```
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.vendor_golang.org_x_text_u
sym.x_cgo_init
sym.x_cgo_munmap
sym.x_cgo_notify_runtime_init_d
sym.x_cgo_set_context_function
sym.x_cgo_sys_thread_create
sym.x_cgo_unsetenv
```

```
// WARNING: [r2ghidra] Var arg_f0h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_188h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_190h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_198h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_1a0h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_98h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_a8h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_b0h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_a0h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_b8h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_c8h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_d0h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_d8h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_e0h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_8h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_10h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_40h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_c0h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_18h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_80h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_88h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_90h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_38h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_48h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_50h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_58h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_60h is stack pointer based, which is not supported for decompilation.
// WARNING: [r2ghidra] Var arg_68h is stack pointer based, which is not supported for decompilation.

void vendor/golang.org/x/text/unicode/norm.nextDecomposed(code *placeholder_0, undefined8 placeholder_1, int64_t arg3)
{
    uint64_t *puVar1;
    uint8_t uVar2;
```

Ok back to looking through the zip file. On opening all the files in notepad, I found something interesting scrolling through `.rodata`



```
805079739270465446667946905279627659399113263569398956308152294913554433653942643c6858e06b70404e9cd9e3ecb662395b4429c648139053fb521f828af606b4d3dbaa14b5e77efe75928fe1dc
MIICXgIBAAKBgQDuLnQAI3mDgey3VBzWnB2L39JUU4txjeVE6myuDqkM/uGlfjb9
SjY1bIw4iA5sBBZzHi3z0h1YV8QPuxEbi4nW91IJm2gsvvZhIrCHS3l6afab4pZB
l2+XsDulrKBxKKtD1rGxlG4LjncdabFn9gvLZad2bSysqz/qTAUStTvqJQIDAQAB
AoGAGRzwwir7XvBOAy5tM/uV6e+Zf6anZzus1s1Y1ClbjbE6HXbnWWF/wbZGOpet
3Zm4vD6MXc7jpTLryzTQIvVdfQbRc6+MUVeLKwZatTXtdZrhu+Jk7hx0nTPy8Jcb
uJqFk541aEw+mMogY/xEcfbWd6IOkp+4xqjlFLBEDytgbIECQQDvH/E6nk+hgN4H
qzzVtxxr397vWrjrIgPbJpQvBsafG7b0dA4AFjwVbFLmQcj2PprIMmPcQrooz8vp
jy4SHEg1AkEA/v13/5M47K9vCxmb8QeD/asydfsgS5TeuNi8DoUBEmiSJwma7FXY
fFUtxuvL7XvjwjN5B30pNEbc6Iuyt7y4MQJBAIt21su4b3sjXNueLKH85Q+phy2U
fQtuUE9txblTu14q3N7gHRZB4ZMhFYyDy8CKrN2cPg/Fvyt0Xlp/DoCzjA0CQQDU
y2ptGsuSmgUtWj3NM9xuwYPm+Z/F84K6+ARYiZ6PYj013sovGKUFfYAqVXVlxtIX
qyUBnu3X9ps8ZfjLZO7BAkEAlT4R5Yl6cGhaJQYZHOde3JEMhNRcVFMO8dJDaFeo
f9Oeos0UUothgiDktdQHxdNEwLjQf7lJJBzV+5OtwswCWA==
-----END RSA TESTING KEY-----
$$$$$$$$\ $$$$$$\  $$$$$$\   $$$$$$\
\__$$  __|\_$$  _|$$  __$$\ $$  __$$\
   $$ |     $$ |  $$ /  \__|$$ /  \__|
   $$ |     $$ |  \$$$$$$\  $$ |
   $$ |     $$ |   \____$$\ $$ |
   $$ |     $$ |  $$\   $$ |$$ |  $$\
   $$ |   $$$$$$\ \$$$$$$  |\$$$$$$  |
   \__|   \______| \______/  \______/

Hello Sir / Madam,

Your computer has been hax0red and your files are now to belong to me.
We use military grade cryptography code to encrypt ur filez.
Do not try anything stupid, u will lose ur beloved data.

You have 48 hours to pay 1 Ethereum (ETH) to 0xc184e8BB0c8AA7326056D21C4Badf3eE58f04af2.
Email divoc-91@protonmail.ch proof of your transaction to obtain your decryption keys.
PLEASE INCLUDE YOUR MACHINE-ID = %s IN YOUR EMAIL

Your move,
Anor0cW4re Team
```

I took 2.5 hours to reach here so not bad?

Help I'm there but not?

Well now I have the below base 64 data?, so time to hash it with an online tool? This gives me a hash of `c6ef8326e531a776a3a1ee9876103f9e2576f223dfdecf58de31cd603589b73c`

```
MIICXgIBAAKBgQDuLnQAI3mDgey3VBzWnB2L39JUU4txjeVE6myuDqkM/uGlfjb9
SjY1bIw4iA5sBBZzHi3z0h1YV8QPuxEbi4nW91IJm2gsvvZhIrCHS3l6afab4pZB
l2+XsDulrKBxKKtD1rGxlG4LjncdabFn9gvLZad2bSysqz/qTAUStTvqJQIDAQAB
AoGAGRzwwir7XvBOAy5tM/uV6e+Zf6anZzus1s1Y1ClbjbE6HXbnWWF/wbZGOpet
3Zm4vD6MXc7jpTLryzTQIvVdfQbRc6+MUVeLKwZatTXtdZrhu+Jk7hx0nTPy8Jcb
uJqFk541aEw+mMogY/xEcfbWd6IOkp+4xqjlFLBEDytgbIECQQDvH/E6nk+hgN4H
qzzVtxxr397vWrjrIgPbJpQvBsafG7b0dA4AFjwVbFLmQcj2PprIMmPcQrooz8vp
jy4SHEg1AkEA/v13/5M47K9vCxmb8QeD/asydfsgS5TeuNi8DoUBEmiSJwma7FXY
fFUtxuvL7XvjwjN5B30pNEbc6Iuyt7y4MQJBAIt21su4b3sjXNueLKH85Q+phy2U
fQtuUE9txblTu14q3N7gHRZB4ZMhFYyDy8CKrN2cPg/Fvyt0Xlp/DoCzjA0CQQDU
y2ptGsuSmgUtWj3NM9xuwYPm+Z/F84K6+ARYiZ6PYj013sovGKUFfYAqVXVlxtIX
qyUBnu3X9ps8ZfjLZO7BAkEAlT4R5Yl6cGhaJQYZHOde3JEMhNRcVFMO8dJDaFeo
f9Oeos0UUothgiDktdQHxdNEwLjQf7lJJBzV+5OtwswCWA==
```

Without the new lines I get `MIICXgIBAAKBgQDuLnQAI3mDgey3VBzWnB2L39JUU4txjeVE6myuDqkM/uGlfjb9SjY1bIw4iA5sBBZzHi3z0h1YV8QPuxEbi4nW91IJm2gsvvZhIrCHS3l6afab4pZBl2+XsDulrKBxKKtD1rGxlG4LjncdabFn9gvLZad2bSysqz/qTAUStTvqJQIDAQABAoGAGRzwwir7XvBOAy5tM/uV6e+Zf6anZzus1s1Y1ClbjbE6HXbnWWF/wbZGOpet3Zm4vD6MXc7jpTLryzTQIvVdfQbRc6+MUVeLKwZatTXtdZrhu+Jk7hx0nTPy8JcbuJqFk541aEw+mMogY/xEcfbWd6IOkp+4xqjlFLBEDytgbIECQQDvH/E6nk+hgN4HqzzVtxxr397vWrjrIgPbJpQvBsafG7b0dA4AFjwVbFLmQcj2PprIMmPcQrooz8vpjy4SHEg1AkEA/v13/5M47K9vCxmb8QeD/asydfsgS5TeuNi8DoUBEmiSJwma7FXYfFUtxuvL7XvjwjN5B30pNEbc6Iuyt7y4MQJBAIt21su4b3sj`

`XNueLKH85Q+phy2UfQtuUE9txblTu14q3N7gHRZB4ZMhFYyDy8CKrN2cPg/Fvyt0Xlp/DoCzjA0CQQDUy2ptGsuSmgUtWj3NM9xuwYPm+Z/F84K6+ARYiZ6PYj013sovGKUFfYAqVXVlxtIXqyUBnu3X9ps8ZfjLZO7BAkEAlT4R5Yl6cGhaJQYZHOde3JEMhNRcVFMO8dJDaFeof9Oeos0UUothgiDktdQHxdNEwLjQf7lJJBzV+5OtwswCWA==`. This gives me a hash of `867fc2756be891892d56eadd5c275d9d2d0146972b0f4d6decfc5f2ee51a3a5d`

Ok now I'm stuck. I've tried both hashes but both are rejected?

I've tried converting the base64 into raw data, putting that into a file, and doing SHA256 on that but the hash still doesn't work

```
$ echo
"MIICXgIBAAKBgQDuLnQAI3mDgey3VBzWnB2L39JUU4txjeVE6myuDqkM/uGlfjb9SjY1bIw4iA
5sBBZzHi3z0h1YV8QPuxEbi4nW91IJm2gsvvZhIrCHS3l6afab4pZBl2+XsDulrKBxKKtD1rGxl
G4LjncdabFn9gvLZad2bSysqz/qTAUStTvqJQIDAQABAoGAGRzwwir7XvBOAy5tM/uV6e+Zf6an
Zzus1s1Y1ClbjbE6HXbnWWF/wbZGOpet3Zm4vD6MXc7jpTLryzTQIvVdfQbRc6+MUVeLKwZatTX
tdZrhu+Jk7hx0nTPy8JcbuJqFk541aEw+mMogY/xEcfbWd6IOkp+4xqjlFLBEDytgbIECQQDvH/
E6nk+hgN4HqzzVtxxr397vWrjrIgPbJpQvBsafG7b0dA4AFjwVbFLmQcj2PprIMmPcQrooz8vpj
y4SHEg1AkEA/v13/5M47K9vCxmb8QeD/asydfsgS5TeuNi8DoUBEmiSJwma7FXYfFUtxuvL7Xvj
wjN5B30pNEbc6Iuyt7y4MQJBAIt21su4b3sjXNueLKH85Q+phy2UfQtuUE9txblTu14q3N7gHRZ
B4ZMhFYyDy8CKrN2cPg/Fvyt0Xlp/DoCzjA0CQQDUy2ptGsuSmgUtWj3NM9xuwYPm+Z/F84K6+A
RYiZ6PYj013sovGKUFfYAqVXVlxtIXqyUBnu3X9ps8ZfjaLZO7BAkEAlT4R5Yl6cGhaJQYZHOde
3JEMhNRcVFMO8dJDaFeof9Oeos0UUothgiDktdQHxdNEwLjQf7lJJBzV+5OtwswCWA==" |
base64 -d > key
$ file key
key: DER Encoded Key Pair, 1024 bits
$ binwalk key

DECIMAL       HEXADECIMAL     DESCRIPTION
----------------------------------------------------------------------------
-----
0             0x0             Private key in DER format (PKCS header
length: 4, sequence length: 606

$ sha256sum key
2effaa383cfb3430ca040a3159a877b8ac38d933c7a7f529194e1531a3016e07  key
```

Did they want the base 64 string of the file?

```
$ cat .rodata.txt | base64  | sha256sum
9673c4ff9a8094f0ebb36227f1c1fc894b9c9e675ddd526c70e167d0af978b13
```

Nope it still fails. After a while I realised that this doesn't meet the format wanted at all. The headers and footers were `-----BEGIN RSA TESTING KEY-----` and `-----END RSA TESTING KEY-----`. However, they should be `-----BEGIN PUBLIC KEY-----` and `-----END PUBLIC KEY-----`? However, I couldn't find any other string or key, so I was quite confused at this point

Out of a whim I decided to try converting the given base64 example string into a file. To my surprise I got ASCII Text inside. Maybe I just read the question entirely wrongly, and what they wanted was the SHA256 hash of a base64 string containing the Key.

```
$ echo
"LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0NCmMyOXRaU0JpWVhObE5qUWdjM1J5YVc1bklHZH
ZaWE1nYUdWeVpRPT0NCi0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLQ==" | base64 -d >
given-example
$ file given-example
given-example: ASCII text, with CRLF line terminators
$ cat given-example
-----BEGIN PUBLIC KEY-----
c29tZSBiYXNlNjQgc3RyaW5nIGdvZXMgaGVyZQ==
-----END PUBLIC KEY-----
```

I just decided to do the same but for this text

```
-----BEGIN RSA TESTING KEY-----
MIICXgIBAAKBgQDuLnQAI3mDgey3VBzWnB2L39JUU4txjeVE6myuDqkM/uGlfjb9
SjY1bIw4iA5sBBZzHi3z0h1YV8QPuxEbi4nW91IJm2gsvvZhIrCHS3l6afab4pZB
l2+XsDulrKBxKKtD1rGxlG4LjncdabFn9gvLZad2bSysqz/qTAUStTvqJQIDAQAB
AoGAGRzwwir7XvBOAy5tM/uV6e+Zf6anZzus1s1Y1ClbjbE6HXbnWWF/wbZGOpet
3Zm4vD6MXc7jpTLryzTQIvVdfQbRc6+MUVeLKwZatTXtdZrhu+Jk7hx0nTPy8Jcb
uJqFk541aEw+mMogY/xEcfbWd6IOkp+4xqjlFLBEDytgbIECQQDvH/E6nk+hgN4H
qzzVtxxr397vWrjrIgPbJpQvBsafG7b0dA4AFjwVbFLmQcj2PprIMmPcQrooz8vp
jy4SHEg1AkEA/v13/5M47K9vCxmb8QeD/asydfsgS5TeuNi8DoUBEmiSJwma7FXY
fFUtxuvL7XvjwjN5B30pNEbc6Iuyt7y4MQJBAIt21su4b3sjXNueLKH85Q+phy2U
fQtuUE9txblTu14q3N7gHRZB4ZMhFYyDy8CKrN2cPg/Fvyt0Xlp/DoCzjA0CQQDU
y2ptGsuSmgUtWj3NM9xuwYPm+Z/F84K6+ARYiZ6PYj013sovGKUFfYAqVXVlxtIX
qyUBnu3X9ps8ZfjLZO7BAkEAlT4R5Yl6cGhaJQYZHOde3JEMhNRcVFMO8dJDaFeo
f9Oeos0UUothgiDktdQHxdNEwLjQf7lJJBzV+5OtwswCWA==
-----END RSA TESTING KEY-----
```

Nope still fails. Even if I change the header and footer to `-----BEGIN PUBLIC KEY-----` and so on, there is no effect.

Flags I have tried and failed

- `TISC20{d38fa9a877cd20976238a7381f80f73b5ce40330e02adee8045435b569ad8e2c}`
- `TISC20{17d4103d53fd8542fe28c914b92aeeaee05a808927a0a6359c9b4ec45b184b1b}`
- `TISC20{867fc2756be891892d56eadd5c275d9d2d0146972b0f4d6decfc5f2ee51a3a5d}`
- `TISC20{2effaa383cfb3430ca040a3159a877b8ac38d933c7a7f529194e1531a3016e07}`
- `TISC20{8ea0df3e76c622131aaa5b50981093c84e1b886abe7259916bd5423b53c4741f}`
- `TISC20{7f85734b9dcd34e2b20aebeabe184b6f0877f35b47c42474304c15b16375179d}`

Couldn't finish in time

Welp I couldn't finish this in time. Nevertheless, it seemed like an interesting challenge, and I learnt somethings about UPX packing binaries, .rodata, Go binaries (I still couldn't really read the code though) and stuff.