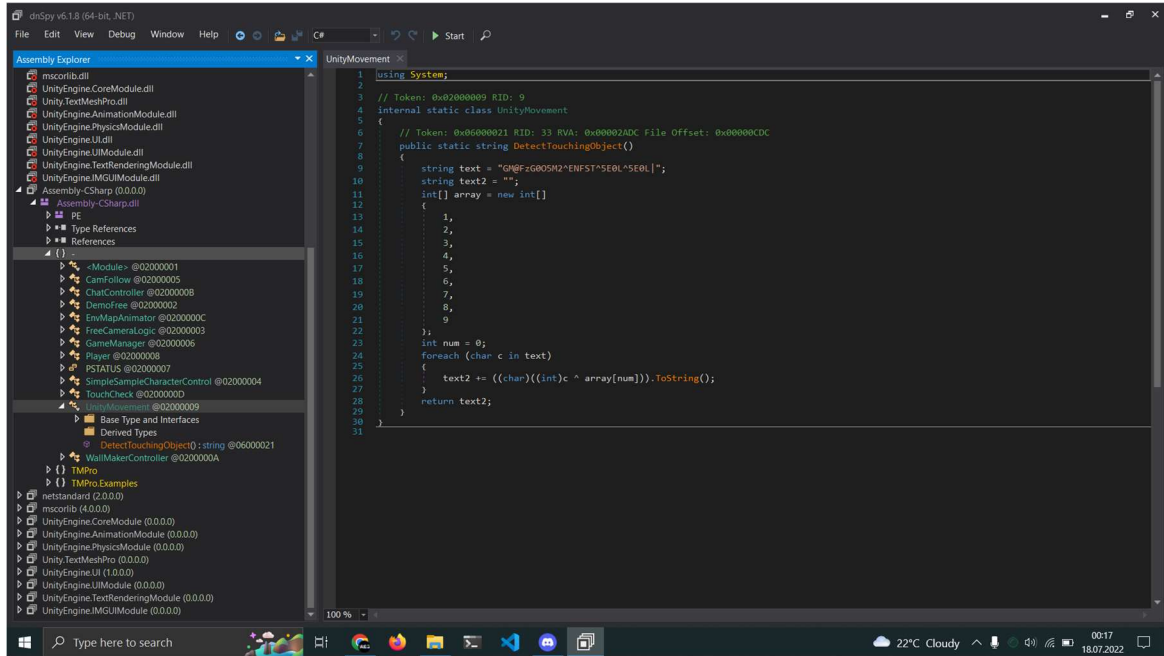


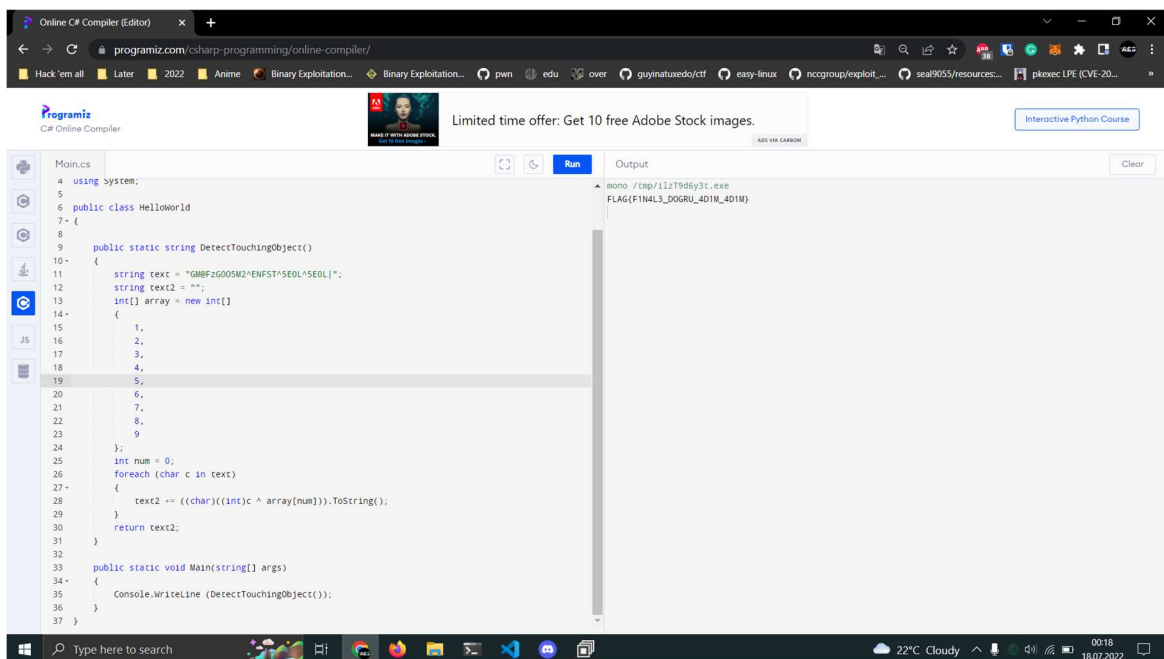
# ForrestGump

Unity tabanlı oyunlarda custom oyun fonksiyonları Managed/Assembly-CSharp.dll'da saklanır. Bu yüzden oyunun fonksiyonlarını görmek için bu dll'i dnSpy adlı programda açıyoruz.



Unity C# ile yazıldığından dolayı dnSpy C#/.NET uygulamalarını decompile edebilmektedir. Fonksiyonlara göz attığımız UnityMovement adlı bir fonksiyon görmekteyiz.

Biraz incelediğimiz zaman aslında burada bir XOR şifrelemesi olduğun görebiliriz. Flagi almak için bu kodu internette ki her hanagi bir Online C# Compiler'da çalıştırarak flagi alabiliriz.



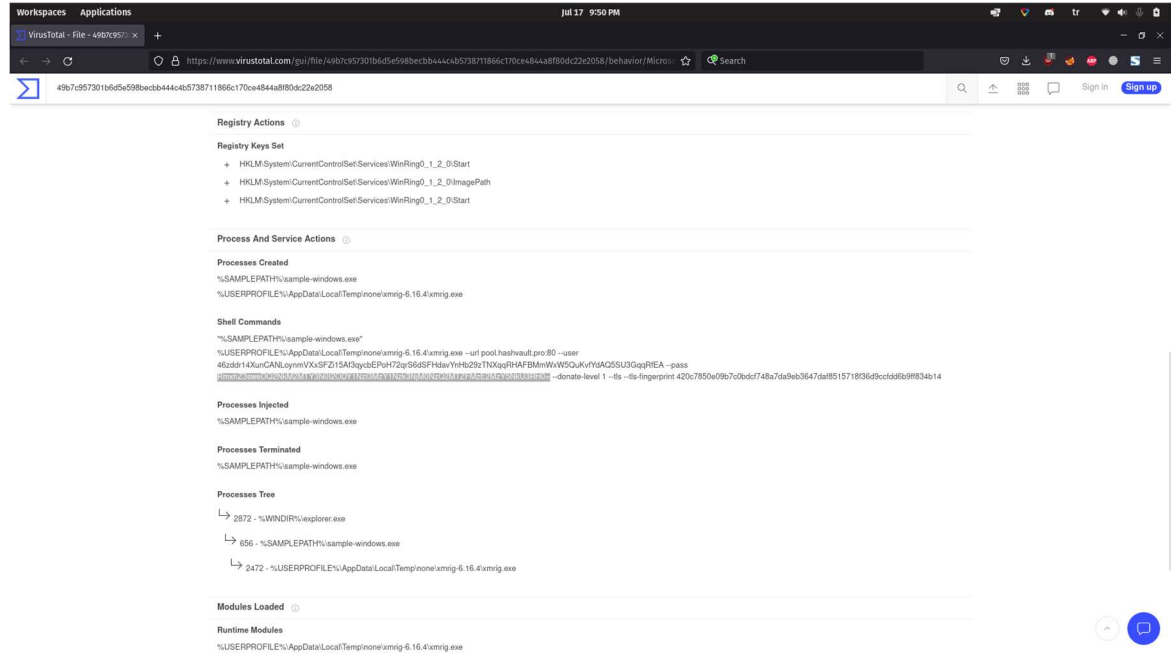
FLAG{F1N4L3\_DOGRU\_4D1M\_4D1M}

## windows.exe

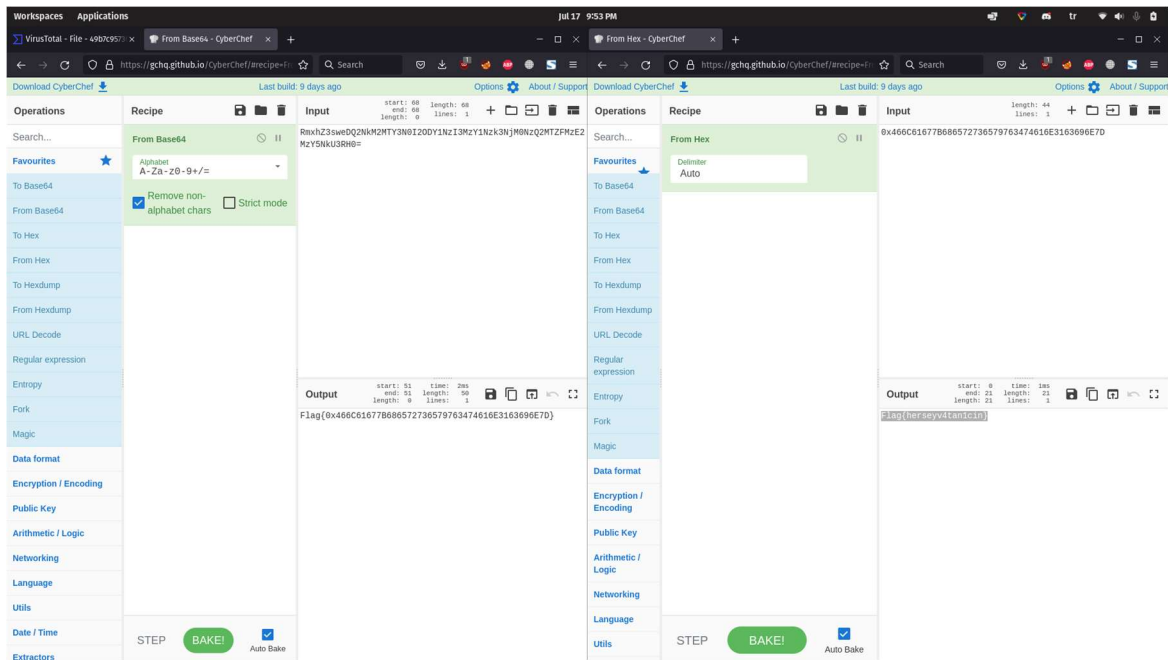
Soru ekindeki EXE dosyasının verilen açıklamadan bir zararlı olabileceği sonucuna vardık ve EXE dosyasının hızlı bir raporunu alabilmek için virustotal.com sitesine verdik.

[VirusTotal Linki](#)

Açılan sekmeler içerisinde Behavior sekmesine geçip EXE dosyasının işlmelerine baktığımızda şüpheli bir işlemi Komut Satırında çalıştırdığı tespit edildiğini gördük.



EXE dosyasının pool.hashvault.pro adlı siteye login olması için gerekli olan bilgileri yolladığını görüyor ve parola kısmındaki ifadenin base64 şifreleme yöntemi ile şifrelendiğini gözlemledik.



Elde edilen flag'in içerisinde bulunan hex verisinin karşılığının ise daha farklı bir flag olduğunu görünce nihai flag olarak elde edilen flag cevap olarak girildi.

**Flag{herseyv4tan1cin}**

Görseller:

<https://i.ibb.co/bW3V9gH/Screenshot-from-2022-07-17-21-50-32.png>

<https://i.ibb.co/8zY7cHW/Screenshot-from-2022-07-17-21-53-33.png>

## FunctionBomber

Sorunun ekinde bir ELF dosyası verilmiş ve sorunun açıklamasından ELF dosyasının içeriğinde fazla sayıda fonksiyonlar olduğu sonucuna varıyoruz. Ve statik olarak dosyayı incelediğimizde birbirine benzer 100000 adet doldurma görevinde fonksiyon ile karşılaşyoruz.

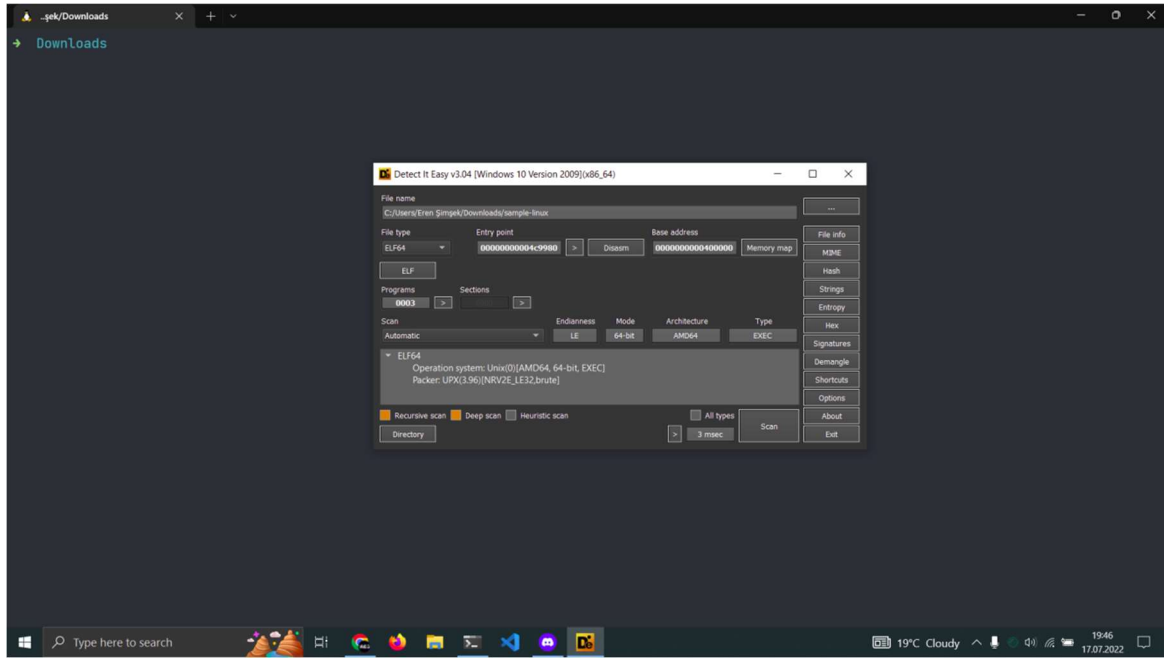
Aralarındaki bir/kaç fonksiyonun farklı olabileceği düşüncesi ile boyutlarını kontrol eden bir script yazıp düşüncemizi doğruluyoruz.



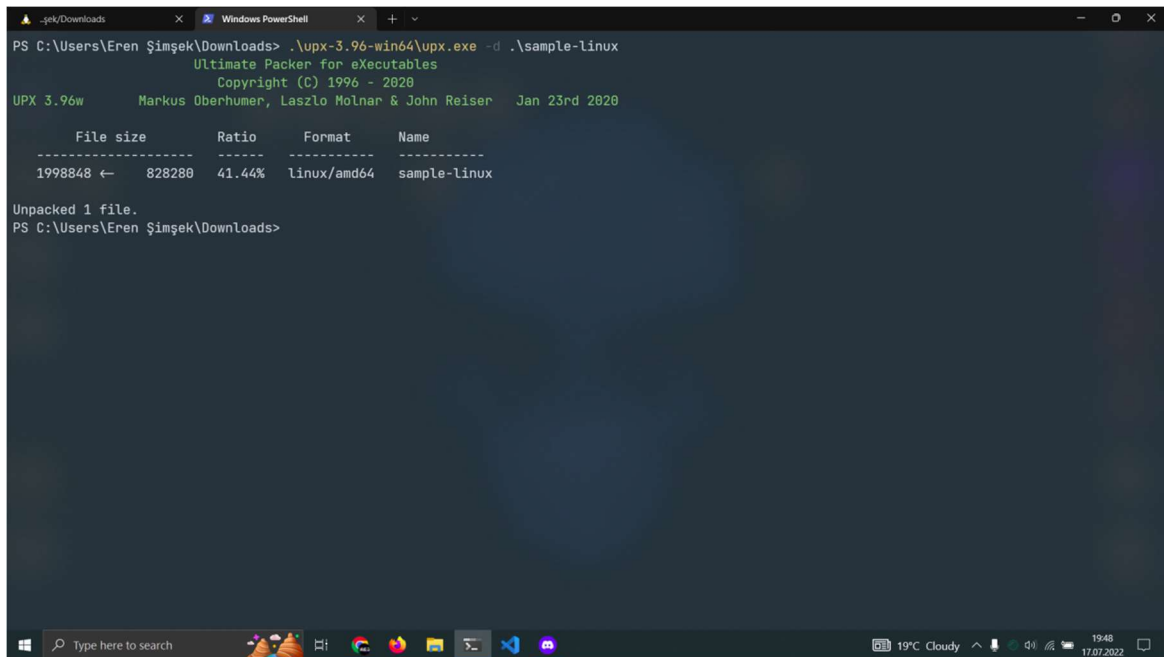
<https://i.ibb.co/0XyWwWQ/Screenshot-from-2022-07-17-22-15-31.png>

# Hesap Makinesi

İlk olarak binarynin hangi packeri kullandığını öğrenmek için Detect It Easy adlı programı kullanıyoruz.



Packerin UPX olduğunu öğrendikten sonra `upx -d sample-linux` ile binari unpack ediyoruz.



Programı çalıştırdığımızda bir hesap makinesi görmekteyiz.

```
./sample-linux

!(Dost Canlisi) hesap makinesine hoş geldiniz :D

Kullanimi oldukca basit! Sadece 2 sayi giriyorsunuz ve islem tamam :DD

+++Menu+++
[1] Toplama
[2] Cikarma
[3] Carpma
[4] Bolme

Seciminiz: 1

1. sayiyi giriniz: 1

2. sayiyi giriniz: 1
Sonuc: 2

!(Dost Canlisi) hesap makinesine hoş geldiniz :D

Kullanimi oldukca basit! Sadece 2 sayi giriyorsunuz ve islem tamam :DD

+++Menu+++
[1] Toplama
[2] Cikarma
[3] Carpma
[4] Bolme

Seciminiz: |
```

Disassembler ile baktığımız zaman stringleri düzgün olarak görememekteydim bu yüzden linuxda ki strings komutunu kullandım. Strings ve grepi kullanarak içerisinde hardcoded olarak saklanan base64 ile encode edilmiş bir string görüyoruz.

```
strings sample-linux | grep -i "hesap makinesi" -A 10 -B 5

runtime: may need to increase max user processes (ulimit -u)
path meido
mod meido (devel)
found bad pointer in Go heap (incorrect use of unsafe or cgo?)runtime: internal error: misuse of lockOSThread/unlockOSThreadABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_runtime.SetFinalizer: pointer not at beginning of allocated blockgo package net: built with netgo build tag; using Go's DNS resolver
bytes.Buffer: UnreadByte: previous operation was not a successful readcannot convert slice with length %y to pointer to array with length %xtoo many concurrent operations on a single file or socket (max 1048575)reflect.Value.Interface: cannot return value obtained from unexported field o
r methodVW0xNGFGbSpjM2RsUkZFvRtdE5NazFVV1R0T01Fa3lUMFJaZUU1NmEzcE9SR013VG5wUmVrNUVXVEZPYTFVeVQmUlpLRTVyu1RKUFZGcERUbXBwKRVl6Rk9La2t6VFhwaWk5V
NTZVVepQVkJzRVRtdE5NazFVV1R0T01Fa3lUMFJaZUU1NmEzcE9SR013VG5wUmVrNUVXVEZPYTFVeVQmUlpLRTVyu1RKUFZGcERUbXBwKRVl6Rk9La2t6VFhwaWk5V

!(Dost Canlisi) hesap makinesine ho
geldiniz :D

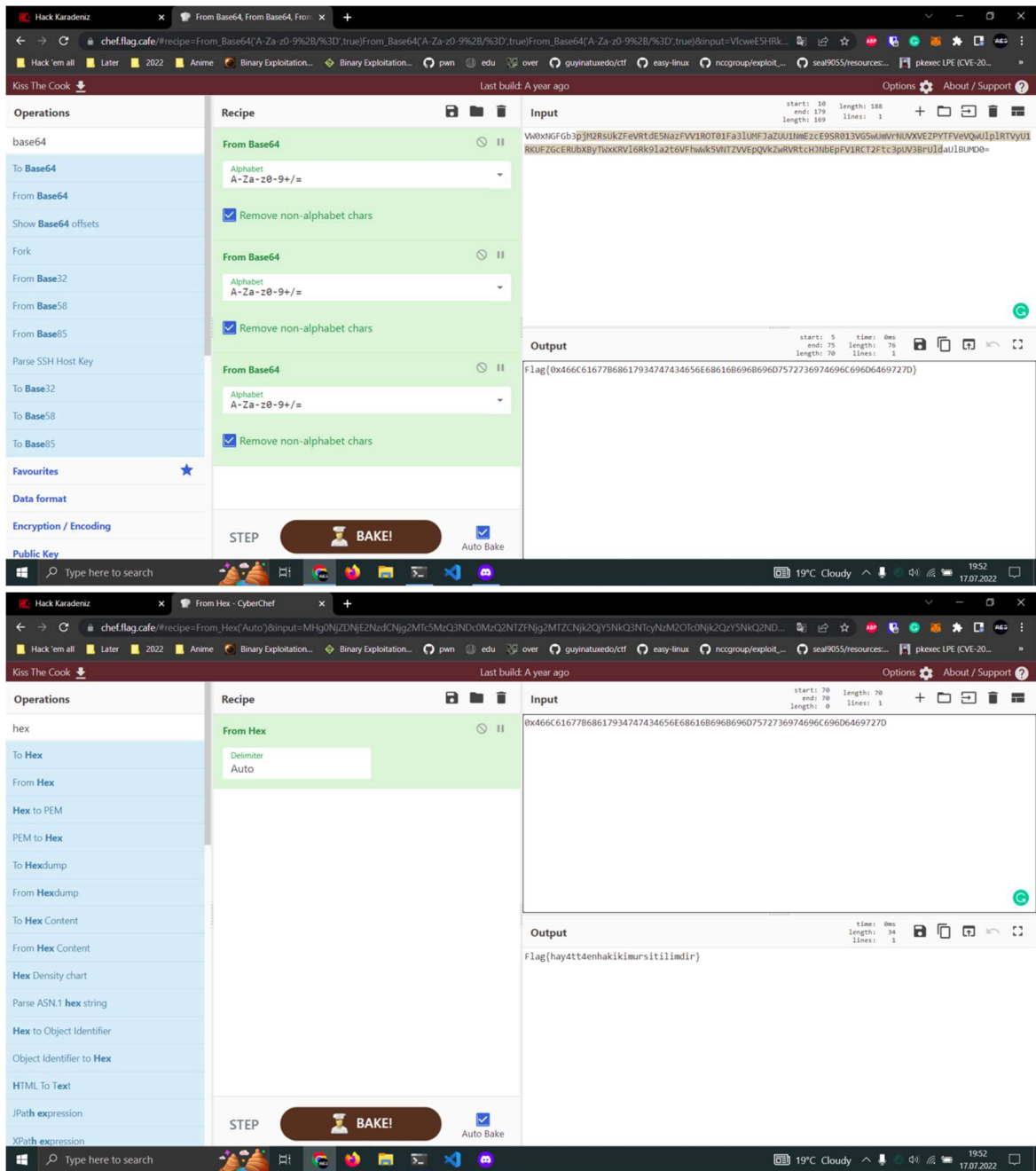
Kullanimi oldukca basit! Sadece 2 sayi giriyorsunuz ve islem tamam :DD

+++Menu+++
[1] Toplama
[2] Cikarma
[3] Carpma
[4] Bolme

0081020304050607080910111213141516171819202122232425262728293031323334353637383940414243444546474849505152535455565758596061626364656667
6869707172737475767778798081828384858687888990919293949596979899

HIHK
Downloads
```

CyberChef'den base64ü bir kaç kere decode ettikten sonra çıkan hex değerini text'e çevirerek flage ulaşıyoruz



**Flag{hay4tt4enhakikimursitilimdir}**

Görseller:

<https://i.ibb.co/r5VcgQr/unknown.png>

<https://i.ibb.co/sPz1VZm/unknown2.png>

<https://i.ibb.co/hLt10rx/unknown3.png>

<https://i.ibb.co/m0ZR2Ld/unknown4.png>

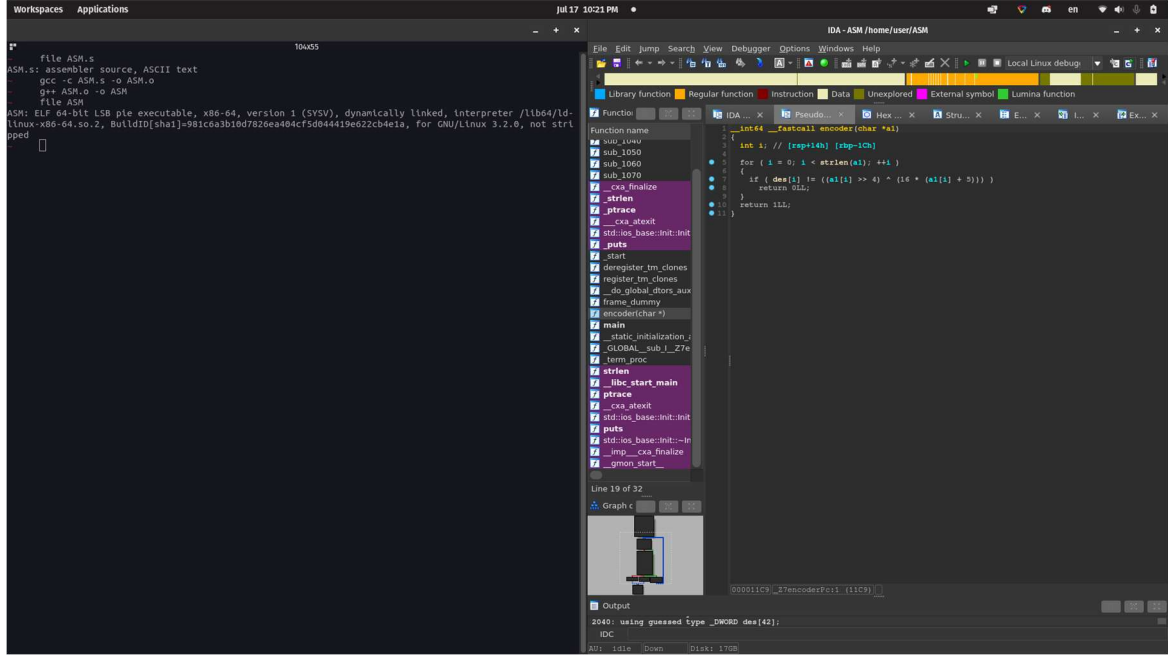
<https://i.ibb.co/JH00sP1/unknown5.png>

<https://i.ibb.co/vP2wb6B/unknown6.png>



# ASM

Sorunun açıklamasındaki bruteforce ifadesi soruya karşı bakış açımızı şekillendirdi ve işleme ekteki assembly dosyasını inceleyerek başladık.



Ekteki assembly dosyasının g++ ile derledik ve statik olarak inceledikten sonra arka tarafta memory adreslerindeki şifreli veriler ile kıyaslayarak argüman verisini kontrol etmekte. Uygulama çalıştırıldığında herhangi bir argüman verilmediğinde veya eşleşmeyen bir argüman girildiğinde hata kodu döndürmektedir. Bu açık kullanılarak bruteforca işlemi çok rahat bir şekilde gerçekleştirilir.



```
Workspaces Applications jul 17 10:33 PM IPython: home/user

In [14]: !./ASR
Invalid input

In [15]: !./ASR a
fail

In [16]: !./ASR H
successful

In [17]:
In [17]: import string
...: at = ''
...: while 1:
...:     for ch in (string.printable):
...:         p = process('ASR' % (ch), level='error')
...:         if b'success' in p.recv():
...:             at += ch
...:             print('###', at)
...:             p.close()
...:             sleep(0.1)
...:
#### H
#### HK
#### HK{
#### HK{K
#### HK{Ka
#### HK{Kar
#### HK{Karad
#### HK{Karade
#### HK{Karaden
#### HK{Karadent
#### HK{Karadent1
#### HK{Karadent1a
#### HK{Karadent1a_
#### HK{Karadent1a_b
#### HK{Karadent1a_b1
#### HK{Karadent1a_b1r
#### HK{Karadent1a_b1r_ha
#### HK{Karadent1a_b1r_ha_hav
#### HK{Karadent1a_b1r_hava
#### HK{Karadent1a_b1r_havas1
#### HK{Karadent1a_b1r_havas1_
#### HK{Karadent1a_b1r_havas1_b1
#### HK{Karadent1a_b1r_havas1_b1r
#### HK{Karadent1a_b1r_havas1_b1r_v
#### HK{Karadent1a_b1r_havas1_b1r_vay
#### HK{Karadent1a_b1r_havas1_b1r_vay1
#### HK{Karadent1a_b1r_havas1_b1r_vayla
#### HK{Karadent1a_b1r_havas1_b1r_vayla1
#### HK{Karadent1a_b1r_havas1_b1r_vayla11
#### HK{Karadent1a_b1r_havas1_b1r_vayla112
#### HK{Karadent1a_b1r_havas1_b1r_vayla1123}
```

HK{Karadenizin\_bir\_havasi\_bir\_yaylasi123}

Görseller:

<https://i.ibb.co/YPDhw7R/Screenshot-from-2022-07-17-22-21-18.png>

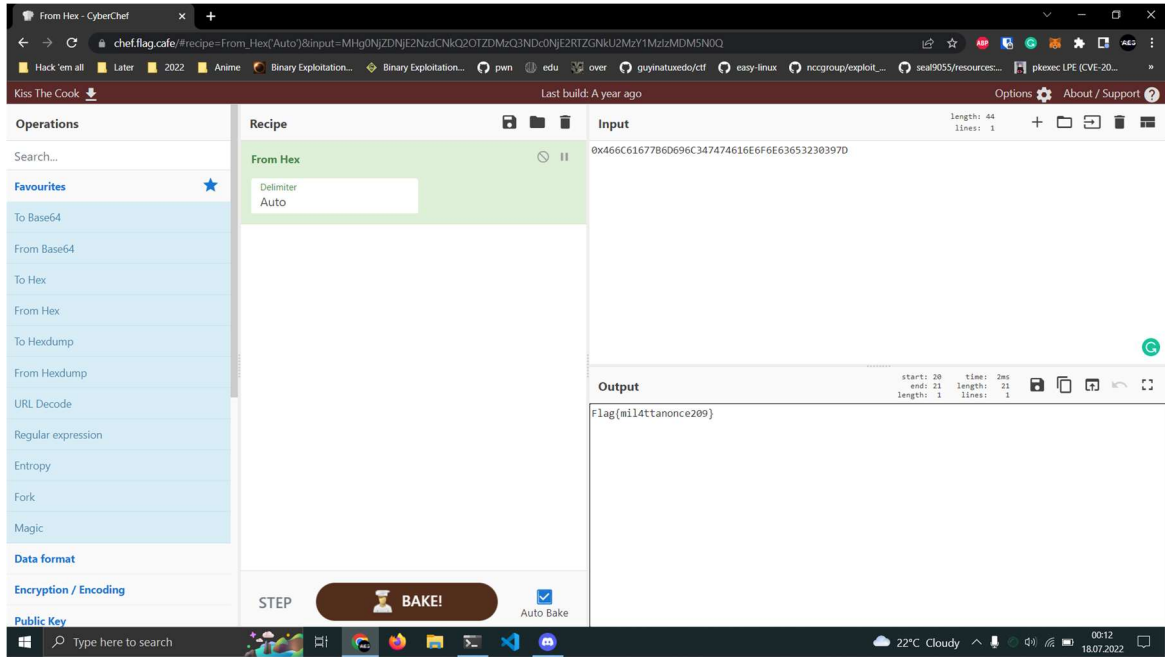
<https://i.ibb.co/LZ3PYVz/Screenshot-from-2022-07-17-22-33-29.png>

## Klasik

Ekte belirtilen dosyayı cmd üzerinde çalıştırınca direkt olarak bizlere flagi vermekte.

```
Windows PowerShell
PS C:\Users\Eren Şimşek\Downloads\HackKaradeniz> .\reverseCtf0ne.exe
Flag{0x466C61677B6D696C347474616E6F6E63653230397D}Parola Girin: |
```

Elde edilen flagin içeriğinde hex ile encodelanmış gerçek flag bulunmaktadır.



**Flag{mil4ttanonce209}**

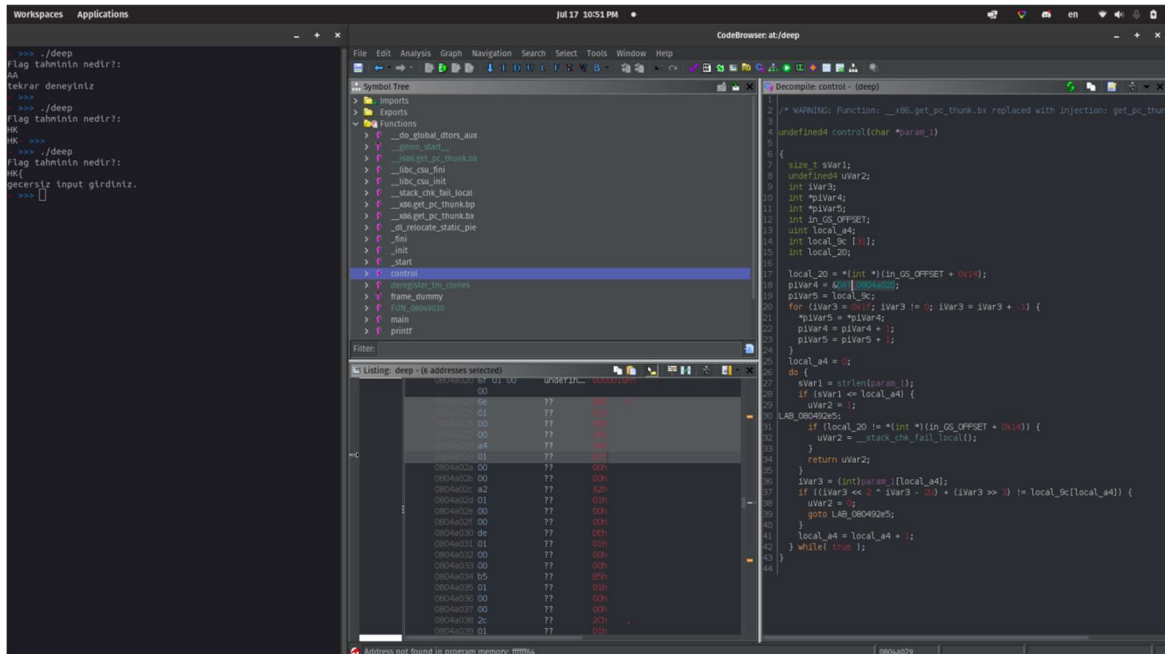
Görseller:

<https://i.ibb.co/L11z88F/unknown0.png>

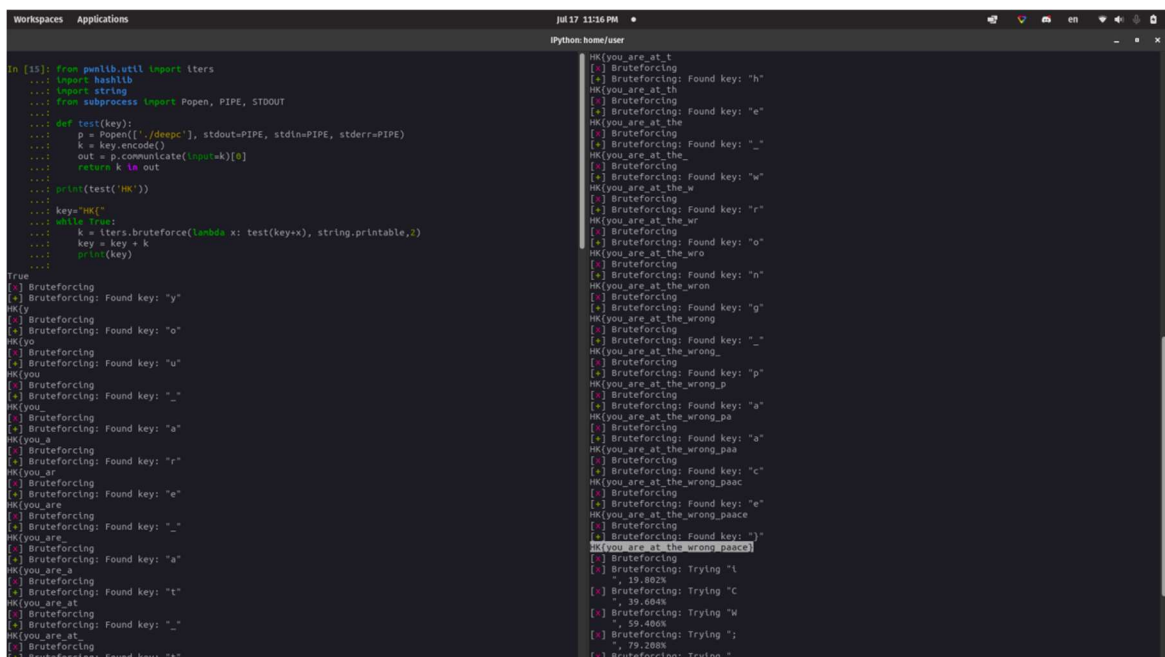
<https://i.ibb.co/pxvtxqr/unknown.png>

## Deep

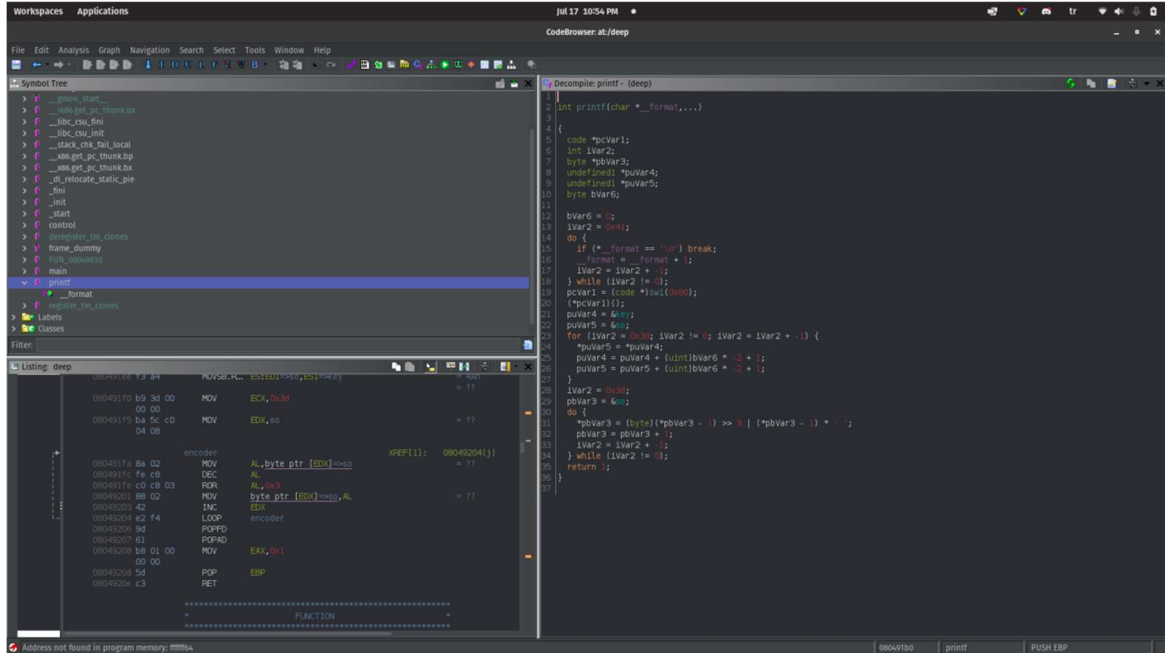
Soru ekinde bir adet 32bit çalıştırılabilir dosya verilmişti, ilk iş olarak dosyayı analiz etmek için çalıştırıldı ve ghidra ile pseudo kodlarını inceledik.



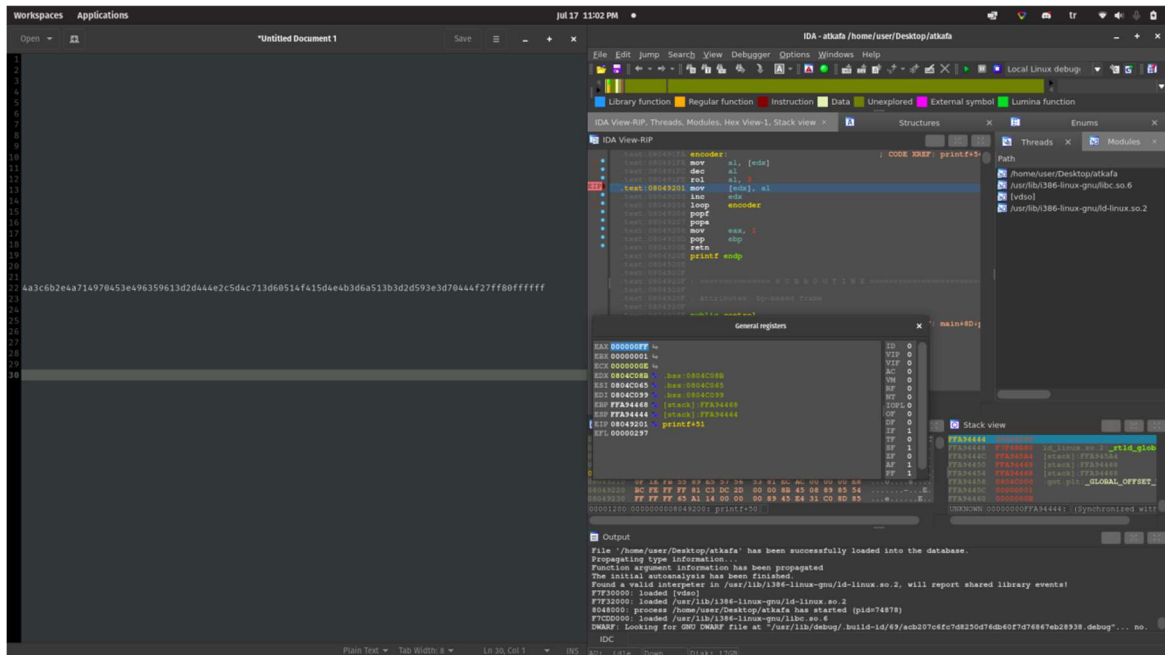
Kodları incelediğimiz zaman alacağı input uzunluğunu maksimum 2 byte olmasını istiyordu. Bizde “en fazla 2 byte alabilir” kod kısmını 34byte alabilecek şekilde patchleyip tekrar deneyebilmek için bruteforce işlemine soktuk.



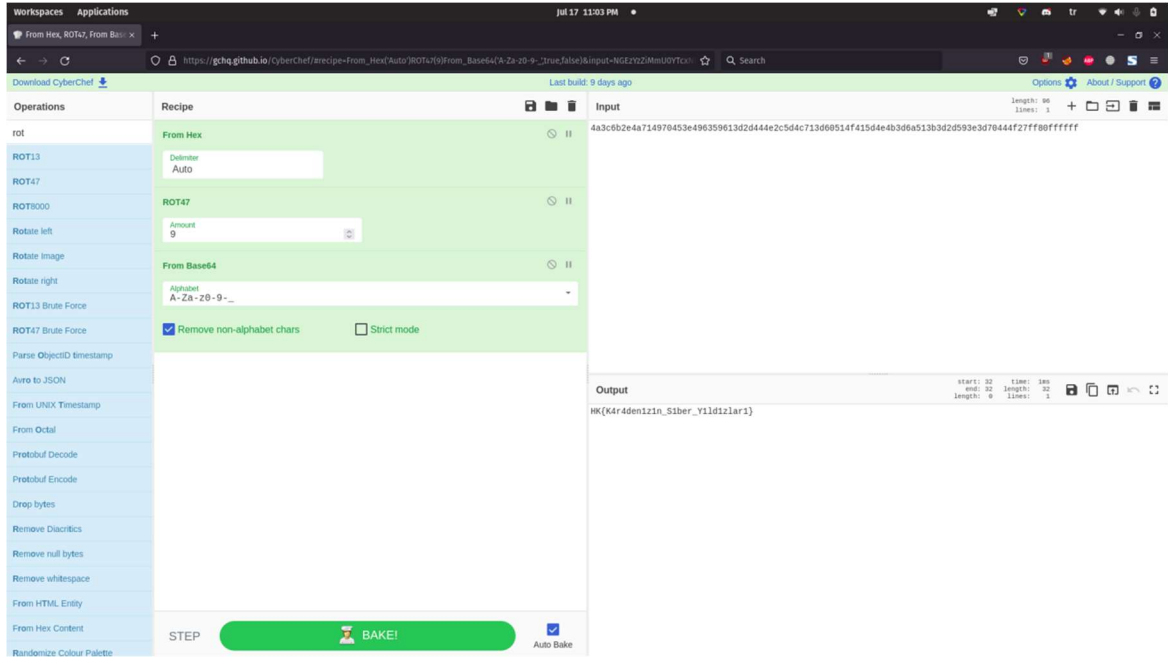
Bize verdiği flag “**HK{you\_are\_at\_the\_wrong\_paace}**” iken, bu flag sistem tarafından kabul edilmedi. Bizde farklı bir kısma yönelmeye başladık. Uygulama klasik printf fonksiyonunu kullanmak yerine kendi printf fonksiyonunu oluşturmuştu. Ve incelediğimiz zaman içeriğinde sadece output vermediğini aynı zamanda bir encryption yaptığını tespit ettik.



Şifrelenen veriyi “key” adındaki memory adresinden düzgünce çıkarabilmek için çokça sayıda reverse algoritmaları denedik ve nihai çözüm şu şekilde gerçekleşti. 3 byte’lık ROR işlemine tutulan veriyi reverseleyebilmek için 3 byte’lık ROL işlemi yapacak şekilde binary dosyasını patch ettik, ve her bir byte için debugger kullanarak “so” memory adresine yazılan verilerini not ettik.



Elde edilen şifrelenmiş veriyi cyberchef yardımı ile flag formatına çevirdik.



**HK{K4r4den1z1n\_S1ber\_Y1ld1zlar1}**

Görseller:

<https://i.ibb.co/Y0VWRD4/Screenshot-from-2022-07-17-22-51-17.png>

<https://i.ibb.co/x6k0M2Q/Screenshot-from-2022-07-17-23-16-36.png>

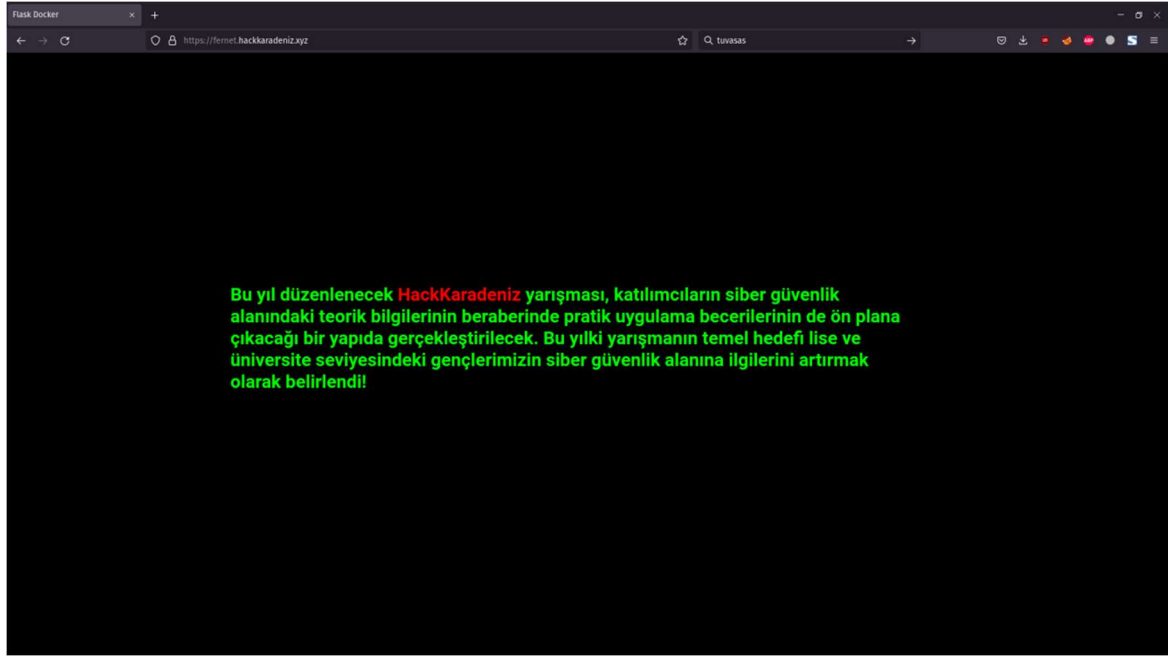
<https://i.ibb.co/sCZ2bhv/Screenshot-from-2022-07-17-22-54-14.png>

<https://i.ibb.co/RcK0pJD/Screenshot-from-2022-07-17-23-03-02.png>

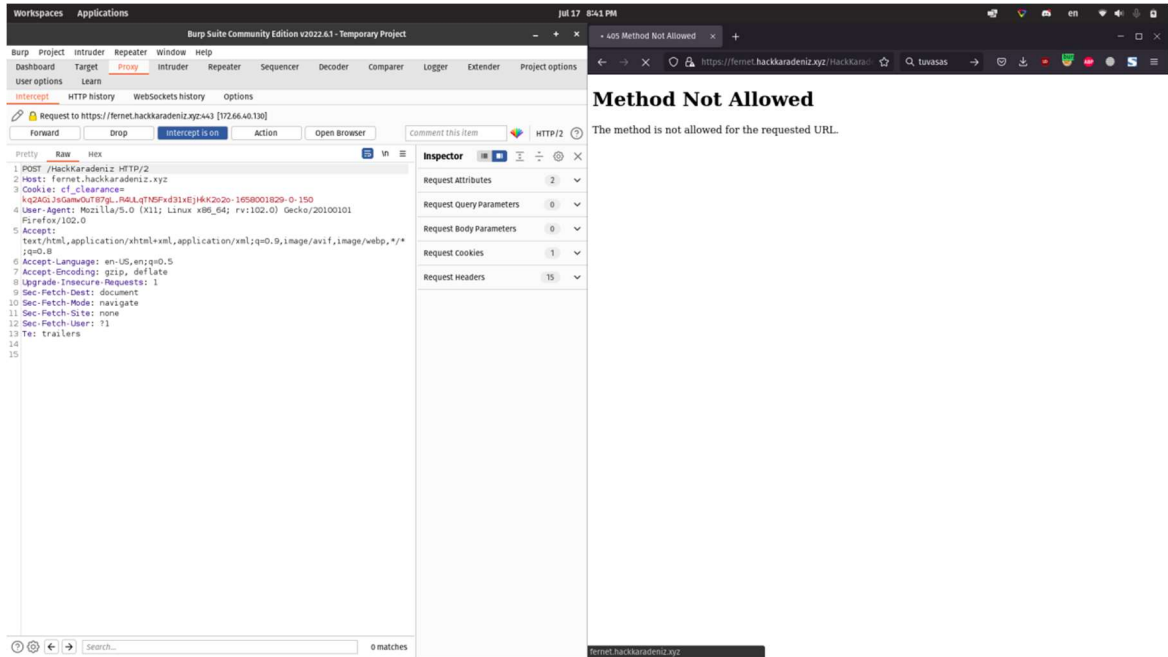
<https://i.ibb.co/gM1pKhd/Screenshot-from-2022-07-17-23-03-35.png>

## Fernet

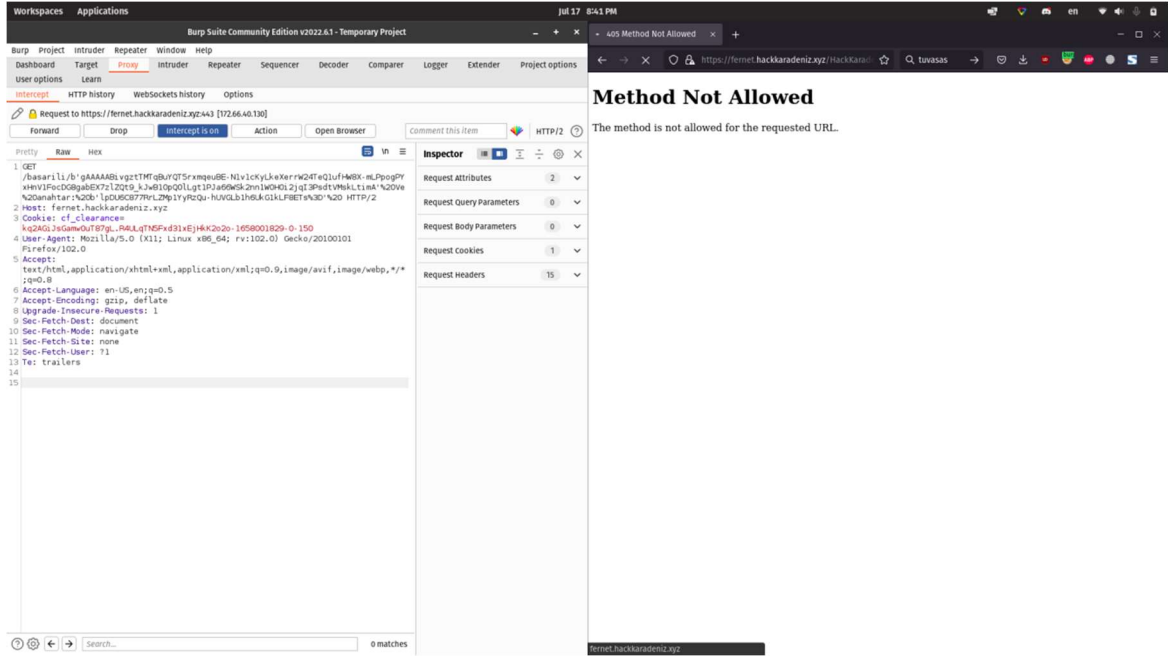
Soru açıklamasında verilen link üzerinden siteye ulaşıyoruz. Karşımıza şu tarz bir görüntü geldi;



Sitedeki açıklamada öne çıkan HackKaradeniz ifadesi referanslanmış bir link iken herhangi bir işlevi bulunmamaktadır. Olası bir yol olarak urle uzantı olarak “HackKaradeniz” girdiğimiz zaman normalde karşılaşılmayacak bir sayfa ile karşılaşıyoruz.

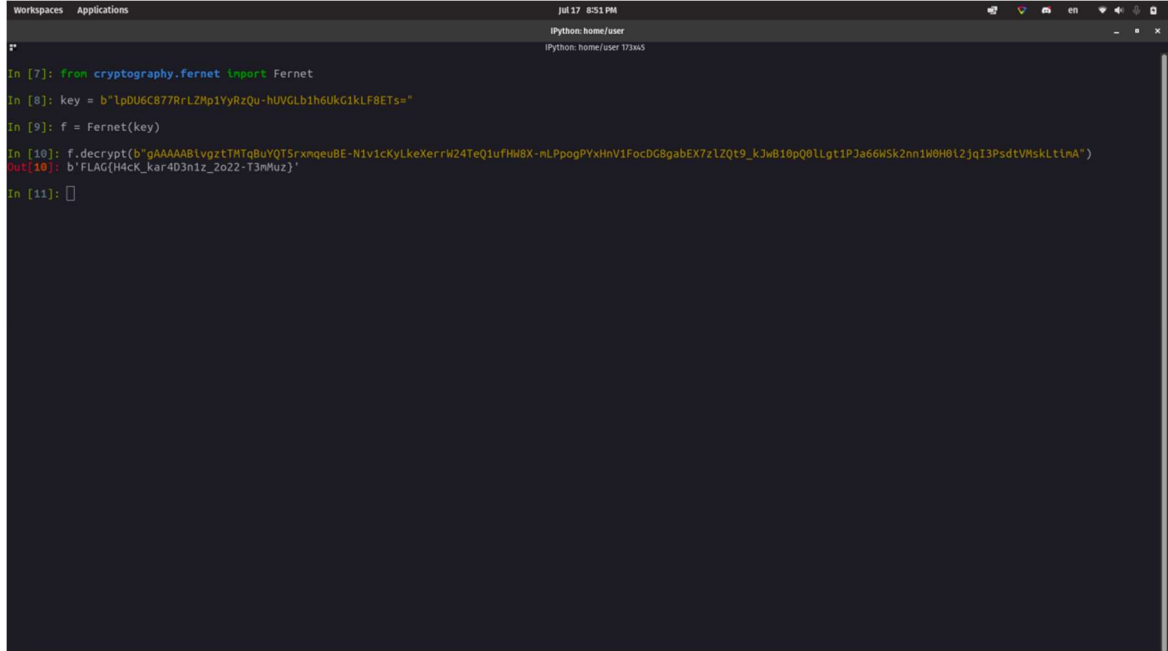


Metodun geçerli olmadığını belirttikten sonra GET metodu yerine POST metodu ile istek göndermeyi deniyoruz. Ve bir redirect ile karşılaşıyoruz.



Redirect bilgileri içerisinde bir adet uzun bir string ve anahtar açıklaması ile aynı şekilde başka bir string ile karşılaşırız.

İnternette yaptığımız araştırmalar sonucunda [Fernet](#) adında pythona uyarlanabilir bir şifreleme algoritması buluyor ve incelemelerini yaptıktan sonra elimizdeki uzun şifrelenmiş string ile anahtar stringini python koduna veriyoruz.



**FLAG{H4cK\_kar4D3n1z\_2o22-T3mMuz}**

Görseller:

<https://i.ibb.co/f9xgTT5/Screenshot-from-2022-07-17-20-31-01.png>



<https://i.ibb.co/yRYqGY8/Screenshot-from-2022-07-17-20-41-30.png>

<https://i.ibb.co/sRcpwJf/Screenshot-from-2022-07-17-20-41-42.png>

<https://i.ibb.co/nkxRxr3/Screenshot-from-2022-07-17-20-51-11.png>