

סדרת אתגרי 2020 Matrix Defense CTF

מאת Dvd848 ,YaakovCohen88

כחלק מקמפיין גיוס של חברת Matrix, נפתח CTF עם שבעה אתגרים מתחומים שונים: Web, Pwn, Forensics ו-Crypto, Reversing. במאמר זה נציג את הפתרון שלנו לאתגרים. האתגרים היו פתוחים החל מחודש פברואר 2020 ולמשך מספר חודשים.

אתגר #1: Behind Blue Eyes (20 נקודות)


Challenge

×

Behind Blue Eyes

20

Not everything meets the eye

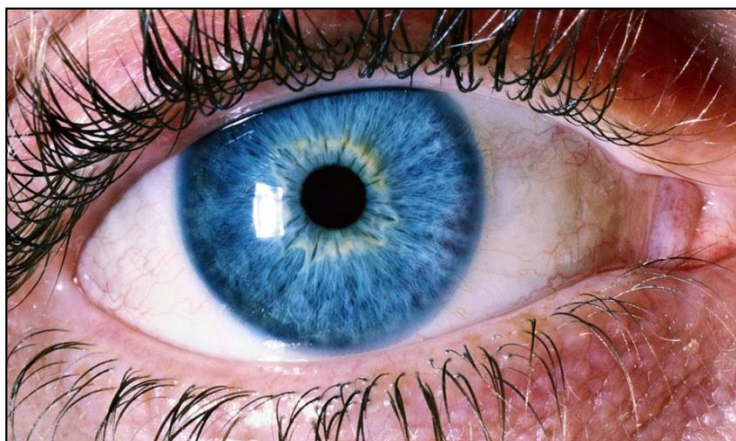
 BehindBlueE...

Flag

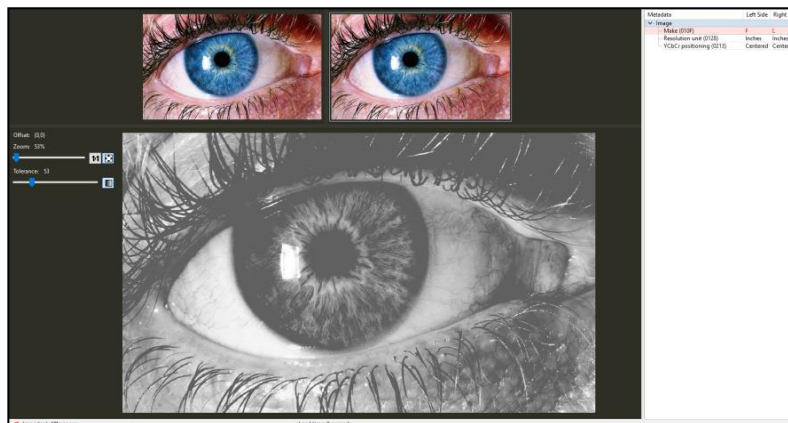
Submit

פתרון:

זהו אתגר חימום. לאתגר צורפו תריסר תמונות זהות (לכאורה) שבמרכזן עין כחולה:



אם קיבלנו תריסר עותקים סימן שכנראה יש ביניהם הבדל כלשהו למרות הכל. דרך אחת להשוות בין שתי תמונות היא באמצעות תוכנת Beyond Compare - תוכנה להשוואה בין קבצים שתומכת במגוון פורמטים (התוכנה לא חנימית אבל היא שווה כל שקל). נשווה בין שני הקבצים הראשונים ונקבל את התוצאה הבאה:



החלק העליון של המסך מראה את שתי התמונות המקוריות ביניהן אנחנו משווים. החלק התחתון מראה את האיחוד של שתי התמונות. אם היה הבדל בתמונה עצמה, הוא היה מוגדש באמצעות סימון צבעוני בתמונה התחתונה. אולם, כפי שניתן לראות, התמונה התחתונה אפורה לחלוטין ולא כוללת הדגשה של הבדל כלשהו. מה כן שונה? ניתן לראות הבדל בפניה הימנית-עליונה של המסך:

Metadata	Left Side	Right Si...
Image		
Make (010F)	F	L
Resolution unit (0128)	Inches	Inches
YCbCr positioning (0213)	Centered	Centered

חלק זה מתאר את ה-metadata של התמונה - שכבת נתונים נוספת שניתן לשמור בקובץ התמונה בנוסף לייצוג הוויזואלי של התמונה עצמה. למשל, כך ניתן לשמור היכן התמונה צולמה, באיזה תאריך, ומי צילם אותה. בעבר התפרסמו מספר מקרים בהם רשויות החוק איתרו מבוקשים באמצעות מידע שחולץ מה-metadata של תמונות שפורסמו על ידי המבוקשים עצמם.

כאן אנו רואים הבדל בשדה ה-Make של התמונות: התמונה הראשונה מכילה את האות F, בעוד השנייה מכילה את האות L.

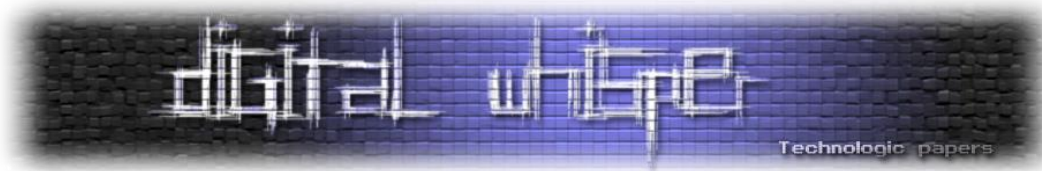
דרך נוחה לצפות במטא-מידע של תמונה כלשהו היא באמצעות תוכנת exiftool:

```
root@kali:/media/sf_CTFs/matrix/BehindBlueEyes# exiftool 0.jpeg | grep Make
Make
: F
root@kali:/media/sf_CTFs/matrix/BehindBlueEyes# exiftool 1.jpeg | grep Make
Make
: L
```

גם כאן אנחנו רואים את ההבדל בשדה Make. כעת נקרא ונאחד את שדה Make של כל התמונות באמצעות הפקודה:

```
root@kali:/media/sf_CTFs/matrix/BehindBlueEyes# for i in {0..11}; do exiftool $i.jpeg | grep Make |
awk '{ printf $3 }'; done
FLAG_THE_WHO
```

הדגל: FLAG_THE_WHO.



אתגר 2: Message From The Dolphins (70 נקודות)

Challenge

Message From The Dolphins

70

The Zip got crumbled and the message got scrambled.

Capture the flag to get the message from the Dolphins.

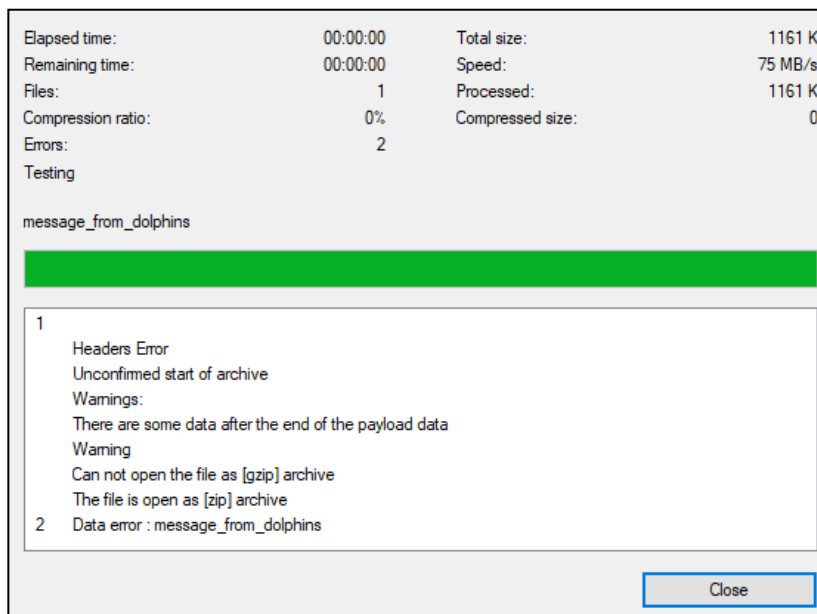
42

Flag

Submit

פתרון:

לאתגר צורף קובץ ארכיון: 42.gz ננסה לפרוס אותו:



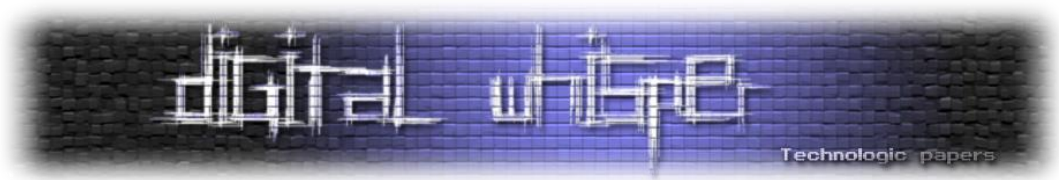
תוכנת 7Zip, שתומכת פחות או יותר בכל פורמט כיווץ שנמצא בשימוש בפועל, לא מצליחה להסתדר עם הקובץ. נראה שיש בו בעיה כלשהי. האם זו סיבה להתייאש, או לחלופין לנסות לנתח את הבעיה ולתקן אותה? ממש לא, פשוט נמצא תוכנה אחרת שמצליחה להתעלם מהבעיה:

```
root@kali:/media/sf_CTFs/matrix/Message_From_The_Dolphins# gunzip -v 42.gz
42.gz:  0.0% -- replaced with 42
```

מה קיבלנו?

```
root@kali:/media/sf_CTFs/matrix/Message_From_The_Dolphins# file 42
42: PNG image data, 1024 x 768, 8-bit/color RGBA, non-interlaced
root@kali:/media/sf_CTFs/matrix/Message_From_The_Dolphins# mv 42 42.png
```

קיבלנו קובץ תמונה.



נפתח את התמונה ונקבל:



Find The Message

גם הפעם הצצה ב-metadata של התמונה תעניק לנו רמז:

```
root@kali:/media/sf_CTFs/matrix/Message_From_The_Dolphins# exiftool 42.png | tail
Components Configuration      : Y, Cb, Cr, -
User Comment                  : Steganography
Flashpix Version              : 0100
GPS Latitude Ref              : North
GPS Longitude Ref             : East
GPS Latitude                  : 32 deg 0' 13.48" N
GPS Longitude                  : 34 deg 52' 22.37" E
GPS Position                   : 32 deg 0' 13.48" N, 34 deg 52' 22.37" E
Image Size                    : 1024x768
Megapixels                     : 0.786
```

שימו לב ל-User Comment - הוא מכיל את הערך "Steganography" שמשמעותו האמנות של הסתרת מסרים סמויים בקבצים תמימים. למשל, הסתרת טקסט על ידי קידודו באמצעות שינוי הערך של הביט התחתון של כל פיקסל (מה שנקרא LSB encoding).

נשתמש בתוכנת zsteg שמחפשת דפוסי steganography נפוצים על מנת לחלץ את הדגל:

```
root@kali:/media/sf_CTFs/matrix/Message_From_The_Dolphins# zsteg 42.png
meta Raw profile type APP1.. text: "\ngeneric profile\n      246\n4578696600004d4d0
9000\n000700000000430323331910100070000000401020300928600070000001500000074a000\n000
0030002000000024500\n000000004000500000000300000000d80000000000000020000000001000000000
b1,rgb,lsb,xy      .. text: "Flag_So_long_and_thanks_for_all_the_fish"
b3,g,msb,xy        .. text: "'\t $B|mH"
b4,g,lsb,xy        .. file: 0420 Alliant virtual executable
b4,bgr,lsb,xy       .. file: PDP-11 UNIX/RT ldp
b4,abgr,msb,xy      .. file: Applesoft BASIC program data, first line number 143
```

אתגר #3: Gotta catch em all (80 נקודות)

Challenge

Gotta catch em all

80

Ash was too busy catching pokemons that he forgot his credentials.

<https://ashmatd.herokuapp.com/>

NOTICE (This is NOT a clue): if you think that the challenge needs a reset - just go to - <https://ashmatd.herokuapp.com/reset>

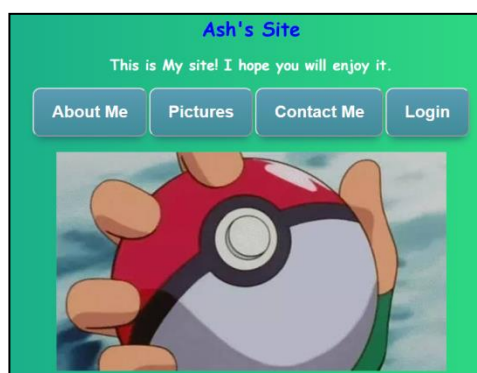
Please reset only if you think that's what holding you back, otherwise it will ruin the game for other players

ALSO - Please don't use brute force or a scan of any kind, it will not get you closet to the solution

Flag

Submit

נכנס לקישור המצורף:



מדובר באתר פשוט המוקדש לפוקימונים. הדפים "אודות" ו"תמונות" מכילים תוכן סטטי, בעוד הדפים "כניסה" ו"צרו קשר" נראים הרבה יותר מעניינים מכיוון שהם מכילים טפסים. דף הכניסה:

Login

username

password

Submit

דף יצירת הקשר:

Contact Me

Leave a message and I will try to help you :)

Title

Body

Submit

מסתבר שאם שולחים קלט ארוך מדי באחד הטפסים, מקבלים חריגה (Exception):

```
Exception
Exception: Given username/password is too big

Traceback (most recent call last)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1836, in __call__
    return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1820, in wsgi_app
    response = self.make_response(self.handle_exception(e))
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1403, in handle_exception
    reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1817, in wsgi_app
    response = self.full_dispatch_request()
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1477, in full_dispatch_request
    rv = self.handle_user_exception(e)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1381, in handle_user_exception
    reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1475, in full_dispatch_request
    rv = self.dispatch_request()
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1461, in dispatch_request
    return self.view_functions[rule.endpoint](**req.view_args)
File "/app/app.py", line 47, in login_attempt
    raise Exception("Given username/password is too big")
Exception: Given username/password is too big
```

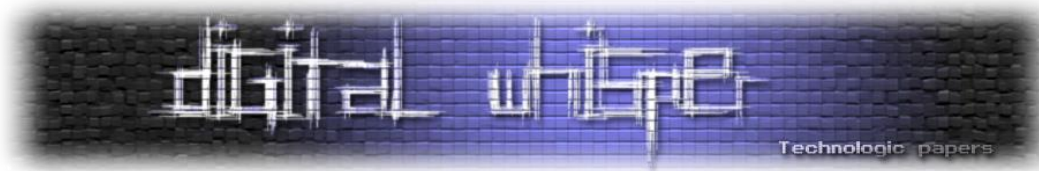
החריגה מדליפה חלק מהמימוש של טופס הכניסה:

```
File "/app/app.py", line 47, in login_attempt
    @app.route('/login', methods=["POST"])
    def login_attempt():
        username = request.form['username']
        password = request.form['password']
        if len(username) > 20 or len(password) > 20:
            raise Exception("Given username/password is too big")
        with open(HOME + 'private/accounts.txt') as f:
            accounts_data = f.readlines()
            accounts = {account.split(':')[0]: account.split(':')[1] for account in accounts_data}
            if username in accounts and accounts[username] == password:
                #Success
```

אנחנו לומדים מקטע הקוד הזה מספר דברים:

- האורך המקסימלי של שם משתמש או סיסמא הוא 20 תווים
 - מסד הנתונים של שמות המשתמש והסיסמאות מנוהל בקובץ טקסט
 - הנתוב היחסי של קובץ הטקסט בתיקיית הבית הוא private/accounts.txt
 - כל זוג של שם משתמש וסיסמא נשמר בשורה נפרדת ומופרד על ידי נקודתיים
- אם נחזור על הפעולה בטופס יצירת הקשר, נקבל הדלפה של קטע קוד נוסף:

```
File "/app/app.py", line 25, in post_message
    body = str(request.form['body'])
    title = title.translate(None, '.*?<>|')
    if len(title) == 0:
        title = "default title"
    if len(request.form['title']) > 30:
        raise Exception("Given title is too big")
    with open(HOME + 'messages/{}.txt'.format(title), "a+") as f:
        f.write(body)
    return render_template('message sent.html')
```



קטע קוד זה מגלה לנו שהטופס לוקח את הכותרת שהגיעה מהמשתמש (title) ומייצרת קובץ טקסט בשם זה תחת תיקיית messages. לתוך הקובץ נשמר תוכן הטופס שהגיע מהמשתמש. החולשה פה היא שהקוד לא מוודא ששדה הכותרת מכיל רק תווים חוקיים, ולכן הוא עשוי להכיל תווים שיש להם משמעות ביצירת נתיב. כלומר, נוכל להכניס כותרת שתאפשר לנו לדרוס את הקובץ accounts.txt שראינו קודם:

Contact Me

Leave a message and I will try to help you :)

../private/accounts

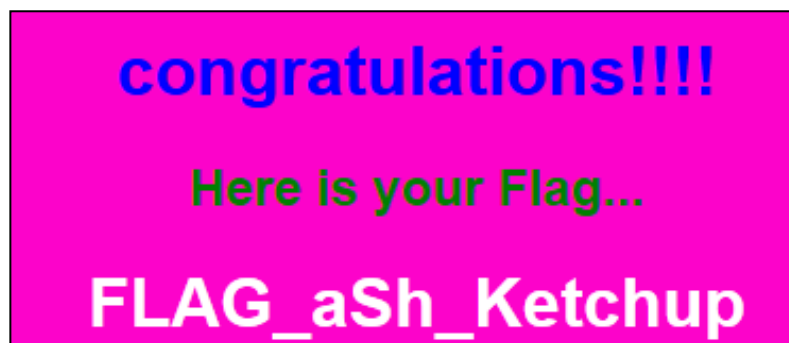
my_username: my_password

Submit

הכותרת הזו תגרום לכך שהקוד ידרוס את התוכן של הקובץ:

HOME + 'messages/../private/accounts.txt'

כעת נוכל להתחבר עם שם המשתמש והסיסמא שדרסנו ולקבל את הדגל:





אתגר #4: Fly Me To The Moon (85 נקודות)

Challenge

Fly Me To The Moon

85

Our Satellite managed to pick up some data from outer space (Aliens? wink wink). We have tried to make some sense of it for a long time, Maybe you can? It's a new year after all...

message

Flag

Submit

פתרון:

אנו מקבלים קובץ בינארי לא מוכר:

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	15	14	13	12	00	00	00	00	2B	C8	E7	17	00	6E	79	6B+π .,nyk
00000010	72	71	45	4D	4B	77	2B	65	57	67	6B	42	45	54	71	67	rqEMKw+eWgkBETqg
00000020	7A	42	46	30	58	75	49	68	74	53	4B	56	46	53	6D	73	zBF0XuIhtSKVFSms
00000030	41	54	73	55	6F	51	6C	4A	38	37	49	59	6E	5A	53	39	ATsUoQLJ87IYnZS9
00000040	6A	4A	64	31	15	14	13	12	01	00	00	00	22	50	E4	4C	jJdl....."PnL
00000050	00	47	46	5A	79	64	39	68	62	35	5A	57	79	72	6D	48	.GFZyd9hb5ZWyrMH
00000060	65	69	2F	6D	4C	53	47	74	56	34	62	76	6C	5A	6D	73	ei/mLSGtV4bvlZms
00000070	54	57	6B	6F	61	49	69	39	7A	34	68	74	4A	58	63	4C	TWkoaIi9z4htJXcL
00000080	79	42	43	78	6F	66	73	2B	15	14	13	12	02	00	00	00	yBCxofs+.....
00000090	44	BE	79	53	00	2F	71	56	31	7A	6F	7A	52	4C	70	6B	D*yS./qVlzoZRLpk
000000A0	58	58	4E	61	76	44	74	44	66	4D	2F	55	2F	50	7A	39	XXNavDtDfM/U/Pz9
000000B0	6B	51	48	38	76	39	64	62	6B	39	48	34	32	2B	62	31	kQH8v9dbk9H42+bl
000000C0	53	53	75	4A	39	57	67	66	77	52	65	41	15	14	13	12	SSuJ9WgfwReA....
000000D0	03	00	00	00	D6	4B	F1	24	00	57	55	6A	31	78	66	4C"Ko\$.WUj1xfL
000000E0	31	48	42	75	4F	65	32	4F	69	61	72	45	61	4E	4E	4F	lHBuOe2OiarEaNNO
000000F0	2B	43	62	48	66	65	54	2F	59	34	6D	39	4F	61	75	6F	+CbHfeT/Y4m9Oauo
00000100	65	4D	67	64	55	51	41	6E	70	4B	67	79	52	6C	63	5A	eMgdUQAnpKgyRlcZ
00000110	15	14	13	12	04	00	00	00	01	BA	B8	47	00	39	6A	79÷,G.9jy

חלק מהקובץ מכיל מחרוזות base64 אך בדיקה מדגמית של המחרוזות לא מעלה כיוון כלשהו.

אם מנסים אורכי תצוגת שורה שונים ב-Hex Editor מגיעים לבסוף לתבנית שנראית קבועה (התבנית מתקבלת עבור אורך שורה של 68 תווים):

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24
00000000	15	14	13	12	00	00	00	00	2B	C8	E7	17	00	6E	79	6B	72	71	45	4D	4B	77	2B	65	57	67	6B	42	45	54	71	67	7A	42	46	30	58
00000044	15	14	13	12	01	00	00	00	22	50	E4	4C	00	47	46	5A	79	64	39	68	62	35	5A	57	79	72	6D	48	65	69	2F	6D	4C	53	47	74	56
00000088	15	14	13	12	02	00	00	00	44	BE	79	53	00	2F	71	56	31	7A	6F	7A	52	4C	70	6B	58	58	4E	61	76	44	74	44	66	4D	2F	55	2F
000000CC	15	14	13	12	03	00	00	00	D6	4B	F1	24	00	57	55	6A	31	78	66	4C	31	48	42	75	4F	65	32	4F	69	61	72	45	61	4E	4E	4F	2B
00000110	15	14	13	12	04	00	00	00	01	BA	B8	47	00	39	6A	79	59	76	5A	77	64	77	42	34	43	46	73	46	41	35	76	48	33	31	43	79	65
00000154	15	14	13	12	05	00	00	00	73	E4	A8	0C	00	32	6C	76	30	2F	2F	32	37	64	39	77	55	6B	70	5A	76	76	79	44	39	39	47	4F	78
00000198	15	14	13	12	06	00	00	00	AA	D1	18	45	00	36	33	7A	6C	76	64	6A	47	59	72	6D	4B	47	57	71	70	42	43	63	30	49	71	51	50
000001DC	15	14	13	12	07	00	00	00	01	80	AF	48	00	61	32	45	61	36	47	67	38	4D	6C	4D	52	73	47	37	30	59	72	51	37	5A	47	4A	38
00000220	15	14	13	12	08	00	00	00	95	01	48	59	00	74	4B	75	71	64	6C	62	79	4E	2F	46	34	56	42	6F	66	50	41	30	72	75	70	71	5A
00000264	15	14	13	12	09	00	00	00	33	1C	E8	3F	00	68	67	79	73	64	58	51	46	74	33	6F	69	45	4D	34	46	4C	4B	33	72	32	72	74	4E
000002A8	15	14	13	12	0A	00	00	00	E3	A7	1F	09	00	39	6C	50	49	58	4D	59	36	79	62	6E	72	4D	38	52	31	66	43	32	6D	66	50	63	59
000002EC	15	14	13	12	0B	00	00	00	23	D2	13	07	00	6F	41	34	4E	71	77	79	70	6D	42	6B	2F	48	2B	52	78	52	48	41	2F	36	38	2F	43
00000330	15	14	13	12	0C	00	00	00	DF	2F	23	01	00	4F	4D	4B	53	50	74	6A	73	35	66	46	73	53	48	4A	53	61	6D	48	48	51	55	38	4D
00000374	15	14	13	12	0D	00	00	00	B0	E1	FF	36	00	2F	41	50	6C	50	45	2F	77	79	4C	74	39	6D	39	56	71	4C	6C	45	4C	78	33	73	42



(התווים 25-43 נחתכו לשם הנוחות). מבחינת ייצוג טקסטואלי התוכן נראה כך:

```
.....+n .nykrqEMKw+eWgkBETqgzBF0XuIhtSKVFSmsATsUoQlJ87IYnZS9jJdl
....."PnL.GFZyd9hb5ZWyrnHei/mLSGtV4bvlZmsTWkoaIi9z4htJXcLyBCxofs+
.....Dkvs./qVlzoZRLpkXXXNavDtDFM/U/Pz9kQH8v9dbk9H42+blSSuJ9WcfwReA
```

אז מה יש לנו פה בעצם?

- נראה שכל רשומה מתחילה עם מספר קסם: 0x12131415
- לאחר מכן מגיע אינדקס רץ באורך ארבעה בתים
- לאחר מכן מגיעים ארבעה בתים עם שונות גבוהה - אולי סוג של checksum?
- לאחר מכן, עוד בית אחד שבדרך כלל מכיל את הערך 0
- לבסוף, שדה נתונים שמכיל מחרוזת base64 (כנראה)

אם נרצה לייצג זאת בתור מבנה ב-C, נקבל:

```
typedef struct
{
    uint32_t magic;      // Equal to 0x12131415
    uint32_t id;         // Running index
    uint32_t checksum;   // Maybe a checksum of an unknown format
    uint8_t  bool_val;   // Unknown
    uint8_t  msg[37];    // Message
} my_struct;
```

עין מדוקדק יותר בערך של הבית הבודד מגלה שיש אך ורק רשומה אחת שבו הוא שווה ל-1:

```
root@kali:/media/sf_CTFs/matrix/Fly_Me_To_The_Moon# cat message.gz | xxd -g 1 -c 68 | awk '$14 == "01" { print $0 }'
0000e42c: 15 14 13 12 5b 03 00 00 00 e1 0b 5e 01 52 6b 78 42 52 31 39 43 55 6b 6c 4f 52 31 39 43 51 55 4e 4c 58 31 4e
4a 54 6b 46 55 55 6b 45 3d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....^..RkxBR1
9CuklOR19COUNTX1NjTKFUUKE=.....
```

אם נקח את מחרוזת ה-base64 המצורפת ונתרגם אותה, נקבל את הדגל:

```
root@kali:/media/sf_CTFs/matrix/Fly_Me_To_The_Moon# echo RkxBR19CUkl0R19CQUNlX1NJTkFUUkE= | base64 -d
FLAG BRING BACK SINATRA
```



אתגר 5: Somewhat SecureBoot (150 נקודות)

Challenge

Somewhat SecureBoot

150

My boss told me to implement a chain of trust mechanism for a remote server boot process. He said: "There are 3 bootloaders - BL1, BL2, BL3. make sure that no one will be able to compromise our images. use a strong integrity solution, You can use SHA256 or something."

So that's exactly what I did. The funny thing is - BL3 was already compromised, I have no idea what BL3 is suppose to do. But I do know that it's something important.

BTW, I don't think my boss knows much about integrity (don't tell him I said it).

nc challenges.ctfd.io 30071

BL2.bin

Flag

Submit

נתחבר לשרת המצורף:

```
root@kali:~/media/sf_CTFs/matrix/Somewhat_SecureBoot# nc challenges.ctfd.io 30071
Bootloader number 1 - Up and running...
Checking environment...
Env is valid.

Setting up memory...
Configuring hardware...
Bootloader 1 completed. Calling next bootloader to load the kernel...

Attempting to load the bootloader /bootloaders/BL2.bin with hash e95e57eff2ae77a55f0559450ffa508a33a21fff0c8a702f8858e682a02281
Bootloader integrity verified successfully.
Bootloader number 2... Up and running...

Configuring kernel...
Setting up process table...
Reserving ports 0-1023...
Initializing scheduler...
Crontab daemon started...
Bootloader 2 finished. Calling next bootloader to open up a shell...

Please enter the desired shell to execute upon startup (press 'a' for /bin/sh)
a
Executing shell: /bin/sh

Attempting to load the bootloader /bootloaders/BL3.bin with hash 20375b4eec91d51308d624660f7e0b4e8cd9fa4f82b5d50b6d38b6d1452c3070
Bootloader integrity compromised... Expected receiving hash: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaa
Exiting...
```

מה שאנחנו רואים פה זה ניסיון לטעון את Boot Loader 3. הבעיה היא שה-hash שלו מחושב בתור ערך שונה מהמצופה. הערך אמור להיות aaa...aaa אבל בפועל מתקבל ערך אחר. לכן, ה-shell שביקשנו לא נפתח.

נטען את קובץ ההרצה שקיבלנו באמצעות Ghidra ובבדוק את הפלט של ה-Decompiler. שם, ניתן לראות שרוב התוכן ממומש בתוך פונקציית main.



תחילה, מוגדרים מספר משתנים מקומיים:

```
char bootloader3_str [21];
char user_input [32];
char expected_hash [65];
```

לאחר מכן, המשתנה expected_hash מאותחל עם aaa...aaa:

```
expected_hash._0_8_ = 0x6161616161616161;
expected_hash._8_8_ = 0x6161616161616161;
expected_hash._16_8_ = 0x6161616161616161;
expected_hash._24_8_ = 0x6161616161616161;
expected_hash._32_8_ = 0x6161616161616161;
expected_hash._40_8_ = 0x6161616161616161;
expected_hash._48_8_ = 0x6161616161616161;
expected_hash._56_8_ = 0x6161616161616161;
expected_hash[64] = '\0';
```

לאחר מספר הדפסות, התוכנה מבקשת קלט מהמשתמש:

```
puts("Please enter the desired shell to execute upon startup (press \'a\' for
/bin/sh)\n");
fflush(stdout);
__isoc99_scanf("%s",user_input);
iVar2 = strcmp(user_input,"a");
if (iVar2 == 0) {
    user_input._0_8_ = 29400045130965551;
}
```

ניתן לראות שהתוכנה קוראת קלט לתוך משתנה user_input באמצעות הפונקציה scanf. כמו כן, ניתן לראות שאין בקריאה לפונקציה שום רמז אודות אורך המשתנה שאליו יוכנס הקלט, מה שאומר ש-scanf תפסיק לקרוא קלט מהמשתמש רק כאשר היא תיתקל בתו whitespace. כלומר, ניתן להשתמש בקריאה זו על מנת לבצע buffer overflow ולדרוס משתנה אחר.

למזלנו, המשתנה שאותו ניתן לדרוס הוא expected_hash, מה שאומר שנוכל להשפיע על הבדיקה שמגיעה אחר כך:

```
ppcVar3 = read_firmware(bootloader3_str);
printf("Attempting to load the bootloader %s with hash %s\n",*ppcVar3,ppcVar3[1]);
fflush(stdout);
iVar2 = strcmp(ppcVar3[1],expected_hash);
if (iVar2 == 0) {
    puts("Bootloader integrity verified successfully.");
    fflush(stdout);
    load(ppcVar3);
}
else {
    printf("Bootloader integrity compromised... Expected receiving hash: %s\nExiting...\n",expected_hash);
    fflush(stdout);
    destroy_bootloader(ppcVar3);
}
```



אנחנו יודעים מהו ה-hash בפועל של ה-bootloader. נשתמש בערך זה על מנת לדרוס את ה-expected_hash ונוכל לעבור את ההשוואה:

```
root@kali:/media/sf_CTFs/matrix/Somewhat_SecureBoot# python3 -c "print ('b'*32 + '20375b4eec91d51308d624660f7e0b4e8cd9fa4f82b5d50b6d38b6d1452c3070') | nc challenges.ctfd.io 30071"
Bootloader number 1 - Up and running...
Checking environment...
Env is valid.

Setting up memory...
Configuring hardware...
Bootloader 1 completed. Calling next bootloader to load the kernel...

Attempting to load the bootloader /bootloaders/BL2.bin with hash e95e57eff2ae77a55f0559450ffafb508a33a21fff0c8a702f8858e682a02281
Bootloader integrity verified successfully.
Bootloader number 2... Up and running...

Configuring kernel...
Setting up process table...
Reserving ports 0-1023...
Initializing scheduler...
Crontab daemon started...
Bootloader 2 finished. Calling next bootloader to open up a shell...

Please enter the desired shell to execute upon startup (press 'a' for /bin/sh)

Executing shell: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb20375b4eec91d51308d624660f7e0b4e8cd9fa4f82b5d50b6d38b6d1452c3070

Attempting to load the bootloader /bootloaders/BL3.bin with hash 20375b4eec91d51308d624660f7e0b4e8cd9fa4f82b5d50b6d38b6d1452c3070
Bootloader integrity verified successfully.
Bootloader number 3... Up and running...
Bootloader number 3 - Finished.

System is fully loaded and ready to use. The time is: 2020-03-10,19:38:48.
Welcome.
Congrats! FLAG_b00t_c0mpleted
```




אתגר #6: CamelCase (175 נקודות)

Challenge

CamelCase

175

Alice attempted to send Bob her encrypted flag over an insecure network channel. Unfortunately, Bob hadn't received her message correctly. She tried to encrypt the same flag and send it to Bob again. Carelessly enough, she reused her random bits for the second encryption, naively assuming her simple and efficient linear function of the original randomness generated truly random bits (with the form $f(x)=ax+b$). An eavesdropper has captured the two ciphers, and it is up to you to retrieve the flag.

Camel.zip

Flag

Submit

פתרון:

אנחנו מקבלים קובץ ארכיון עם מספר קבצי טקסט:

```
root@kali:/media/sf_CTFs/matrix/CamelCase/Camel# ls
details.txt  enc1  enc2
root@kali:/media/sf_CTFs/matrix/CamelCase/Camel# ls enc1
0.txt  11.txt  13.txt  15.txt  17.txt  19.txt  20.txt  22.txt  24.txt  3.txt  5.txt  7.txt  9.txt
10.txt  12.txt  14.txt  16.txt  18.txt  1.txt  21.txt  23.txt  2.txt  4.txt  6.txt  8.txt
root@kali:/media/sf_CTFs/matrix/CamelCase/Camel# ls enc2
0.txt  11.txt  13.txt  15.txt  17.txt  19.txt  20.txt  22.txt  24.txt  3.txt  5.txt  7.txt  9.txt
10.txt  12.txt  14.txt  16.txt  18.txt  1.txt  21.txt  23.txt  2.txt  4.txt  6.txt  8.txt
```

הקובץ details.txt מכיל את התוכן הבא:

```
root@kali:/media/sf_CTFs/matrix/CamelCase/Camel# cat details.txt && echo
q: 152871471657632351370080285966569305872910029684999383757718597560660696292043021501818591583896441093221262574940955
568908605354899083774467939670101882574152087670760348311597609704580106114746180425765515742210551990876619209582167486
148632819497795640798995338204874439231771371145977215524548029682689081
generator: 3666349124397068204714019214122128513676048872812799237332246573113973065582939707652247300845021029736514832
018208358019539371760616620271697031572918598021898314908694686444077571971204225430515790733581097968670453134283079764
0673605246577229552444420262221538684487141010130159030104861473030951021412841
public key: 1100967330597828785037734673493192806022803054650865445600313411816013966268162522563634357913364825338629531
885173306011475388873171910781099217716510906770977860144396462254185820857377849690107039226518519189825831859321182619
80125435797193381932031798376926559470337119492326542050969119495186180201932737
```

כל אחד משאר הקבצים מכיל צמד מספרים, לדוגמה:

```
root@kali:/media/sf_CTFs/matrix/CamelCase/Camel# cat enc1/0.txt && echo
(10525733172244703598885878907687050909154604301550549594697514241979029661423740912069162478094911466262805603276900593
193567035946690084905560570538950243470726550263841349529525308922285422698866664745945875645682718901385202453196602096
1344137840857732342976428560692415596028125551456213440612181862052698, 812963664467898358038539659596556524701107802083
737340843447788881923887678825802062988886371373491283515248389058445760064517829584507691428260183138167288895644179248
174936643465006453016171622156805636131106819518320665155031488995548008367699606042316433991961403841148477914812405742
0683038129440084018910)
```

הנתונים האלו, יחד עם שם האתגר, מצביעים על כך שמדובר ב**צופן אל-גמאל**:

הצפנת אל גמאל (ElGamal encryption) היא שיטת הצפנה אסימטרית אקראית שהומצאה ב-1984 על ידי טאהר אל-גמאל, קריפטוגרף אמריקאי ממוצא מצרי. בדומה לפרוטוקול דיפי-הלמן, ביטחונה מסתמך על הקושי המשווער שבבעיית הלוגריתם הברידי ובעיית דיפי-הלמן. היא יכולה לשמש הן להצפנה והן לחתימה דיגיטלית.



שיטת הצפנה זו מתבצעת מעל חבורה ציקלית סופית. במקרה שלנו, נראה שמדובר בחבורה הכפולית \mathbb{Z}_q^* של השלמים החיוביים מודולו q , כאשר q הינו מספר ראשוני בטוח, כלומר מתקיים של- $(q - 1)$ ישנו לפחות גורם ראשוני אחד גדול.

לכל חבורה כזו ישנה מספר שנקרא יוצר ומסומן ב- g . המספר הזה הוא מספר שבאמצעותו ניתן לייצר כל מספר בחבורה:

$$\mathbb{Z}_q^* = \{g^0, g^1, g^2, \dots, g^{q-1}\}$$

למשל, אם ניקח את \mathbb{Z}_5^* , נגלה ש-2 משמש בתור היוצר של חבורה זו:

```
>>> print (set([pow(2, i, 5) for i in range(5)]))
{1, 2, 3, 4}
```

כעת, כדי לייצר את המפתח הפרטי, בוחרים מספר אקראי x (מודולו q) ומפרסמים את השלישייה: $(q, g, h = g^x)$

בקובץ הטקסט שקיבלנו, g נקרא generator ו- h נקרא public key.

את המספר x שומרים בסוד בתור המפתח הפרטי.

למרות שאנחנו יודעים מהו g ומהו g^x , מציאת x עצמו נחשבת [קשה](#).

על מנת להצפין הודעה m , מגדילים מספר אקראי k (מודולו q) ומפרסמים את:

$$(c_1 = g^k, c_2 = m \cdot h^k)$$

בבעיה שלנו, נתון לנו שכל זוג קבצים מקבילים מתוך enc1 ו-enc2 הצפינו את אותו ה- m , כאשר המספר האקראי השני הוא תוצר לינארי של המספר האקראי הראשון. כלומר, כל זוג ניתן לייצוג בתור:

$$(c_1 = g^k, c_2 = m \cdot h^k), (c'_1 = g^{ak+b}, c'_2 = m \cdot h^{ak+b})$$

המספר האקראי עצמו (k) מוגרל מחדש עבור כל הודעה ב-enc1, אך המספרים a ו- b שמהם נגזר המספר האקראי של enc2 נשארים קבועים לאורך כל ההודעות.

בהנתן הנחות אלו, עלינו לפצח את ההצפנה.

במהלך החישוב, אנחנו נשתמש בנגזרת [המשפט הקטן של פרמה](#) לפיו לכל מספר ראשוני p ולכל מספר שלם a מתקיים ש- $a^{p-1} \equiv 1 \pmod{p}$.

נתון לנו q (שהינו ראשוני), לכן אנחנו יכולים לחשב את $q - 1$. את $q - 1$ אנחנו יכולים לפרק למחלקים. נניח שמצאנו ש- d כלשהו הוא מחלק כלשהו של $q - 1$.

כעת נתבונן ב- g^k . את k , כמו כל מספר שלם אחר, אפשר לבטא בתור $k = d \cdot t + r$, וזאת כאשר מתקיים $r \in \{0 \dots d - 1\}$. לכן:

$$g^k = g^{dt+r}$$

נעלה את המספר בחזקת $\frac{q-1}{d}$ (שהוא מספר שלם כי d הוא מחלק של $q - 1$).

$$\begin{aligned} (g^k)^{\frac{q-1}{d}} &= (g^{dt+r})^{\frac{q-1}{d}} = g^{(dt+r) \cdot \frac{q-1}{d}} = g^{(dt) \cdot \frac{q-1}{d} + r \cdot \frac{q-1}{d}} = g^{(dt) \cdot \frac{q-1}{d}} \cdot g^{r \cdot \frac{q-1}{d}} \\ &= g^{(t) \cdot \frac{q-1}{1}} \cdot g^{r \cdot \frac{q-1}{d}} = (g^t)^{(q-1)} \cdot g^{r \cdot \frac{q-1}{d}} = (1 \pmod{q}) \cdot g^{r \cdot \frac{q-1}{d}} = g^{r \cdot \frac{q-1}{d}} \\ &= \left(g^{\frac{q-1}{d}}\right)^r \pmod{q} \end{aligned}$$



כמובן שכל החישובים הם $\text{mod } q$. כעת, מכיוון שידועים לנו כל המשתנים במשוואה הזו מלבד r , ו- r עצמו נמצא בתחום ידוע ($r \in \{0 \dots d - 1\}$), אנחנו יכולים להציב כל r אפשרי ולחשב את כל התוצאות האפשריות של $(g^{\frac{q-1}{d}})^r$. ומצד שני, אנחנו יכולים לחשב גם את $(g^k)^{\frac{q-1}{d}}$ שהוא בעצם $(c_1)^{\frac{q-1}{d}}$. מכאן אנחנו יכולים להסיק משהו על k :

If ...	Then ...
$k \text{ mod } d = 0$	$(c_1)^{\frac{q-1}{d}} = (g^{\frac{q-1}{d}})^0 = 1$
$k \text{ mod } d = 1$	$(c_1)^{\frac{q-1}{d}} = (g^{\frac{q-1}{d}})^1$
$k \text{ mod } d = 2$	$(c_1)^{\frac{q-1}{d}} = (g^{\frac{q-1}{d}})^2$
...	...
$k \text{ mod } d = d - 1$	$(c_1)^{\frac{q-1}{d}} = (g^{\frac{q-1}{d}})^{d-1}$

רק אחת מהמשוואות הללו תתקיים, וכך נדע מה השארית של k מחלוקה ב- d ($k \text{ mod } d$). את אותו תהליך אפשר להפעיל גם כדי להסיק מסקנה דומה על $ak + b$. בהנתן מספיק מסקנות כאלו, נוכל גם להסיק דברים על a ו- b עצמם.

עד כאן המתמטיקה היבשה. בפועל, על מנת למנוע [התקפה ידועה](#) של גילויי ביט בחזקה של g באמצעות אנליזה דומה לזו שעשינו זה עתה, נהוג להשתמש בערך של היוצר בריבוע ולא בערך של היוצר עצמו.

כלומר, אם נגדיר $\Gamma = g^2$, בפועל המספר שאנחנו מקבלים באתגר הינו $\Gamma^k = (g^2)^k = g^{2k}$ ולא $c_1 = g^k$. לכן אנחנו נחשב את $\Gamma^{\frac{q-1}{2d}}$ ולא את $(g^{\frac{q-1}{d}})^{\frac{q-1}{2d}} = (g^2)^{\frac{q-1}{2d}} = g^{2 \cdot \frac{q-1}{2d}} = g^{\frac{q-1}{d}}$ (מכיוון ש- $\Gamma^{\frac{q-1}{2d}} = (g^2)^{\frac{q-1}{2d}} = g^{2 \cdot \frac{q-1}{2d}} = g^{\frac{q-1}{d}}$).

נתחיל עם דוגמא. כדי לקרוא בנוחות את קבצי הקלט של התרגיל, נממש תחילה מודול קצר שיקרא אותם:

```
import os, glob
from pathlib import Path
from collections import namedtuple

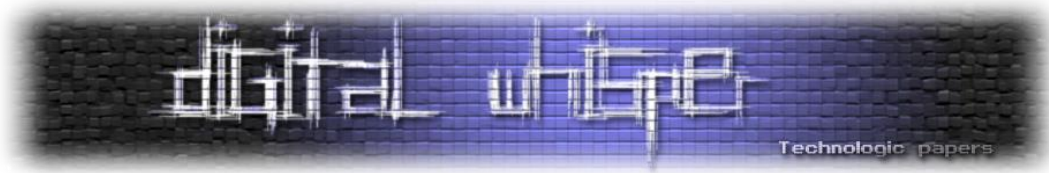
ElGamalPair = namedtuple('ElGamalPair', ['c1', 'c2'])

def read_data(base_path: str) -> dict:
    res = {}

    with open(os.path.join(base_path, "details.txt")) as f:
        res["q"] = int(f.readline().split(":")[1])
        res["g"] = int(f.readline().split(":")[1])
        res["h"] = int(f.readline().split(":")[1])

    for enc in ["enc1", "enc2"]:
        res[enc] = {}
        for infile in glob.glob(os.path.join(base_path, enc, '*.txt')):
            with open(infile) as f:
                line = f.readline().strip("\n")
                res[enc][int(Path(infile).stem)] = ElGamalPair._make([int(s) for s in line.split(", ")])
    return res
```

המודול הזה מאפשר לנו לקבל את הפרמטרים של התרגיל כמילון.



את q קיבלנו כנתון, נחשב את $q - 1$:

```
root@kali:/media/sf_CTFs/matrix/CamelCase# python3.8
Python 3.8.1 (default, Jan 19 2020, 22:34:33)
[GCC 9.2.1 20200117] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import data_reader
>>> params = data_reader.read_data("Camel")
>>> params["q-1"] = params["q"] - 1
>>> params["q-1"]
152871471657632351370080285966569305872910029684999383757718597560660696292043021501818591583896441093221262574940955568
908605354899083774467939670101882574152087670760348311597609704580106114746180425765515742210551990876619209582167486148
632819497795640798995338204874439231771371145977215524548029682689080
>>>
```

כעת נחפש את המחלקים של המספר הזה (למשל, באמצעות [האתר הזה](#)). התוצאה היא:

2, 2, 2, 3, 3, 5, 8357651, <a very large number...>

נתמקד במחלקים הקטנים משיקולי כוח-חישובי. בדוגמא שלנו, נשתמש ב-3.

כזכור, אנחנו הולכים לחשב את:

If ...	Then ...
$k \bmod d = 0$	$(c_1)^{\left(\frac{q-1}{d}\right)} = (\Gamma^{\left(\frac{q-1}{2d}\right)})^0 = 1$
$k \bmod d = 1$	$(c_1)^{\left(\frac{q-1}{d}\right)} = (\Gamma^{\left(\frac{q-1}{2d}\right)})^1$
$k \bmod d = 2$	$(c_1)^{\left(\frac{q-1}{d}\right)} = (\Gamma^{\left(\frac{q-1}{2d}\right)})^2$

נעשה זאת באמצעות הפונקציה:

```
def get_mod(params: dict, c1: int, num: int) -> int:
    assert(params["q-1"] % num == 0)
    t = pow(c1, params["q-1"] // num, params["q"])
    for i in range(num):
        if t == pow(params["g"], (params["q-1"] // (num * 2)) * i, params["q"]):
            return i
    raise RuntimeError(f"Can't find result for {num}")
```

נריץ לדוגמא על ההודעה השנייה שניתנה לנו:

```
>>> get_mod(params, params["enc1"][1].c1, 3)
1
>>> get_mod(params, params["enc2"][1].c1, 3)
2
```

קיבלנו שעבור הודעה מספר 2, מתקיים:

$$k \bmod 3 = 1$$

וגם:

$$(ak + b) \bmod 3 = 2$$



אנחנו יכולים להמשיך לעשות אותו דבר עבור כל שילוב של מחלקים, ולקבל עוד נתונים:

```
k[1]      % 4      == 0
(a*k[1] + b) % 4      == 0

k[1]      % 45     == 28
(a*k[1] + b) % 45     == 17

k[1]      % 60     == 28
(a*k[1] + b) % 60     == 32

k[1]      % 10     == 8
(a*k[1] + b) % 10     == 2

k[1]      % 18     == 10
(a*k[1] + b) % 18     == 8

k[1]      % 9      == 1
(a*k[1] + b) % 9      == 8

k[1]      % 30     == 28
(a*k[1] + b) % 30     == 2

k[1]      % 3      == 1
(a*k[1] + b) % 3      == 2

k[1]      % 15     == 13
(a*k[1] + b) % 15     == 2

k[1]      % 12     == 4
(a*k[1] + b) % 12     == 8

k[1]      % 90     == 28
(a*k[1] + b) % 90     == 62

k[1]      % 36     == 28
(a*k[1] + b) % 36     == 8

k[1]      % 6      == 4
(a*k[1] + b) % 6      == 2
```

עכשיו שימו לב לתופעה מעניינת: כאשר השאריות שונות מאפס, הסכום שלהן הוא המספר שאנחנו בודקים. התוצאה הזו חוזרת בכל הדגימות. כבר מפה אפשר לנחש מהם a ו- b , אבל אנחנו נמשיך עוד צעד ונדחוף הכל לתוך z_3 בשביל הכיף.

הנה הסקריפט:

```
import data_reader
import functools, operator, argparse, sys, random
from itertools import chain, combinations

def powerset(iterable):
    "powerset([1,2,3]) --> () (1,) (2,) (3,) (1,2) (1,3) (2,3) (1,2,3)"
    s = list(iterable)
    return chain.from_iterable(combinations(s, r) for r in range(1, len(s)+1))

def get_mod(params: dict, c1: int, num: int) -> int:
    assert(params["q-1"] % num == 0)
    t = pow(c1, params["q-1"] // num, params["q"])
    for i in range(num):
        if t == pow(params["g"], (params["q-1"] // (num * 2)) * i, params["q"]):
            return i
```



```
raise RuntimeError(f"Can't find result for {num}")

def find_small_factors(n):
    small_factors = []
    t = n
    for i in range(2, 10):
        while (t % i == 0):
            small_factors.append(i)
            t = t // i
    return small_factors

def analyze(params: dict, solve_ctx: dict):
    verbose = True
    if solve_ctx is not None:
        try:
            import z3
        except:
            raise RuntimeError("Error: Z3 module needed")
        verbose = solve_ctx.get("verbose", False)

    params["q-1"] = params["q"] - 1
    small_factors = find_small_factors(params["q-1"])
    if verbose: print(f"Small factors: {small_factors}\n")

    assert(len(params["enc1"]) == len(params["enc2"]))

    samples = list(range(len(params["enc1"])))
    if solve_ctx is not None:
        solver = z3.Solver()
        a = z3.Int('a')
        b = z3.Int('b')
        samples = random.sample(samples, k=solve_ctx.get("num_samples", 3))
        k = {i: z3.Int('k{}'.format(i)) for i in samples}

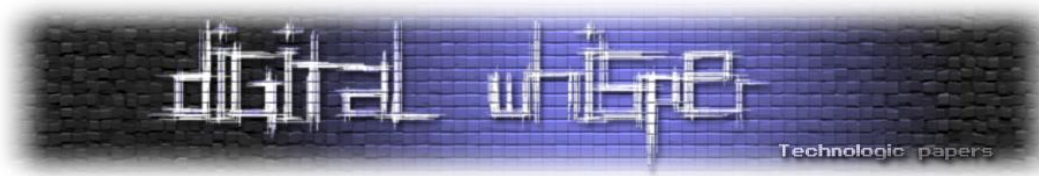
    for i in samples:
        if solve_ctx is not None:
            solver.add(k[i] >= 0)
            solver.add(k[i] < params["q"])
        for combination in set(powerset(small_factors)):
            factor = functools.reduce(operator.mul, combination)
            try:
                m = get_mod(params, params["enc1"][i].c1, factor)
                if verbose: print(f"    k[{i}] % {factor} \t== {m}")
                if solve_ctx is not None: solver.add(k[i] % factor == m)

                m = get_mod(params, params["enc2"][i].c1, factor)
                if verbose: print(f"(a*k[{i}] + b) % {factor} \t== {m}\n")
                if solve_ctx is not None: solver.add((a * k[i] + b) % factor ==
m)

            except RuntimeError:
                pass

    if solve_ctx is not None:
        print(f"Trying to solve with {len(samples)} samples {samples}: ", end =
'', flush = True)
        if solver.check() == z3.sat:
            m = solver.model()
            #print(m)
            print("a = {}, b = {}".format(m[a], m[b]))
        else:
            print("Can't find solution")

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
```



```
parser.add_argument('-s', '--
solve', action='store', type=int, default=0, metavar=('NUM_SAMPLES'), help='Try
to solve constraints using up to <NUM_SAMPLES> samples')
parser.add_argument('--
path', default="Camel", type=str, help='Path to data folder')
args = parser.parse_args()

try:
    params = data_reader.read_data(args.path)
    if args.solve:
        solve_ctx = {"num_samples": args.solve}
    else:
        solve_ctx = None
    analyze(params, solve_ctx)
except RuntimeError as e:
    print (str(e))
    sys.exit(1)
```

את הסקריפט ניתן להריץ בשני אופנים. הרצה ללא פרמטרים תדפיס את השאריות עבור כל ההודעות כפי שראינו קודם. אולם, אם נריץ את הסקריפט עם פרמטר של מספר דגימות, הוא ינסה לפתור את המשוואה עבור מספר זה של דגימות אקראיות באמצעות \mathbb{Z}_3 (אנחנו מגבילים את מספר הדגימות משיקולי כוח חישוב).

```
root@kali:/media/sf_CTFs/matrix/CamelCase# python3 analyze.py -s 3
Trying to solve with 3 samples [9, 10, 23]: a = -1, b = 0
root@kali:/media/sf_CTFs/matrix/CamelCase# python3 analyze.py -s 3
Trying to solve with 3 samples [22, 1, 18]: ^CCan't find solution
root@kali:/media/sf_CTFs/matrix/CamelCase# python3 analyze.py -s 3
Trying to solve with 3 samples [8, 11, 24]: a = -1, b = 0
root@kali:/media/sf_CTFs/matrix/CamelCase# python3 analyze.py -s 3
Trying to solve with 3 samples [10, 21, 5]: a = -1, b = 0
root@kali:/media/sf_CTFs/matrix/CamelCase# python3 analyze.py -s 3
Trying to solve with 3 samples [4, 6, 21]: ^CCan't find solution
root@kali:/media/sf_CTFs/matrix/CamelCase# python3 analyze.py -s 3
Trying to solve with 3 samples [2, 5, 13]: a = -1, b = 0
root@kali:/media/sf_CTFs/matrix/CamelCase# python3 analyze.py -s 3
Trying to solve with 3 samples [2, 16, 19]: a = -1, b = 0
root@kali:/media/sf_CTFs/matrix/CamelCase# python3 analyze.py -s 3
Trying to solve with 3 samples [10, 16, 11]: a = -4321, b = -1526580
```

אנחנו רואים פה מספר הרצות עם 3 דגימות - מספר הדגימות הגבוה ביותר שמאפשר פתרון בזמן סביר.

כפי שניתן לראות - בהרבה מאוד מקרים התוצאה היא: $a = -1, b = 0$. זה גם מסתדר עם התופעה שראינו קודם, שבה סכום השאריות שווה למספר שאותו אנו בודקים (למעשה עברנו לחבורה ציקלית חדשה שיש בה d איברים מתוך \mathbb{Z}_q^* , ומכיוון שזוהי חבורה ציקלית, תכונת המודולו עדיין נשמרת מתוך

$$\text{איברי החבורה: } (k \bmod d \xrightarrow{\text{yields}} -k \bmod d = d - k)$$

כעת, כשמצאנו מועמדים טובים עבור a ו- b , ננסה לפצח את ההצפנה. אנחנו יודעים ש:

$$c_2 = m \cdot h^k \bmod q$$
$$c'_2 = m \cdot h^{a+b} = m \cdot h^{-k} \bmod q$$



נכפיל אותם אחד בשני ונקבל:

$$c_2 \cdot c_2' = (m \cdot h^k) \cdot (m \cdot h^{-k}) = m \cdot m \cdot h^k \cdot h^{-k} = m^2 \cdot h^{k+(-k)} = m^2 \cdot h^0 = m^2 \bmod q$$

ניחשנו שכל הודעה מכילה תו אחד בלבד מהדגל ואכן לפי התוצאות שקיבלנו ניתן לראות ש- m^2 לא גדול מ- q ולכן מספיק להוציא שורש ולקבל את m .

הסקריפט הבא עושה זאת:

```
import data_reader
import math

if __name__ == "__main__":
    params = data_reader.read_data("Camel")
    res = ""

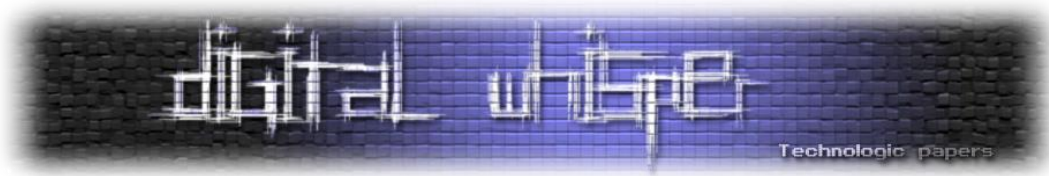
    assert(len(params["enc1"]) == len(params["enc2"]))

    for i in range(len(params["enc1"])):
        m = math.isqrt(params["enc1"][i].c2 * params["enc2"][i].c2 % params["q"])
        res += chr(m)

    print(res)
```

התוצאה:

```
root@kali:/media/sf_CTFs/matrix/CamelCase# python3.8 decrypt.py
FLAG__CRYPTANALYSIS_ROCKS
```

אתגר 7: Save Israel (200 נקודות)

Challenge ×

Save Israel
200

According to an intel we received, Iran managed to assemble nuclear warheads onto an intercontinental ballistic missiles while evading all of the west intelligence agencies. Israeli intelligence agents succeeded to put their hands on a file from the PC of none other than Iran's supreme leader, Ali Khamenei. They suspect that the code required to neutralize the nuclear warheads is in this file but they failed retrieving the code from the file. Therefore, they ask for your help. The fate of the state of Israel rests on your shoulders. Good luck to all of us.

printflag

Flag Submit

פתרון:

נריץ את הקובץ המצורף:

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# file printflag
printflag: ELF 32-bit LSB executable, Intel 80386, version 1 (GNU/Linux), statically linked, stripped
root@kali:/media/sf_CTFs/matrix/Save_Israel# ./printflag
80eaf9c
```

הפלט היחיד הוא 80eaf9c. זה נראה כמו מספר בייצוג הקסאדסימלי. אך מה משמעותו?

אם נפתח את הקובץ בדיסאסמבלר, נגלה שהוא קשה מאוד לקריאה. בנוסף, הקובץ הוא stripped, כלומר כזה שהוסר ממנו מידע שמסייע לדיבוג. אפשר לפתוח את הקובץ עם דיבאגר ולעבור פקודה אחרי פקודה, אבל זה לא פחות קשה. במקום זאת, ננסה לקבל סיכום ממבט עילי (יחסית) של הפקודות שהתוכנה מריצה.

תחילה, נריץ את התוכנה באמצעות strace - כלי עזר שמפרט את ה-system calls שהתוכנה משתמשת בהם בזמן ריצה:

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# strace ./printflag
execve("./printflag", ["/printflag"], 0x7fff4cf4bbd0 /* 22 vars */) = 0
strace: [ Process PID=1001 runs in 32 bit mode. ]
mmap(0xc48000, 4096, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0xc48000) = 0xc48000
readlink("/proc/self/exe", "/media/sf_CTFs/matrix/Save_Israel...", 4096) = 43
mmap(0x8048000, 656148, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x8048000
mprotect(0x8048000, 656145, PROT_READ|PROT_EXEC) = 0
mmap(0x80e9000, 8067, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x80e9000
mprotect(0x80e9000, 8064, PROT_READ|PROT_WRITE) = 0
mmap(0x80eb000, 3492, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x80eb000
brk(0x80ec000) = 0x97d2000
munmap(0xc02000, 290816) = 0
uname({sysname="Linux", nodename="kali", ...}) = 0
brk(NULL) = 0x97d2000
brk(0x97d2d40) = 0x97d2d40
set_thread_area({entry_number=-1, base_addr=0x97d2840, limit=0x0ffffff, seg_32bit=1, contents=0, read_exec_only=0, limit_in_pages=1, seg_not_present=0, useable=1}) = 0 (entry_number=12)
readlink("/proc/self/exe", "/media/sf_CTFs/matrix/Save_Israel...", 4096) = 43
brk(0x97f3d40) = 0x97f3d40
brk(0x97f4000) = 0x97f4000
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
fstat64(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(136, 0), ...}) = 0
write(1, "80eaf9c\n", 80eaf9c
) = 8
exit_group(0) = ?
+++ exited with 0 +++
```



אפשר לראות שהתוכנה קוראת ל-mmap ול-mprotect על טווח שמכיל את הערך המסתורי שלנו. בואו ננסה לשים שם breakpoint:

```
(gdb) start
Starting program: /media/sf_CTFs/matrix/Save_Israel/printflag

Program stopped.
0x00c46a10 in ?? ()
(gdb) b *0x80eaf9c
Breakpoint 1 at 0x80eaf9c
(gdb) c
Continuing.
Warning:
Cannot insert breakpoint 1.
Cannot access memory at address 0x80eaf9c

Command aborted.
(gdb)
```

זה לא עובד, לא ניתן לגשת לזיכרון בכתובת זו.

נמשיך לנסות לקבל מושג טוב יותר מה התוכנה עושה. לשם כך, נריץ באמצעות gdb את הסקריפט הבא:

```
set style enabled off
set pagination off
set logging on
set disassembly-flavor intel
set disassemble-next-line on
start
define nstep
set $limit = $arg0
while ($limit--)
    ni
end
nstep 40000
```

בגדול, הסקריפט הזה מגדיר פונקציה שנקראת nstep אשר מבצעת את הפקודה ni (כלומר next instruction) מספר מוגדר של פעמים, ומדפיסה את ה-instruction לקובץ. אנחנו קוראים לפונקציה זו בבקשה להריץ 40,000 פקודות.

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# gdb -nh ./printflag -x trace.gdb
```

את התוצאה נוכל לראות בקובץ gdb.txt. הקובץ המתקבל בעל כ-60,000 שורות. כך הוא מתחיל:

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# cat gdb.txt | head

Program stopped.
0x00c46a10 in ?? ()
=> 0x00c46a10: e8 8b 02 00 00 call 0xc46ca0
0x00c46ca0 in ?? ()
=> 0x00c46ca0: 5d      pop    ebp
0x00c46ca1 in ?? ()
=> 0x00c46ca1: e8 ad ff ff ff call 0xc46c53
0x00c46c53 in ?? ()
=> 0x00c46c53: 5e      pop    esi
```



וכך הוא מסתיים:

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# cat gdb.txt | tail
=> 0x0804e313: e8 59 e5 01 00 call 0x806c871
0x0806c871 in ?? ()
=> 0x0806c871: 8b 5c 24 04 mov ebx,DWORD PTR [esp+0x4]
0x0806c875 in ?? ()
=> 0x0806c875: b8 fc 00 00 00 mov eax,0xfc
0x0806c87a in ?? ()
=> 0x0806c87a: ff 15 f0 a9 0e 08 call DWORD PTR ds:0x80ea9f0
[Inferior 1 (process 1051) exited normally]
trace.gdb:13: Error in sourced command file:
The program is not being run.
```

אפשר לראות שהתוכנה מתחילה מריצה במרחב 0x00CXXXXX ומסתיימת בריצה במרחב 0x08XXXXXX. אם כך אולי הכתובת המסתורית הופכת להיות זמינה רק מאוחר יותר? זה מסתדר עם הקריאה ל-mmap שראינו קודם. נחפש את הכתובת המסתורית בתוך הלוג שלנו:

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# cat gdb.txt | grep -i 80eaf9c
root@kali:/media/sf_CTFs/matrix/Save_Israel#
```

אין מזל. אולי ניתן לתוכנה לרוץ קצת, ורק אז ננסה לשים breakpoint?

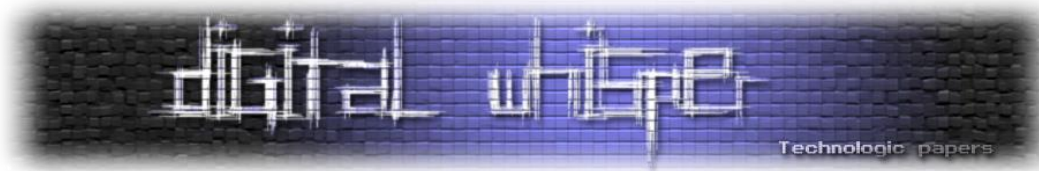
```
(gdb) starti
Starting program: /media/sf_CTFs/matrix/Save_Israel/printflag

Program stopped.
0x00c46a10 in ?? ()
(gdb) ni 28000
0x0806c1f2 in ?? ()
(gdb) b *0x80eaf9c
Breakpoint 1 at 0x80eaf9c
(gdb) c
Continuing.
80eaf9c

FLAG_taylorswift4evah<3_
[Inferior 1 (process 1099) exited normally]
```

זה עבד! וקיבלנו את הדגל! רגע, מה? איך ריצה עם breakpoint פתאום הדפיסה לנו את הדגל?

כדי לענות על השאלה הזאת, ננסה לנתח בכל זאת את התוכנה עם דיסאסמבלר. ממה שראינו עד עכשיו, התוכנה מתנהגת באופן שמזכיר מאוד [packer](http://www.packer.org/) - תוכנה שמסתירה בתוכה תוכנה נוספת, כאשר היא פורסת ומריצה אותה בזמן ריצה. כלומר, הקוד שאנחנו רואים כשפותחים את התוכנה בדיסאסמבלר הוא קוד שנועד לפרוס ולהריץ תוכנה אחרת שמסתתרת אי שם בבינארי. אותנו לא מעניינת התוכנה החיצונית, לכן אנחנו צריכים להגיע אל התוכנה הפנימית שהיא העיקר.



ישנם packer-ים רבים אך המפורסם שבהם הוא UPX, לכן קודם ננסה לפרוס את התוכנה באמצעות UPX:

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# upx -d printflag
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2018
UPX 3.95      Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

File size      Ratio      Format      Name
-----
upx: printflag: NotPackedException: not packed by UPX

Unpacked 0 files.
```

זה לא עבד. אם כך, ננסה להשתמש בכלי-עזר שאמור לזהות את ה-packer שבו נעשה שימוש, אם הוא מוכר:

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# ~/utils/die/die_lin64_portable/diec.sh printflag
ELF: packer: UPX(3.X)[executable 386-32]
ELF: compiler: gcc((Ubuntu 5.4.0-6u~16??? 5)/ 20 089)[executable 386-32]
```

מעניין, [Detect it Easy](#) דווקא טוען בתוקף שכן מדובר ב-UPX. אז מי צודק?
מחקר קצר גילה שקל מאוד לגרום ל-UPX להכשל בפריסת קובץ שהוא עצמו יצר, כל מה שצריך זה מספר שינויים קלים בבינארי. הקובץ עצמו ירוץ כשורה, אך לא יהיה אפשר לחלץ ממנו את קובץ ההרצה הפנימי. איך אפשר להתגבר על בעיה זו? UPX הוא כלי קוד-פתוח. אפשר להוריד את הקוד שלו, להוסיף הדפסות בכל מקום שבו הוא יוצא עם כישלון, ולהבין מה בדיוק מפריע לו בבינארי שלנו. אם נעשה זאת, נקבל שהיה צריך לשנות את המקומות הבאים:

00000070 00 10 00 00 D4 BF 17 09 85 01 89 80 34 08 00 0C0z...%4...	00000070 00 10 00 00 D4 BF 17 09 55 50 58 21 34 08 00 0C0z...UPX!4...
0004C7C0 EF 10 00 67 9F 92 90 DA C7 54 9E 5E 6D 9F A0 92 i..gY'.UcTz^mY '	0004C7C0 EF 10 00 67 9F 92 90 DA C7 54 9E 5E 6D 9F A0 92 i..gY'.UcTz^mY '
0004C7D0 24 49 FF 00 00 00 00 D2 5D E0 96 00 00 00 00 \$Iy....0]a-.....	0004C7D0 24 49 FF 00 00 00 00 55 50 58 21 00 00 00 00 \$Iy....UPX!.....
0004C7E0 49 D8 80 E0 00 0C 08 07 9F 29 2B 56 94 0A 45 8E IUea....Y)+v^..Ez	0004C7E0 55 50 58 21 00 0C 08 07 9F 29 2B 56 94 0A 45 8E UPX!....Y)+v^..Ez
0004C7F0 08 F2 00 00 71 65 00 00 88 11 08 00 49 13 00 CB .d..qe...^...I..E	0004C7F0 08 F2 00 00 71 65 00 00 88 11 08 00 49 13 00 CB .d..qe...^...I..E
0004C800 80 00 00 00 00 00 00 00 00 00 00 00 00 00 e...	0004C800 80 00 00 00 00 00 00 00 00 00 00 00 00 00 e...

ננסה לחלץ:

```
root@kali:/media/sf_CTFs/matrix/Save_Israel# upx -d printflag_fix_upx -o printflag_out_upx
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2018
UPX 3.95      Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

File size      Ratio      Format      Name
-----
725384 <-    313348    43.20%    linux/i386    printflag_out_upx

Unpacked 1 file.
```

הקובץ חולץ בהצלחה! כעת ניתן לפתוח אותו עם Ghidra.
כעת מסתבר שהכתובת המסתורית שלנו היא משתנה גלובלי:

target	XREF[3]:	Entry Point(*), printf:080488e0(*), printf:08048900(R)
080eaf9c	undefined4 ??	

נעקוב אחרי השימוש שלה ונקבל:

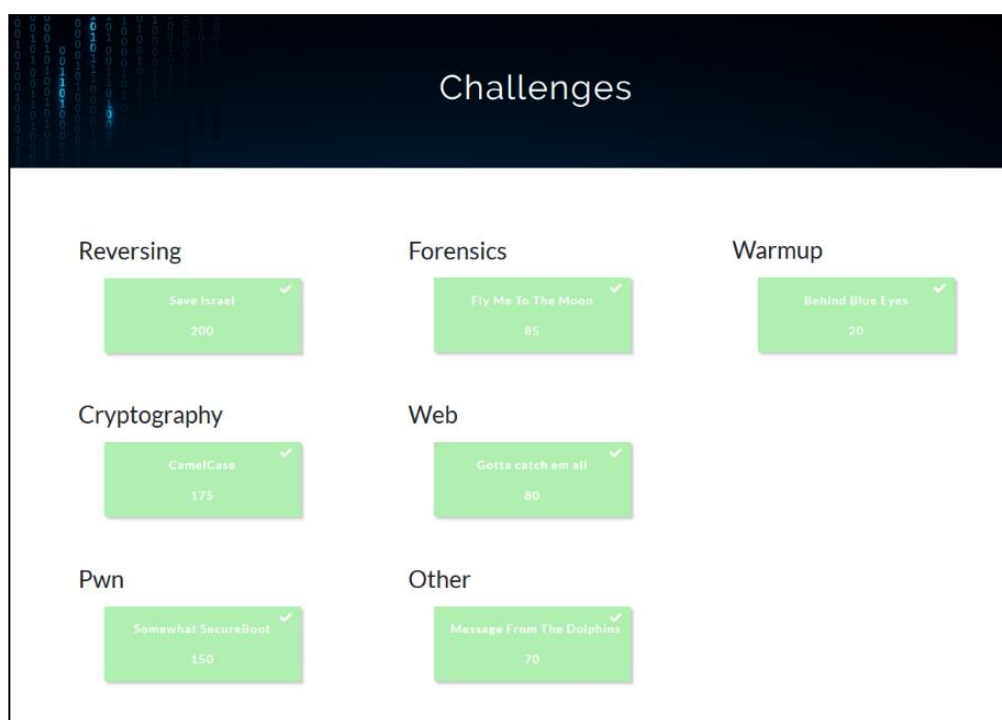
```
printf("%x\n",&target);
printf(param_1);
if (target != 0) {
    local_29 = (EVP_PKEY_CTX)0x46;
    local_28 = 10;
    local_27 = 0x4b;
    local_26 = 0xc;
    local_25 = 0x53;
    local_24 = 0x27;
    local_23 = 0x46;
    local_22 = 0x3f;
    local_21 = 0x53;
    local_20 = 0x3c;
    local_1f = 0x4e;
    local_1e = 0x3d;
    local_1d = 0x4a;
    local_1c = 0x23;
    local_1b = 0x45;
    local_1a = 0x31;
    local_19 = 5;
    local_18 = 0x60;
    local_17 = 0x16;
    local_16 = 0x77;
    local_15 = 0x1f;
    local_14 = 0x23;
    local_13 = 0x10;
    local_12 = 0x4f;
    local_11 = 0;
    iVar1 = decrypt(&local_29,&DAT_00000018,
    printf("\n%s\n",iVar1);
```

כלומר, התוכנה מדפיסה את הכתובת של המשתנה, ומיד לאחר מכן בודקת אם הוא שווה ל-0. מכיוון שהצבת breakpoint בכתובת כלשהי משנה זמנית את התוכן של הזיכרון שלו ל-0xCC, רק כאשר הצבנו breakpoint ראינו את ההדפסה. התעלומה נפתרה!

סיכום

ה-CTF הזה כלל שבעה אתגרים ממספר תחומים מגוונים. על אף שהוא נבנה ברמת קושי עולה, רוב השאלות בסדרת האתגרים הזו נחשבות יחסית קלות ביחס ל-CTF-ים אחרים.

הקושי העיקרי ב-CTF היה בשאלת ה-Crypto, שהצריכה אנליזה מתמטית לא-טריוויאלית. משיטוט בקבוצות ה-CTF הייעודיות, נראה היה שזהו האתגר שרוב המשתמשים חיפשו בו רמז או הכוונה, והוא בעצם היווה את הטריגר העיקרי לכתיבת הפתרון כולו. מהחיפוש שעשינו, לא מצאנו writeup אחר אודות הדרך לתקוף הצפנה חוזרת של הודעה בסכמת ElGamal תוך שימוש ב-[LCG](#) עבור בחירת המשתנה האקראי כאשר a ו- b לא ידועים.



תודה רבה למארגנים על הכנת ה-CTF ועל השארתו פתוח למשך תקופה ארוכה ביותר, מה שאיפשר את העבודה עליו בצורה נוחה ואת הכנת פתרון זה.

תודה גם ל-zVaz ול-DHG על תרומתם במהלך פתרון ה-CTF.